INGENIERIE DES SYSTEMES D'INFORMATION :
MERISE DEUXIEME GENERATION

4°édition - 2001

Dominique NANCI - Bernard ESPINASSE

Avec la collaboration de Bernard COHEN, Jean-Claude ASSELBORN et Henri HECKENROTH

Dominique NANCI - Bernard ESPINASSE

Avec la collaboration de Bernard COHEN, Jean-Claude ASSELBORN et Henri HECKENROTH

INGENIERIE DES SYSTEMES D'INFORMATION : MERISE DEUXIEME GENERATION

4°édition - 2001

Cet ouvrage a été publié jusqu'en 2010 par les éditions VUIBERT, collection informatique. Les auteurs ayant récupéré leurs droits, il est maintenant disponible en téléchargement gratuit, pour qu'il serve de témoignage de cette belle aventure qu'aura été le développement et l'évolution de cette méthode Merise, et afin que son enseignement et sa mise en œuvre soit toujours possible.

Les auteurs, mars 2014.

Référence de l'ouvrage à citer pour tout emprunt :

Nanci D., B. Espinasse avec la collaboration de B. Cohen, J.C. Asselborn et H. Heckenroth (2001), « Ingénierie des systèmes d'information : Merise deuxième génération », Vuibert éditions, Paris. ISBN : 2-7117-8674-9 (416 pages).

Impression du document :

Le formatage de ce document en pdf a été fait pour permettre une impression en 2 pages par page A4 avec un agrandissement de 135% sous Acrobat Reader.

"Construire existe entre un projet ou une vision déterminée, et les matériaux que l'on a choisis. On substitue un ordre à un autre qui est initial, quels que soient les objets qu'on ordonne. Ce sont des pierres, des couleurs, des mots, des concepts, des hommes, etc. Leurs nature particulière ne change pas les conditions générales de cette sorte de musique où elle ne joue encore que le rôle du timbre..."

Paul Valery

Introduction à la méthode de Léonard De Vinci.

Sommaire général

Remerciements

Comme reviennent à nos esprits à chacun les moments forts que nous avons passés ensemble, voilà près d'une vingtaine d'années et sous leur stimulante animation, nous remercions encore Jean-Louis Le Moigne et Hubert Tardieu qui ont bien voulu préfacer cet ouvrage.

Nous remercions aussi tous les contributeurs à la méthode Merise, souvent anonymes, qui ont fait que cette méthode s'enrichisse et évolue au fil du temps et de la complexification constante de l'ingénierie des systèmes d'information.

Enfin, la gestation, la naissance et l'actualisation d'un tel ouvrage ne se font pas sans quelques petits sacrifices et nous n'oublions pas la confiance et le soutien de nos institutions respectives, ainsi que la patience compréhensive de nos entourages familiaux.

Avant-propos à la 4°édition

Cet ouvrage a été publiée dans sa première édition en 1992 chez Sybex. Une deuxième édition (1994) en a permis une révision mineure. La troisième édition, profondément remaniée et augmentée a permis une réactualisation majeure de l'ouvrage.

De nombreux témoignages d'enseignants mais aussi de professionnels concepteurs de systèmes d'information nous ont convaincus que ce livre avait toujours sa place dans la communauté francophone et a conduit à cette quatrième édition publiée chez Vuibert.

Nous espérons que tous les "merisiens" d'hier, d'aujourd'hui et de demain trouveront dans cette nouvelle édition un témoignage de la réelle volonté d'ouverture et de la vivacité des réflexions et contributions qui permettent à la méthode Merise d'être toujours d'actualité dans le domaine de l'ingénierie des systèmes d'information.

A ce jour, cette quatrième édition est le livre le plus complet et le plus actualisée sur la méthode Merise. Cette méthode de conception de système d'information est largement diffusée en France, où elle fait office de standard, mais aussi dans une bonne partie de l'Europe où elle est adoptée. Près de vingt ans de pratique dans des projets opérationnels en conception de systèmes d'information et autant d'années en formation dans ce domaine ont permis d'affiner cet ouvrage écrit par des hommes de terrain et des enseignants chercheurs reconnus. La méthode MERISE y est présentée de façon originale et pertinente selon ses trois grandes composantes principales : les modèles et raisonnements, la démarche préconisée et enfin l'organisation des moyens associés. La méthode ainsi que ses plus récentes évolutions y sont traités de façon claire, précise, rigoureuse mais aussi pratique.

Cette édition se caractérise par une actualisation des différentes composantes de la méthode MERISE et de nouveaux développements relatifs à l'objet métier (problématique de la réutilisation), au BPR (Business Process Reengineering) et a un positionnement par rapport à UML (Unified Modeling Language).

Ce livre s'adresse aussi bien à des professionnels, concepteurs de systèmes d'information, qu'à des étudiants de second et troisième cycles universitaires, écoles d'ingénieur ou de commerce, désirant mieux maîtriser cette méthode. Il est depuis plusieurs années le support de séminaires de formation intra-entreprise et de nombreux programmes universitaires, les écoles d'ingénieurs, et formations spécialisées en systèmes d'information.

Les auteurs

Dominique Nanci est directeur technique de CECIMA, il a en charge l'activité Ingénierie des systèmes d'information.

Bernard Espinasse est professeur d'informatique à Aix-Marseille Université, à l'Ecole Polytechnique Universitaire de Marseille, et chercheur au laboratoire LSIS UMR CNRS, où il a dirigé pendant plus de quinze ans une équipe de recherche et dirigé une quinzaine de thèses.

Bernard Cohen est directeur général fondateur de CECIMA, il a en charge l'activité Génie logiciel.

Jean-Claude Asselborn est professeur d'informatique au centre universitaire à Luxembourg et chef de projet dans la recherche publique.

Henri Heckenroth mène une activité de consultant auprès de grands comptes dans le domaine des systèmes d'information, de la qualité du logiciel, de l'organisation et du management de programme.

Hommage

Dominique Nanci nous a quitté en 2012, nous lui dédions ce livre, à lui sans qui toute cette aventure n'aurait eu lieu.

Jean-Claude Asselborn est décédé en 2011, nous garderons de lui et de notre collaboration un souvenir ému.

Préface à la 1° édition (1992)

Méthode ? L'étymologie du mot est incertaine, mais elle tient pour plausible l'association de deux racines, celle qui dit le raisonnement rusé ou les ruses de l'intelligence, "la Métis" (1), et celle qui dit le chemin, la voie à suivre, "Hodos" . La méthode ou les chemins de la ruse? Cette définition vous conviendra si vous "cherchez une méthode" ... Quel qu'en soit le projet : "Chercher une méthode, disait P. Valéry (dans "Variétés"), c'est chercher un système d'opérations extériorisables qui fasse mieux que l'esprit le travail de l'esprit" . Mais vous la contesterez peut-être, si vous tentez de vendre ou d'enseigner une méthode que vous avez trouvée antérieurement? Les ruses du concepteur de méthodes peuvent-elles être autorisées à l'utilisateur de ses méthodes? C'est l'unicité et la cohérence visible du cheminement qui alors semblent importer, et non plus la multiplicité des stratégies et des tactiques rusées susceptibles d'être mises en oeuvre. Dès que possible, ne faudra-t-il pas transformer la Méthode en une Logique, les raisonnements dialectiques et rhétoriques en raisonnements syllogistiques, voire en sophismes? On disposera alors d'un énoncé unique, normé et enseignable, incontestable puisque déductivement rationnel, indifférent à la diversité des esprits qui s'y réfèrent?

Tension familière aux concepteurs de méthodes, quel que soit le projet qui les motive : "Le système de conception d'une méthode" ("chercher une méthode") doit-il être "la méthode de conception d'un système" (mettre en oeuvre une méthode)? Et la méthode que trouvera P. Valéry au terme de sa recherche ne séduit guère le concepteur puisqu'elle s'exprime en deux lignes seulement "et je développais une méthode sans lacune. Où? Pour quoi? Pour qui? A quelle fin? De quelle grandeur? ("Eupalinos ou l'architecte"). Car si cette méthode est sans lacune, elle incite pourtant plus au questionnement rusé qu'à la nécessaire déduction; et elle ne permet guère à l'expert d'affirmer son autorité par sa maîtrise des procédures enchevêtrées qu'il a conçues et que lui seul sait démêler!

Mais pour les entreprises contemporaines, la qualité formelle de la méthode importe moins que les qualités "médiatiques" des acteurs qui sont censés la mettre en oeuvre : c'est par ces qualités cachées que s'expriment les ruses de la méthode lorsque la Méthode, devenant "Logique", ne prétend plus être rusée. C'est par leurs qualités médiatiques plus que par leurs vertus ratiocinatrices que les méthodes MERISE rendent ou peuvent rendre aujourd'hui services aux organisations qui s'efforcent de maîtriser leur propre processus auto-informationnel, avec ou sans l'aide des nouvelles technologies de l'information.

---

(1) - M. Detienne et J.P. Vernant : "Les ruses de l'intelligence, la Métis des Grecs". Ed. Flammarion, Coll. Champ, 1974.

Ces qualités médiatiques se révèlent par l'expérience des médiateurs, par leur sens de l'organisation et leur compréhension de la symbolisation par laquelle l'entreprise se décrit. "Un système intelligent peut et doit observer son propre comportement" argumentait Jacques Pitrat il y a peu (2). C'est cette capacité à observer (et donc à symboliser) délibérément son comportement qui permet à l'entreprise de se piloter dans des contextes en permanentes transformations. Il importe dès lors que des "médiateurs" l'assistent dans cet exercice complexe d'autoreprésentation. La qualité de leur médiation révèle leur capacité à ruser plus que la puissance intrinsèque de la méthode à laquelle ils se réfèrent. Mais sans l'exposé loyal et vivant de cette méthode, pourraient-ils apprendre à ruser et aider ainsi l'entreprise à "apprendre à s'observer dans ses activités et ses projets". Apprentissage que longtemps interdirent les triomphantes "méthodes d'analyse des applications", mais que peu à peu rendent possibles puis praticables les méthodes systémiques de conception . Dans leur exposition, la plupart des méthodes de la famille MERISE (et des familles apparentées, anglo-saxones ou européennes (3)), semblent souvent imprégnées encore par la culture analytique dont l'informatique et l'organisation dite scientifique ont longtemps fait leur Principe. Il est toujours difficile de se réaccoutumer "à la méthode de Léonard de Vinci" (P. Valéry, 1884), lorsqu'on a été élevé dans le culte de la méthode cartésienne (1637). C'est pourquoi, plus que par l'exposé sur la méthode, c'est par la pratique médiatisante de la méthode, que les responsables évaluent aujourd'hui la capacité de leur organisation à s'auto-observer...; et donc à concevoir intelligemment son système d'information. Cette pratique ici est certifiée par les quinze années d'expérience, dans tant de situations différentes, des auteurs de cet ouvrage qui vient compléter une production déjà riche d'une vingtaine de titres sur la méthode Merise.

Le "noyau" de MERISE ne fut-il pas établi entre 1974 et 1978 par une équipe d'ingénieurs et de chercheurs aixois (3) : on vérifiera la pérennité de ce noyau en relisant le livre que publièrent en 1979, H. Tardieu, et D. Nanci et D. Pascot, que préfaçait Jean-Louis Le Moigne : "La conception du système d'information, construction de la base de données" (Ed. d'Organisation, Paris, 1979). Plus tard, avec le concours du ministère de l'industrie qui souhaitait disposer d'une méthode standard pour organiser les rapports contractuels entre l'administration et ses sous-traitants, fut développée MERISE à partir du "noyau" aixois. L'arrêt brutal et inopiné des financements publics à la fin des années 70 obligea les sociétés de service qui avaient contribué à la création de

---

(2) - Titre de sa contribution à COGNITIVA 90 (Actes AFCET-Cognitiva, Madrid, Paris, 1990, pp. 337-346, vol.1).

(3) - T.W. Olle et al. Eds. : "Information Systems Design : Methodologies : A Comparative Review (1983) et "ISDH, A Feature Analysis" (1983). North Holland Pub. Cy.

(3) - L'équipe d'Aix comprenait alors, sous la responsabilité conjointe de J.L. LE MOIGNE et de H.TARDIEU : D.NANCI, H.HECKENROTH, auxquels s'étaient joints D.PASCOT puis B.ESPINASSE.

MERISE à reprendre le flambeau ; ce fût aussi l'initiative privée des trois auteurs originaux de MERISE (Hubert Tardieu, Arnold Rochfeld et René Colletti) qui permit en 1983 la publication du livre vert, livre de référence sur MERISE, depuis près de dix ans et déjà diffusé à près de 30000 exemplaires. Vers la même époque, Bernard Cohen créait CECIMA, société de conseil qui allait bientôt se spécialiser dans MERISE avec le renfort de Dominique Nanci et d'Henri Heckenroth, qui bientôt plus de quinze ans enseignent, pratiquent et diffusent cette méthode.

Ainsi se forgea une riche expérience dont ce traité d'ingénierie ne révèle bien sûr que la face "écrite". Nous faisons le voeu que ses lecteurs s'efforcent de le travailler comme il fut écrit et vécu, avec l'intelligence de la "médiation" par laquelle une organisation sociale apprend à s'observer dans sa quête passionnée des quelques projets qui l'inspirent. "Le chemin se construit en marchant" : cette belle phrase de Machado qu'aime rappeler E. Morin pour introduire "La Méthode" , définit aussi "l'ingénierie des systèmes d'information avec Merise" .

La Méthode n'est pas donnée, elle se construit en marchant. En accompagnant les auteurs de ce manuel qui ont prouvé, en marchant, qu'ils savaient "construire le chemin", le lecteur apprend non pas la méthode, mais l'intelligence rusée par laquelle se construit le chemin. Ainsi ces sentiers de grande randonnée que des marcheurs expérimentés ont soigneusement balisé : le promeneur peut à tout moment adapter sans réserve sa propre progression au gré de sa forme et de ses humeurs. Il sait qu'il existe un solide système de repérage qui l'assurera de la légitimité de ses éventuelles escapades hors de l'itinéraire balisé. Ce livre sera, pour bien des ingénieurs en systèmes d'information encore inexperts, l'équivalent de la carte du randonneur qui révèle les balises jalonnant sa marche ; il n'est pas le premier ni le dernier, mais il est sûrement celui qui bénéficie le plus du retour d'expérience car écrit plus de dix ans après la conception de Merise.

Il présente aussi un autre avantage pédagogique. A l'heure où les méthodes de conception - et les méthodes MERISE en particulier - connaissent une phase normale de régénération, un premier changement de génération dans la lignée d'un même génotype, il était utile de "recaler" soigneusement la base de départ, tant dans les prémisses théoriques et conceptuels que dans les définitions validées pragmatiquement de ses procédures. Dans la prolifération des méthodes se référant à MERISE, le débutant peut légitimement s'inquiéter. Ce manuel très soigneusement travaillé par une équipe riche de quinze ans d'expérience constitue sans doute à ce jour le dossier publié le plus soigneusement étalonné dont nous disposons. A partir de ce repère bien défini, les progressions vers les deuxièmes générations qui ont vu le jour depuis peu pourront être repérées par rapport à une base de départ heureusement "revisitée".

Qu'on nous permette enfin de souligner le caractère exemplaire de la

"production" de ce livre : à l'heure où chacun se plaint des difficultés de la coopération Université-Entreprise, voila une démonstration de sa faisabilité, et nous semble-t-il, de son efficacité. Depuis la rencontre des deux préfaciers (en 1974), puis, dans leur proche environnement, des auteurs de ce livre, nous plaidons pour la nécessité, l'intérêt, l'enrichissement..... et la praticabilité de cette coopération. Cet ouvrage sur "L'ingénierie des systèmes d'information" écrit par des consultants confirmés, Bernard Cohen et Dominique Nanci de CECIMA, Henri Heckenroth, et par des universitaires, Bernard Espinasse, Professeur à l'Université Aix-Marseille et Jean Claude Asselborn Professeur au Centre Universitaire de Luxembourg, ne témoigne-t-il pas par là de la richesse de cette coopération qui s'étend aujourd'hui au niveau européen... Puissent-ils être encouragés dans cette voie, et bien d'autres avec eux, par l'audience que connaîtra leur manuel. Les organisations sociales contemporaines n'ont-elles pas besoin de tels témoignages?

Jean Louis Le Moigne et Hubert Tardieu (1992)

Introduction

La méthode Merise, méthode de conception et de développement de systèmes d'information informatisés, a maintenant plus de vingt ans. La diffusion de cette méthode est un succès, tant en France où elle fait office de standard que dans une bonne partie de l'Europe, où elle est adoptée, parfois avec quelques aménagements mineurs. Indiscutablement, la méthode Merise constitue une étape importante en ingénierie des systèmes d'information.

A l'heure où l'information n'est plus seulement considérée comme une ressource opérationnelle mais aussi comme une ressource stratégique pour l'entreprise, son système d'information devient un facteur de différenciation par rapport à ses concurrents. C'est par sa culture et son système d'information performant que l'entreprise pourra s'adapter à son environnement concurrentiel. C'est dire toute l'importance des méthodes de conception et de développement de systèmes d'information mises en oeuvre dans l'entreprise.

De l'ingénierie des systèmes d'information

Concevoir un système d'information, réaliser les applications informatiques supportant ce système d'information, constituent un ensemble, une suite de tâches complexes. A l'instar d'un produit industriel, un système d'information et son logiciel associé s'inscrivent dans la durée. D'une approche artisanale s'appuyant essentiellement sur un savoir faire individuel, l'informatique a progressivement été confrontée aux contraintes d'industrialisation afin d'assurer la qualité et la pérennité de ses productions logicielles.

Pour assurer ce développement de systèmes d'information, différents "métiers" (utilisateurs, experts, organisateurs, informaticiens, ...) interviennent ensemble dans un processus de production, constitué de différentes activités exercées dans un environnement organisationnel et technique. Coordonner, ordonnancer ces différentes activités dans un souci de professionnalisme (c'est-à-dire être capable de maîtriser et reproduire le processus de production) relève de l'ingénierie informatique.

Aujourd'hui, sous le terme d'ingénierie informatique, on réunit en général l'ingénierie de systèmes d'information et le génie logiciel (figure 0.1).

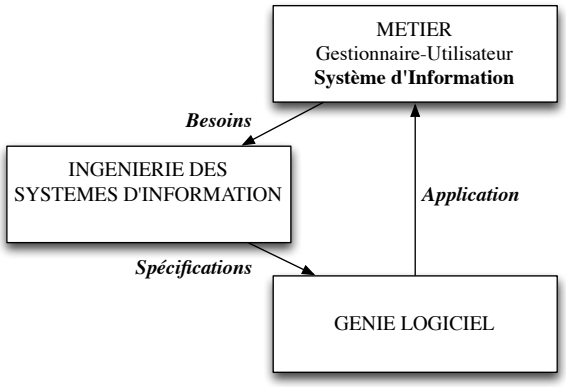


Figure 0.1 : Les deux composantes de l'ingénierie informatique

Le terme **Génie Logiciel** (Software Engineering) est né en Europe à la fin des années 60 (en 1968 à Garmisch - Partenkirchen) sous le patronage de l'OTAN. Le génie logiciel regroupe l'ensemble des méthodes, techniques et outils de développement de logiciel. Le génie logiciel est d'abord une discipline centrée sur la maîtrise de la technique informatique, et concerne essentiellement un public d'informaticiens.

Le terme d'**Ingénierie de Systèmes d'Information** (Requirement Engineering ou User Engineering) est bien plus récent, il n'est apparu qu'au début de la décennie 90. L'ingénierie de systèmes d'information vise à transformer les besoins et attentes des utilisateurs en spécifications formalisées d'une future application informatique. Transdisciplinaire par nécessité (métier destinataire, organisation, informatique), l'ingénierie de systèmes d'information s'est progressivement constituée un ensemble de méthodes et techniques qui en font aujourd'hui une discipline spécifique.

L'ensemble des activités de l'ingénierie informatique peut être présenté selon le schéma illustré dans la figure 0.2.

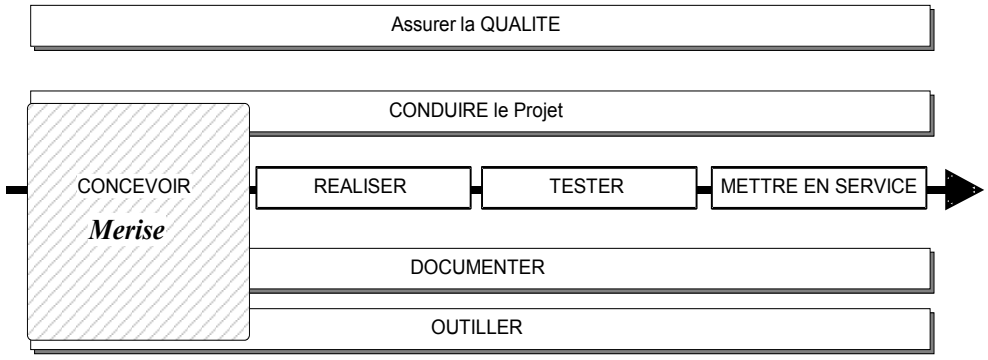


Figure 0.2 : Panorama général des activités de l'ingénierie informatique

Autour de chacune de ces activités, des savoir faire se sont progressivement élaborés jusqu'à constituer des méthodes et des techniques, supports de l'ingénierie informatique. Ces méthodes et techniques sont d'une part issues des efforts de recherche et d'innovation développés au sein de la communauté universitaire et professionnelle, d'autre part le fruit de l'expérience acquise sur le terrain. **La contribution de la méthode Merise se situe principalement en support à une activité de conception.**

De la notion de méthode

L'élaboration et l'usage de méthodes pour la réalisation d'objets artificiels (conçus et réalisés par l'homme) se retrouve dans de nombreux domaines tels que le génie civil, le génie chimique, le génie mécanique, la gestion et l'informatique. Dans la mise en oeuvre de techniques, face aux tâtonnements d'une démarche intuitive, tirant les enseignements des succès et échecs antérieurs, les concepteurs ont progressivement synthétisé leurs expériences. Sous certaines conditions, ce savoir-faire devient une méthode.

Mais qu'est-ce qu'une méthode ? Plusieurs définitions peuvent en être données comme l'indique le petit Robert. Une méthode peut être "*une marche, un ensemble de démarches que suit l'esprit pour découvrir et démontrer la vérité (dans les sciences)*", ce peut être aussi "*un ordre suivi pour exécuter quelque ouvrage de l'esprit et l'arrangement qui en résulte*". C'est encore "*un ensemble de démarches raisonnées, suivies pour parvenir à un but*". On peut définir une méthode comme "*un ensemble des règles, des principes normatifs sur lesquels reposent l'enseignement, la pratique d'un art*". Enfin ce peut être "*le livre, l'ouvrage exposant de façon graduelle ces règles, ces principes*".

La nature spécifique du système d'information, à la fois objet "naturel" et objet "artificiel" comme nous le verrons plus loin, nous conduit à définir une méthode pour leur conception comme avant tout un schéma de réflexion fournissant au concepteur un guide continu indiquant la manière d'aborder les problèmes. C'est ce que nous appelons les principes généraux de la méthode.

La façon de mettre en oeuvre ces principes généraux au cours du processus de conception se concrétise par une démarche, proposant une succession progressive d'étapes. Cette démarche est fréquemment appuyée sur des raisonnements qui permettent de mettre en oeuvre plus aisément la démarche. Ces raisonnements sont soit l'émanation directe des principes généraux et constituent ainsi l'originalité de la démarche, soit plus généraux et adaptés aux principes de la méthode.

Pour conduire ces raisonnements, et pour assurer la communication entre les intervenants dans le processus de conception, la méthode doit proposer des modèles : "*Nous ne raisonnons que sur des modèles*" (P.Valery) et "*Nous ne communiquons que par des modèles*" (G.Bateson). Les modèles seront exprimés et validés en utilisant des formalismes, langages permettant de désigner et décrire tous les concepts nécessaires à la spécification des systèmes étudiés. Nous retrouverons dans la méthode Merise, l'apport fondamental de la Sciences des

Systèmes dont l'objet est "*l'élaboration et la mise en oeuvre des méthodes de modélisation des phénomènes perçus ou conçus complexes*" (J-L. Le Moigne).

Une méthode peut proposer des normes, ou standards de présentation des différents résultats de la conception, facilitant la communication et le contrôle du respect de la démarche. Il nous semble fondamental de ne pas confondre méthode et normes. Chaque entreprise pratiquant une méthode donnée peut définir ses propres normes, adaptées à son contexte d'utilisation, tout en respectant l'intégralité de la méthode. Inversement, trop d'équipes de concepteurs ont encore tendance à avoir l'impression de s'être dotés d'une méthode parce qu'ils se mettent à utiliser certains formulaires ou à dessiner des "modèles" en référence à quelques normes, tout en continuant par ailleurs à faire comme avant, c'est à dire sans avoir adopté des raisonnements et une démarche méthodologiques.

La mise en application d'une méthode nécessite la mobilisation de moyens, principalement une structure de groupes avec les définitions des fonctions de chacun. Comme dans de nombreux domaines d'ingénierie, les processus de conception peuvent être assistés par ordinateur ; aussi les concepteurs pourront être assistés éventuellement d'outils logiciels adaptés, facilitant la conception et la documentation. En ingénierie des systèmes d'information, de nombreux Ateliers de Génie Logiciels (AGL) sont actuellement proposés. Il importe de ne pas confondre raisonnements, normes d'une méthode et AGL. Un AGL est un ensemble d'outils logiciels permettant de garantir l'usage de certaines normes, de remplir des fonctions correspondantes à l'automatisation de certaines tâches (C.A.O., aide au dessin, dictionnaire, générateurs de programmes, ...). Ces outils logiciels peuvent être spécifiques à une méthode, communs à un ensemble de méthodes ou d'un usage totalement indépendant de la méthode pratiquée.

Ainsi une méthode est, à notre sens :

- une démarche,
- reflétant des principes généraux définis,
- proposant des raisonnements spécifiques et généraux pour manipuler des concepts aptes à donner une représentation fidèle des systèmes étudiés,
- permettant une utilisation efficace grâce à une structure d'équipe, une répartition des rôles et des outils logiciels adaptés.

De la méthode Merise

Merise est une méthode française de conception de systèmes d'information. Elaborée à partir de 1978 sous l'égide du Ministère de l'Industrie, sur la base de travaux de recherche et avec la collaboration de grandes sociétés de service françaises, son utilisation s'est progressivement étendue dans les services

informatiques des entreprises et des administrations où elle est largement recommandée.

Aujourd'hui encore, la méthode Merise est la méthode de conception de systèmes d'information la plus largement pratiquée en France. Au début des années 90, des enquêtes ont révélé que 50 % des services informatiques déclarent utiliser une méthode de conception, et que parmi eux, 75 % ont choisi la méthode MERISE. Cette réussite se concrétise par la présence sur le marché d'outils de conception dédiés à la méthode MERISE.

Depuis son introduction, la méthode Merise s'est confrontée aux réalités de la mise en oeuvre dans une grande variété d'organismes. En une vingtaine d'années, elle a connu des développements et des enrichissements profitables. Elle a pris en compte les évolutions de l'informatique et continue de s'adapter aux nouvelles technologies: architectures client-serveur, interfaces graphiques, démarche de développement rapide, approche objet, applications ouvertes intra /internet.

Il n'y a pas de méthode unique, car il n'y a pas de réponse unique à des contextes variés. Les méthodes n'ont certes pas inventé ou découvert ce qui constitue les activités de conception et de développement des projets informatiques, ni la succession des étapes à franchir. Mais tous les "méthodologues" s'accordent sur l'importance de définir précisément, dans une méthode, ces activités et ces étapes ainsi que sur l'importance de l'unicité de méthode pour un organisme.

Il n'y a pas de méthode éternelle car une méthode correspond à un savoir-faire dépendant d'un environnement culturel et technique. Aujourd'hui, la méthode Merise, avec ses évolutions progressives correspond encore globalement aux savoir-faire actuels en ingénierie de systèmes d'information de gestion..

Sur cet ouvrage

Depuis 1983, date à laquelle a été publié le premier livre intitulé "La méthode Merise: principes et outils" de H.Tardieu, A.Rochfeld R.Colletti [Tardieu Rochfeld Colletti 83], de nombreux ouvrages traitant de cette méthode ont vu le jour. Certains de ces ouvrages n'offrent qu'une présentation simplifiée de la méthode, se limitant la plupart du temps à un exposé des modèles proposés par la méthode et occultant la démarche et les moyens, ainsi que les fondements mêmes de la méthode.

D'autres ouvrages ou articles, au contraire enrichissent avec profit Merise, soit ses fondements théoriques, c'est la cas par exemple de l'ouvrage de Y. Tabourier [Tabourier 86], soit sa démarche avec ceux de A. Rochfeld et J. Moréjon [Rochfeld, Moréjon 89] et de G. Panet, R. Letouche, C. Peugeot [Panet &al. 91], soit ses formalismes et ses modèles de données avec l'ouvrage de J.P. Matheron [Matheron 91], soit enfin abordent son évolution vers l'objet, citons notamment les ouvrages de J. Moréjon [Morejon 94], sur Merise 2 de G.

Panet et R. Letouche [Panet et Letouche 94] et enfin le récent ouvrage de M. Bouzeghoub et A. Rochfeld, [Bouzeghoub et Rochfeld 00].

Plus de vingt années de pratique dans des projets opérationnels en conception de systèmes d'information et autant d'années en formation en ce domaine ont permis d'affiner cet ouvrage. Il présente de façon originale et pertinente la méthode Merise selon trois grandes composantes principales : tout d'abord les modèles et raisonnements proposés par la méthode, ensuite la démarche préconisée et enfin l'organisation des moyens associés. Toutes ces années d'expérience confirment que la maîtrise de ces trois composantes s'avère indispensable à l'efficacité de la mise en oeuvre de la méthode Merise.

L'ouvrage s'articule autour de quatre grandes parties :

- La partie I présente les principes généraux et les fondements théoriques de la méthode Merise. Nous précisons aussi à quels types de systèmes d'information la méthode Merise nous semble adaptée.
- La partie II traite des raisonnements mis en oeuvre par le concepteur pour l'élaboration des modèles pour la compréhension et la conception du système d'information organisationnel (SIO).
- La partie III traite des raisonnements mis en oeuvre par le concepteur pour l'élaboration des modèles pour la compréhension et la conception du système d'information informatisé (SII).
- La partie IV expose les différentes mises en œuvre possibles de la méthode. Tout d'abord sont présentées en détail deux démarches l'une dite "classique" et l'autre "rapide" qui peuvent être adoptées. Ensuite est abordée l'organisation des intervenants impliqués dans cette mise en œuvre ainsi que les outils logiciels qui peuvent être utilisés en support. Enfin l'usage de Merise pour la réingénierie d'organisation ou Business Process Reengineering (BPR) est abordé.

L'ouvrage se termine par une conclusion générale présentant un bilan et un positionnement de la méthode Merise, notamment par rapport à l'approche Objet et UML, les objets métiers et la réutilisation.

Dans cet ouvrage, écrit par des hommes de terrain et de recherche, nous avons voulu réunir l'expérience de nombreuses années de mise en oeuvre opérationnelle et les résultats des réflexions et recherches continuelles et récentes suscitées par la pratique. Nous avons essayé de présenter la méthode Merise de façon précise et complète mais également claire et pratique. Nous avons voulu proposer une référence réactualisée pour la deuxième décennie de cette méthode, face à la poussée d'autre méthodes dont l'efficacité et la facilité de mise en œuvre en ingénierie de systèmes d'information n'est pas encore démontrée sur le terrain..

Enfin cet ouvrage s'adresse aussi bien à des professionnels, concepteurs de

systèmes d'information qu'à des étudiants des cycles universitaires de filière professionnelles spécialisées des Universités, ainsi qu'aux étudiants des Grandes Ecoles d'Ingénieur ou de Commerce désirant mieux maîtriser cette méthode de conception de systèmes d'information.

Première Partie

Principes généraux et fondements théoriques de la méthode Merise

Partie 1
Principes généraux et fondements théoriques de la méthode Merise

1

Origines et évolution de la méthode MERISE

Les techniques et leurs conditions de mise en oeuvre évoluant, les méthodes de mise en oeuvre de ces techniques à leur tour évoluent. Ainsi, la méthode Merise a trouvé ses origines dans l'évolution d'une part de l'informatisation des entreprises et d'autre part de celle des méthodes dites d'analyse en informatique de gestion. Depuis sa naissance, en 1978, la méthode Merise a également connu des évolutions.

Origines de la méthode Merise

Jusqu'aux années 70, l'informatisation des entreprises s'est attachée à l'automatisation des processus administratifs (facturation, paye, suivi des stocks, ..) avec une technologie encore coûteuse. Ce type d'informatisation privilégiait les traitements par rapport au partage des informations. Les méthodes de mise en oeuvre de cette génération étaient destinées à concevoir des "chaînes de traitements" avec l'approche suivante : à partir des résultats à produire, définir les traitements à effectuer, puis en déduire les données nécessaires pour alimenter les traitements. La structure des données mémorisées sur les fichiers était contingente aux traitements à réaliser; d'où une multiplication des fichiers (temporaires ou permanents) générant une redondance importante des informations mémorisées. Nous retiendrons de ces premières méthodes, les deux plus marquantes, MINOS et CORIG, dont des propositions ont été reprises dans Merise.

Début des années 70, c'est l'apparition des systèmes transactionnels, de la multiprogrammation, des écrans claviers, des disques de grande capacité à coût abordable, mais aussi la concurrence stimulante de la mini-informatique et le développement des premiers systèmes de gestion de bases de données. Tous ces éléments ont conduit les informaticiens à reconcevoir des applications intégrées en essayant de tirer parti au mieux de cette évolution. C'est l'époque des "transpositions" ou "reconversions" de systèmes importants.

Le contexte de crise économique de cette décennie a rendu indispensable le

développement des méthodes de management qui désormais introduisent fortement l'usage de l'informatique à travers des tableaux de bord, interrogations aléatoires, statistiques. Cette période est celle de la prise de conscience de la difficulté de concevoir des systèmes qui intègrent l'ensemble de l'activité de l'entreprise, en conservant une facilité d'évolution. Il s'agit en particulier :

- du manque de cohérence globale entre les informations des différentes applications,
- de la lourdeur de la mise en oeuvre informatique (de la conception à la réalisation).

La fin des années 70 se caractérise, au niveau de la technologie, par plus de puissance et de capacité à des coûts réduits, le développement des réseaux locaux et nationaux, l'éclosion du phénomène micro, la généralisation des systèmes de gestion de bases de données. C'est aussi l'apparition de langages permettant à l'utilisateur d'accéder plus facilement aux informations, ainsi que l'amélioration de la productivité de la réalisation avec l'utilisation d'outils (ateliers génie logiciel, langage 4ème génération, ...).

La prise de conscience révélée lors de la période précédente devient une obligation. Il s'agit de reconcevoir l'architecture générale de l'informatique au sein des entreprises en :

- assurant la cohérence générale des informations
- préservant l'évolutivité des modes de gestion et d'organisation
- permettant l'introduction des nouvelles technologies sans compromettre l'acquis
- associant, dans leurs responsabilités respectives, décideurs, utilisateurs et informaticiens.

C'est dans ce contexte qu'ont émergé :

- la notion de système d'information
- *la nécessité de méthode* complète de conception et de spécification permettant l'informatisation des systèmes d'information, prenant le pas sur les méthodes d'analyse.

L'évolution de la technologie informatique, les insuffisances des précédentes méthodes stimulent la recherche de nouvelles voies. Tout d'abord, au niveau des principes fondamentaux, l'approche par les besoins (ou traitements) est condamnée. Deux veines de réflexion contribuent au renouveau :

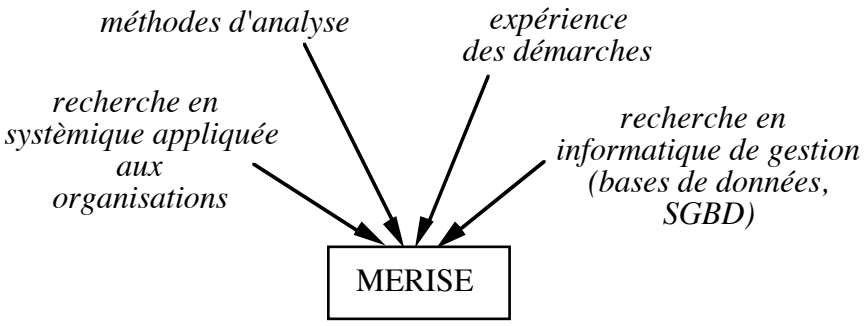
- *l'approche systémique* du système d'information, approche modélisatrice de l'organisation (l'entreprise) et de son système d'information, développée principalement en France par J.L. Le Moigne et J. Mélèse.
- *la modélisation des données*, proposant des formalismes et des outils pour

décrire les données indépendamment de leurs utilisations dans des traitements. Ces méthodes ont alors comme objectif principal la construction de bases de données.

Des promoteurs de ces deux réflexions se trouvent réunis à Aix en Provence, dans une équipe de recherche composée de chercheurs du Centre d'Etude Technique de l'Equipement (CETE) d'Aix-en-Provence (animée par H. Tardieu) et du groupe de recherche GRASCE (URA CNRS) de l'Université d'Aix-Marseille III (animée par J.L. Le Moigne), qui fondent de 1974 à 1978 les bases théoriques et pratiques d'une nouvelle méthode de conception de système d'information [Tardieu, Nanci, Heckenroth 75-78] [Tardieu, Pascot, Nanci, Heckenroth 79] [Tardieu, Nanci, Pascot 79] [Heckenroth, Tardieu, Espinasse 80]. Simultanément, on enregistre, sur le terrain, une accumulation d'expériences en termes de démarche.

En 1977, sous l'égide du Ministère de l'Industrie, soucieux de concevoir et définir une méthode d'intérêt national, avec la collaboration des principales sociétés de service françaises et du CETE d'Aix-en-Provence, un groupe de travail se constitue et entreprend une synthèse qui :

- réactualise les acquis sur la spécification des traitements, issus des méthodes antérieures ;
- intègre les nouvelles méthodes orientées système d'information et approche par les données;
- propose une démarche, fruit de l'expérience, qui garantisse la rigueur de la méthode et sa facilité d'application sur le terrain.



La méthode Merise naît officiellement [CTI 79], marquant ainsi l'apparition de méthodes complètes qui ambitionnent de répondre efficacement aux problèmes posés par la conception des systèmes d'information adaptés aux fonctionnements des entreprises et technologies informatiques des années 80. Les premiers ouvrages sur la méthode Merise paraissent en 83 et 85 [Tardieu, Rochfeld, Colletti 83][Tardieu et al. 85].

Evolutions de la méthode Merise

Depuis 1980, la méthode Merise s'est confrontée aux réalités d'une mise en oeuvre dans une grande variété d'organismes. Elle est largement diffusée en France, souvent pratiquée en Europe du Sud (parfois avec des adaptations mineures) et plus récemment dans certains pays d'Europe du Nord comme la Belgique, la Suisse et depuis peu l'Allemagne par laquelle un projet du programme européen Force a financé sa diffusion dans des landers du nord. Ainsi, on peut dire que Merise constitue un standard de fait en conception de système d'information.

En presque vingt ans, elle a connu des développements et des enrichissements profitables, dont les principaux sont les suivants :

- extension du formalisme entité-relation, avec notamment l'explicitation de types et sous-types, de contraintes d'intégrité, ...
- clarification de la modélisation des traitements à l'aide du formalisme issu des réseaux de Pétri, à différents niveaux de préoccupation,
- extension des niveaux d'abstraction et de modèles, avec l'émergence du modèle logique de traitements (MLT) et du modèle organisationnel de données (MOD), [Panet, Letouche, Peugeot 91] ainsi que les propositions très intéressantes autour de Merise 2 de G.Panet et R.Letouche [Panet et Letouche 94],
- couplage avec des méthodes de conduite de projet ,
- développement d'ateliers de génie logiciel (A.G.L.) de conception intégrant de façon plus ou moins complète la Méthode Merise (AMC, MEGA, SILVERRUN, WIN'DESIGN, ...),
- ouverture vers les autres méthodes de génie logiciel (Merise et Yourdon [Phan 85],...), de génie cognitif (Merise et KADS [Brunet 90],...), ...
- adaptation à d'autres types d'activités ; domaine de la productique (Merise et GRAI [DCN, Cecima, Grai-Productique SA, LaboGrai 91]), l'EDI (Merise et l'EDI [Bergman, Cucchi, Espinasse, Lagaert 91]), le BPR (Business Process Reengineering - cf partie VI de l'ouvrage) et d'environnements techniques (bases de données réparties, architectures client-serveur, monétique, cartes à puce,...).

Cette évolution de la méthode s'est faite principalement à l'initiative des Sociétés de Service qui l'avaient adoptée. La situation concurrentielle n'a pas toujours permis un développement convergent et collectif de la méthode (discrétion des recherches sérieuses, annonces d'"innovations" et "adaptations" trop souvent dictées par le marketing, dépréciations et dénigrements par absence de compétence, récupérations hâtives et fragmentaires, ...). Rappelons encore le rôle très important de la société savante concernée, l'AFCET,

aujourd'hui disparue et de plusieurs de ses groupes de travail, comme structure de rencontre, de reflexion et de diffusion des travaux de recherche élaborés par les différents promoteurs de la méthode Merise.

Ouvertures de la méthode Merise

Depuis le début des années 90, l'ingénierie des systèmes d'information a vu apparaître et se développer de nouvelles technologies, de nouveaux savoir-faire. Fidèle à ses principes d'origine (aptitude à la greffe...), la méthode Merise a résolument cherché à s'ouvrir à ces innovations.

De tous horizons, des contributions souvent trop silencieuses, ont su analyser de façon critique et au delà des modes, les nouvelles tendances, en extraire les idées réellement innovantes dans le domaine de la conception des systèmes d'information et en enrichir la méthode Merise. Cette ouverture s'est particulièrement manifestée vers :

- le développement rapide, associé à la diffusion d'outils de productivité, qui a remis en cause la démarche classique adoptée par Merise,
- le client-serveur, nouvelle technologie, qui a installé définitivement le micro-ordinateur comme poste de travail,
- le Business Process Reengineering (BPR) qui ravive l'intérêt sur les problèmes d'organisation et les systèmes d'information,
- l'approche objet qui, à partir des langages de programmation, ambitionne progressivement à couvrir l'ensemble du champ du génie informatique.

Ces diverses ouvertures seront traitées tout au long de cet ouvrage, tant dans les parties II et III sur les raisonnements que dans la partie IV sur la mise en œuvre de la méthode Merise.

2

Les principes fondamentaux de Merise

Ce chapitre est plus particulièrement destiné au lecteur souhaitant mieux comprendre les bases théoriques et découvrir les principes fondamentaux de la méthode Merise. Ainsi, le lecteur ayant déjà une pratique de la méthode devrait y trouver certaines justifications relatives à sa mise en œuvre opérationnelle.

Le lecteur découvrant la méthode Merise et souhaitant une approche plus pragmatique peut directement passer au chapitre 3, et revenir ultérieurement sur ce chapitre 2.

Contribution de la science des systèmes

La science des systèmesVoir systèmique, appelée aussi systémique, est une discipline scientifique autonome récente, puisqu'elle date de la fin des années 70. La systémique se définit plutôt par son projet que par son objet. Elle prend ses racines principalement dans la théorie des systèmes, la théorie de la commande, la théorie du contrôle, la cybernétique. La systémique a pour projet *" la modélisation des phénomènes perçus ou conçus complexes : modélisation, à fin d'anticipation, d'éventuelles interventions intentionnelles et de leurs conséquences enchevêtrées "* [Le Moigne 87]. La science des systèmes a ainsi pour finalité de proposer des modèles pour l'action ou la compréhension d'objets ou de phénomènes complexes, dans des domaines les plus variés (biologie, sciences sociales, gestion...).

Sa contribution aux sciences de l'organisation, au management et plus particulièrement à la définition du concept de système d'information est indéniable, et appellerait de passionnants mais longs développements ; nous invitons les lecteurs intéressés à se reporter aux ouvrages [Le Moigne 77 et 90] et [Mélèse 72 et 79].

Notre propos n'est pas de présenter en détail la science des systèmes (voir les ouvrages déjà cités), mais plutôt de développer quelques contributions de cette théorie qui nous semblent les plus pertinentes pour la conception des systèmes d'information et qui constituent, à nos yeux, le creuset de la méthode Merise.

Hypothèses du paradigme systémique

Le paradigme systémique repose sur les trois hypothèses fondamentales suivantes :

- *hypothèse téléologique* où l'objet à modéliser est supposé doté d'au moins un projet identifiable. Le fonctionnement et l'évolution de cet objet peuvent être interprétés par des projets qui eux-mêmes détermineront des structures possibles ;
- *hypothèse d'ouverture sur l'environnement* où l'objet à modéliser est ouvert sur l'environnement que l'on doit présenter, même s'il n'est pas descriptible de façon exhaustive ;
- *hypothèse structuraliste* où l'objet à modéliser doit être décrit dans sa totalité, fonctionnant et évoluant.

Ces hypothèses sont schématisées dans la figure 2.1 [Le Moigne 77].

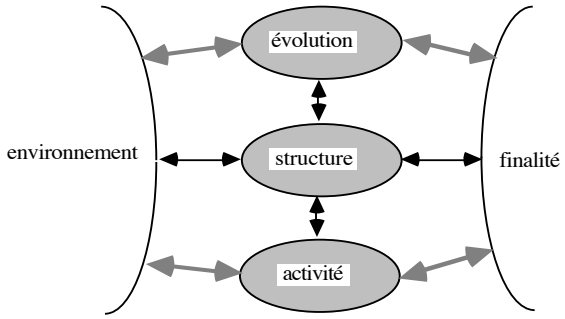


Figure 2.1: le paradigme systémique.

Appliqué à l'entreprise, ce paradigme met l'accent sur les inter-relations entre sa structure, son activité, son évolution et ses finalités, cela dans un environnement changeant.

Référentiel des complexités croissantes

K.E. Boulding [Boulding 56] puis J.L. Le Moigne [Le Moigne 77] ont proposé une classification des systèmes fondée sur leur complexité croissante. Cette classification se décompose en 9 niveaux. Ces niveaux sont bien sûr artificiels, et permettent à un observateur de situer sa modélisation, en fonction de la complexité de l'objet réel et surtout en fonction des objectifs du modèle représentant cet objet.

Cette classification en complexité croissante nous permet d'illustrer l'émergence de la notion de système d'information, de nous doter d'une définition du système d'information et d'appréhender les limites de l'informatisation de ces systèmes.

- *Niveau 1 :* l'objet est passif et sans activité (une pierre, une table).
- *Niveau 2 :* l'objet est actif et transforme des flux (structure active, mouvement prédéterminé : pressoir, ampoule électrique, la plupart des machines).
- *Niveau 3 :* l'objet actif est régulé. Il présente quelques régularités de comportement, et l'on observe des refus d'autres comportements possibles. On peut le modéliser comme doté d'un autre processeur chargé d'assumer cette régulation (la cocotte-minute, le régulateur à boules de Watt).

La figure 2.2 illustre ces niveaux 2 et 3 de complexité.

Figure 2.2 : Les niveaux 2 et 3 de complexité.

- *Niveau 4 :* l'objet s'informe. Au lieu d'un couplage physique entre les deux processeurs, le processeur de régulation s'informe sur l'activité du processeur actif (injection électronique, freinage ABS...). A ce niveau, le système, pour se réguler par rapport à quelque finalité intelligible, capte de l'information - représentation sur son comportement. C'est aussi l'apparition du schéma classique de la théorie de l'information [Weaver &Shannon 49] :

  émetteur – codage _ transmission _ décodage – **récepteur**

  Cette modélisation de l'objet régulé grâce à un flux d'informations constitue le schéma de base de la cybernétique.
- *Niveau 5 :* l'objet décide de son activité. Il passe d'un comportement théoriquement prévisible (ou programmé) à un comportement " libre " et apparaît doté d'un projet. C'est l'émergence d'un processeur décisionnel autonome.

La figure 2.3 illustre ces niveaux 4 et 5.

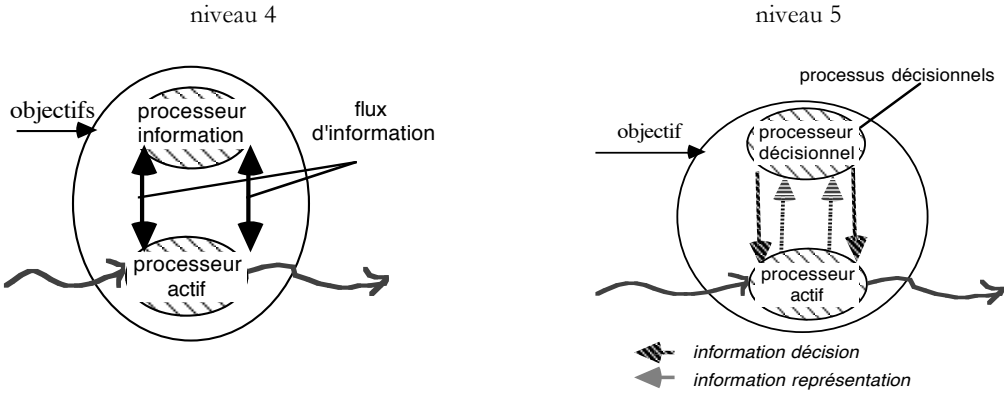
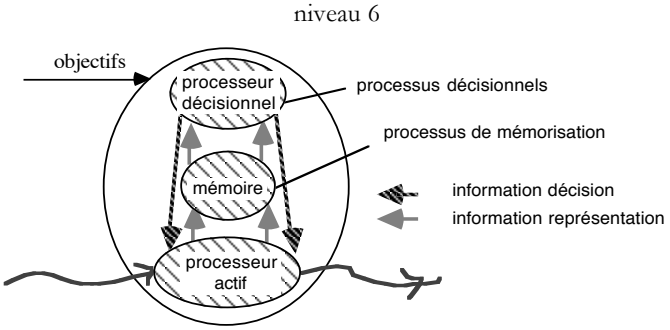


Figure 2.3 : Les niveaux 4 et 5 de complexité.

- *Niveau 6 :* l'objet a une mémoire. Pour prendre ses décisions, le processeur décisionnel fait appel à des informations-représentations non seulement du comportement actuel, mais aussi des comportements passés. Il s'appuie donc sur une chronique des informations-représentations passées. Ce niveau est ainsi caractérisé par l'émergence de processus de mémorisation. L'objet est alors doté d'une mémoire.
- *Niveau 7 :* l'objet se coordonne. Le passage du niveau 6 au niveau 7 est la traduction d'un gain de complexité dans la modélisation de l'objet. Le processeur actif devient une fédération de processeurs actifs, nécessitant une coordination ; il est alors appelé système opérant (SO). La même évolution s'applique au processeur décisionnel qui devient système de pilotage (SP), ainsi qu'à la mémoire qui devient système d'information (SI).

La figure 2.4 illustre ces niveaux 6 et 7 de complexité.



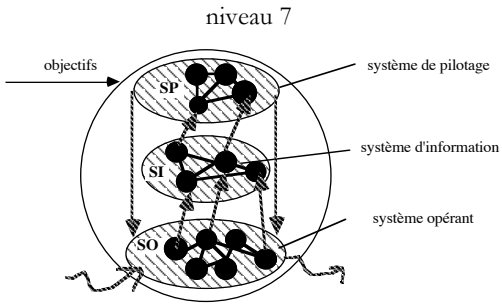
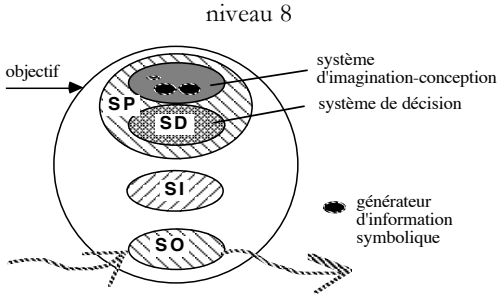


Figure 2.4 : Les niveaux 6 et 7 de complexité.

Les niveaux ultérieurs concernent des différenciations successives du système de pilotage.

- *Niveau 8 :* l'objet imagine et s'auto-organise. Il est capable, pour atteindre ses objectifs, d'imaginer l'organisation de ses sous-systèmes la mieux adaptée. Le système de pilotage dispose de capacités d'imagination, de conception qu'il mettra à profit pour élaborer des plans d'actions et conduire l'adaptation des autres sous-systèmes.
- *Niveau 9 :* l'objet s'autofinalise. Ultime stade de l'évolution, l'objet est désormais capable de définir son projet, ses objectifs. C'est l'émergence de la conscience. Le système de pilotage dispose de capacités de finalisation (système de finalisation), lui permettant de changer ses objectifs. Pour les atteindre, il fera évoluer en conséquence ses sous-systèmes opérants, de pilotage et d'information.

La figure 2.5 illustre ces deux derniers niveaux. Pour ne pas surcharger la figure, les interrelations entre les sous-systèmes ne sont pas représentées.



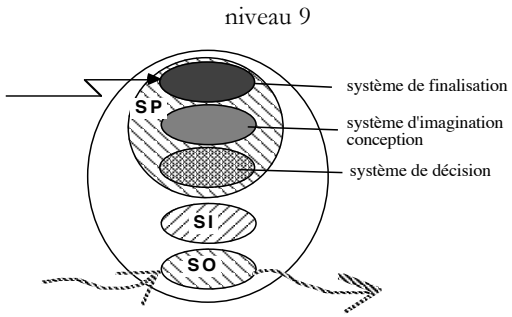


Figure 2.5 : Les niveaux 8 et 9 de complexité.

Modélisation systémique de l'entreprise

En nous proposant une modélisation progressive des objets, la systémique facilite la compréhension de l'entreprise, objet complexe actif et organisé. Tout corps social organisé, en particulier les entreprises ou les administrations, pourra être modélisé comme un système dont la complexité se situera au niveau 9 précédemment défini.

Au travers de l'étude des niveaux de complexité, nous avons vu progressivement apparaître le rôle du sous-système d'information (émergence de processus de mémorisation). Nous pourrons ainsi mieux situer les fonctions et le rôle du système d'information de l'entreprise et ses relations avec les autres sous-systèmes, le sous-système opérant et le sous-système de pilotage.

La figure 2.6 illustre la modélisation systémique de l'entreprise (ou organisation), avec ses trois sous-systèmes : système opérant (SO), Système d'information (SI) et système de pilotage (SP).

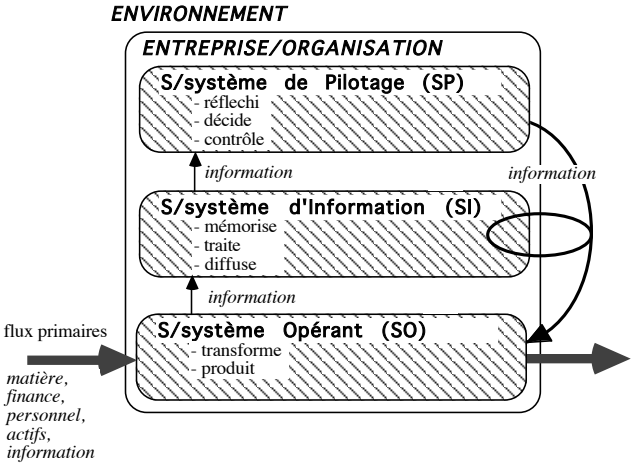


Figure 2.6 : Les trois sous-systèmes de l'entreprise.

Décrivons de façon succincte le rôle de chacun de ces systèmes composant l'entreprise-système.

- Le *système opérant* est le siège de l'activité productive de l'entreprise. Cette activité consiste en une transformation de ressources ou flux primaires. Ces flux primaires peuvent être des flux de matière, des flux financiers, des flux de personnel, des flux d'actifs ou enfin des flux d'information.

Notons au passage que ce dernier type de flux concernera des "informations-matière première", qui ne seront pas directement utilisées par l'entreprise à des fins de pilotage. Très souvent, dans le secteur tertiaire, la transformation de flux d'information de ce type est la principale activité de l'entreprise. Cela revient à assimiler le système opérant à un système transformant de l'information, on parlera alors de système d'information de production.

- Le *système de pilotage* est le siège de l'activité décisionnelle de l'entreprise. Cette activité décisionnelle est très large et est assurée par tous les acteurs de l'entreprise, à des niveaux divers, depuis les acteurs agissant plutôt dans l'activité productrice de l'entreprise, à ceux dirigeant cette dernière. Elle permet la régulation, le pilotage mais aussi l'adaptation de l'entreprise à son environnement. C'est cette activité qui conduira l'évolution, décidera notamment de l'organisation et de l'évolution des systèmes opérants et d'information.

Dans un contexte classique de gestion, cette activité concerne entre autres l'allocation des ressources impliquées (prévision, planification...), ainsi que leur suivi (contrôle de gestion, contrôle budgétaire...). Pour un affinement des fonctions de ce système de pilotage, on pourra s'inspirer notamment des travaux de J.Mélèse [Mélèse 82].

- Le *système d'information* que nous considérons, pour l'instant, comme un système de mémorisation dont le rôle est de permettre au système de pilotage d'assurer ses fonctions, notamment en assurant son couplage avec le système opérant. Le paragraphe suivant traite en détail des fonctions de ce système.

Les fonctions du système d'information dans l'entreprise

L'analyse systémique nous a permis de faire émerger la notion de système d'information comme une *représentation de l'activité du système opérant et/ou du système de pilotage*, et de ses échanges avec l'environnement, *conçue à l'initiative du système de pilotage* en fonction des objectifs à atteindre et de l'organisation choisie.

Ce système d'information est destiné :

- au système de pilotage pour pouvoir connaître et maîtriser le fonctionnement du système opérant,
- au système opérant lorsque les flux transformés sont de nature

" information ".

Le système d'information (SI) assure dans l'entreprise, vue en tant que système, les fonctions primaires présentées dans la figure 2.7 [Le Moigne 78].

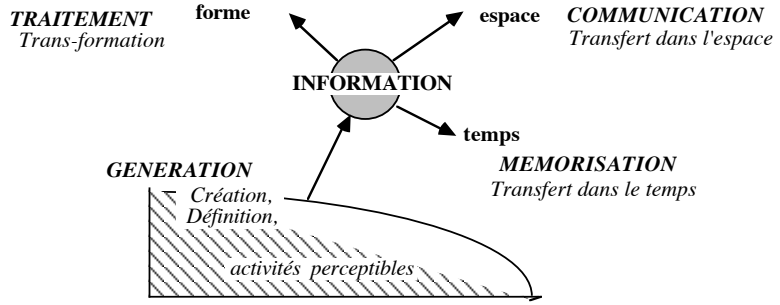


Figure 2.7 : Les fonctions primaires du système d'information.

- *La génération des informations.*

  Comme nous venons de le voir, le système d'information est *conçu à l'initiative du système de pilotage.* Le système de pilotage doit alors faire preuve d'imagination (niveau 8 de Boulding) dans la définition de l'information nécessaire à l'émergence du système d'information et de ses fonctions.

  La génération de l'information est ainsi une fonction indispensable que le système de pilotage doit exercer pour permettre la conception du système d'information. Cette génération est un préalable *nécessaire à toute mémorisation de l'information,* elle permettra toute saisie future de l'information et elle est *propre à chaque organisation.*

  Cette génération de l'information consiste à donner à toute information un nom et une définition, reconnus et partagés au sein de l'entreprise, cela revient à définir en fait le vocabulaire spécifique de l'entreprise. Elle concerne également la définition des événements déclarés " d'intérêt pour l'organisation ", conduisant aussi à définir les réactions que l'organisation devra développer en réponse à ces événements.

- La mémorisation des informations (transfert des informations dans le temps).

  La fonction de *mémorisation* (collective) des informations a un rôle central comme nous l'avons déjà vu dans le référentiel des complexités croissantes de K.Boulding : sans mémoire, pas d'apprentissage, pas d'intelligence. La nature et la signification des informations à mémoriser seront des éléments essentiels de la conception d'un système d'information.

- La *communication* et la *diffusion* des informations (transfert des informations dans l'espace).

  Le système d'information assure les échanges (acquisition et restitution)

d'informations avec le système opérant et le système de pilotage. L'organisation de l'acquisition et de la restitution des informations constituera un autre élément important de la conception.

- *L'exécution de traitements* (transfert des informations dans la forme).

  Par référence à l'approche système, les traitements sont soit des activités de transformation d'information-matière première (relevant donc du système opérant), soit des activités de décision, élémentaires ou complexes (relevant du système de pilotage). Le système d'information accueille, pour le compte du système de pilotage ou du système opérant, les traitements suffisamment formalisés et répétitifs. Ce qui était une décision-réflexion au niveau du système de pilotage devient un réflexe au niveau du système d'information.

Ces fonctions primaires du système d'information peuvent être représentées comme dans la figure 2.8.

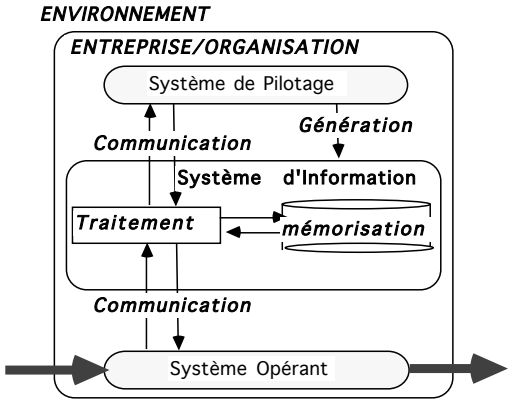


Figure 2.8 : Les fonctions du systèmes d'information dans l'entreprise.

Ainsi défini, le système d'information ne fait aucune hypothèse sur les moyens le supportant ; le système d'information existe indépendamment (et bien avant l'apparition) des techniques informatiques. Toutefois, ces techniques informatiques vont permettre d'amplifier les fonctions de mémorisation, de communication et de traitement des informations.

L'informatisation du système d'information comporte ainsi deux préoccupations majeures ; d'une part la compréhension et l'explicitation du système d'information (activité, information, organisation), et d'autre part la construction de logiciels (fichiers, programmes), support du système d'information.

Informatisation d'un système d'information

Système d'information organisationnel et système d'information informatisé

Dans ce qui précède, nous avons vu que les fonctions de mémorisation, communication et traitement du système d'information pouvaient être amplifiées par les techniques informatiques.

L'informatisation du système d'information conduit à distinguer dans la conception d'un système d'information deux niveaux d'étude différents (voir la figure 2.9).

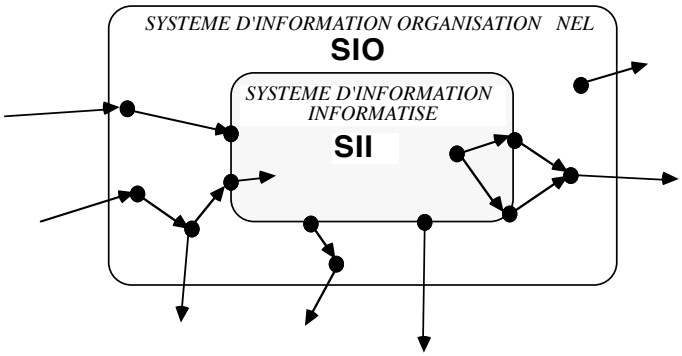


Figure 2.9 : Système d'Information Organisationnel et Informatisé.

- le niveau du *système d'information organisationnel* (SIO) qui exprime l'activité organisée associée au fonctionnement du système d'information (signification des informations, tâches humaines/informatisées),
- le niveau du *système d'information informatisé* (SII) qui ne concerne que le contenu informatisé (logiciel, fichiers ou bases).

Cette distinction se traduira par des préoccupations et des raisonnements différents lors de la conception du système d'information. Le système d'information organisationnel est essentiellement tourné vers les utilisateurs et fera appel à des disciplines des sciences de la gestion. Le système d'information informatisé est plus l'affaire des informaticiens et fera appel aux disciplines du génie logiciel. Cependant, le système d'information informatisé doit s'inscrire dans le système d'information organisationnel ; la conception du système d'information informatisé s'appuie donc sur celle du système d'information organisationnel.

Pour tenter d'illustrer l'imbrication et la différence de nature de ces systèmes d'information, on peut considérer le SIO comme "naturel et vivant" et le SII comme "artificiel". Concevoir un nouveau système d'information, c'est amplifier certaines fonctions du système naturel par une " prothèse artificielle". Pour ce faire, il faut d'abord comprendre le fonctionnement du système naturel, choisir les fonctions à " appareiller ", imaginer le fonctionnement du futur système mixte (naturel + artificiel), concevoir, spécifier et réaliser le système artificiel, puis remplacer le système actuel par le futur système.

Cette image rappelle que la conception du système artificiel ne peut se faire en dehors du système naturel et que le système naturel verra son comportement

modifié par la greffe du système artificiel.

Ces rétroactions mutuelles entre le SIO et le SII lors du processus de conception sont une particularité de l'ingénierie des systèmes d'information par rapport à d'autres ingénieries. Ainsi, dans l'ingénierie informatique de systèmes plus physiques (robotique, process, logiciel de base...), le milieu recevant l'application informatique ne réagit pas sur sa greffe. Par contre, dans l'ingénierie de systèmes d'information, le système à informatiser, par sa capacité de réaction due à son niveau de complexité (voir niveaux 8 et 9 du paragraphe " Référentiel des complexités croissantes " au début de ce chapitre) et à son caractère vivant, va réagir à l'implantation de l'application informatique en l'intégrant, la détournant ou la rejetant. Imaginer, comprendre, anticiper ces réactions sont des réflexions déterminantes dans l'informatisation des systèmes d'information, à prendre en compte dans les méthodes de conception de ces systèmes.

Ces deux aspects d'un système d'information permettent d'expliquer certaines différences d'appréciations de la méthode Merise, selon que l'on est gestionnaire ou informaticien. Pour un gestionnaire, une activité manuelle déclenchée par un flux d'informations, l'organisation de cette activité et la répartition des tâches informatisées sont des préoccupations majeures du système d'information organisationnel. L'architecture des divers programmes, l'organisation des données sur des fichiers sont des préoccupations majeures de l'informaticien et concernent le système d'information informatisé.

Nous y trouverons également une explication, d'une part aux difficultés de généralisation de la méthode Merise à certains compartiments du génie logiciel, d'autre part aux inadéquations de certains développements du génie logiciel, plus orientés systèmes physiques, par rapport à l'ingénierie des systèmes d'information.

La méthode Merise traitera de la spécification et de la validation à la fois du système d'information organisationnel (SIO) et du système d'information informatisé (SII).

Statique et dynamique du système d'information

Les premières méthodes de conception de systèmes d'information s'appuyaient essentiellement sur une approche par les traitements des données. Les concepteurs identifient une typologie de traitements, puis spécifient et organisent les informations en fonction de leur utilisation par chaque traitement.

On constate ainsi une similitude totale entre la structure des informations mémorisées, utilisées pour le traitement, et la structure des informations perçues par l'utilisateur conformément à son besoin ponctuel.

Avec l'introduction des bases de données, l'idée de séparation données-traitements s'est diffusée ; elle a été adoptée par les méthodes de la deuxième

génération. Notons toutefois que cette séparation est essentiellement artificielle ; les données n'ont d'usage qu'à travers les traitements, les traitements ne peuvent fonctionner sans données.

Dans la méthode Merise, nous retrouvons cette distinction entre données et traitements.

- Les données représentent *l'aspect statique du système d'information : ce qui est*. Les données présentent, dans leur signification, une certaine stabilité et une invariance dans le temps. Cette signification (sémantique) est essentiellement déterminée par le type d'activité de l'entreprise.
- Les traitements représentent *l'aspect cinématique du système d'information : ce qui se fait*. Les traitements, et en particulier leur organisation, présentent une plus grande variabilité, en fonction essentiellement de l'évolution des besoins. Par abus de langage, on parlera de dynamique plutôt que de cinématique.

Ces deux volets, données et traitements, constituent une composante fondamentale de Merise.

L'analyse des données d'une part et celle des traitements d'autre part, ne s'effectuent pas dans une ignorance mutuelle : lorsque le concepteur analyse et spécifie les données, il doit garder présent à l'esprit que la justification d'une information est issue des traitements, sans pour cela conditionner sa structuration.

Inversement, lors de l'analyse et la spécification des traitements, le concepteur s'intéressera essentiellement au fonctionnement du système d'information, sans systématiquement associer les informations utilisées (voir figure 2.10).

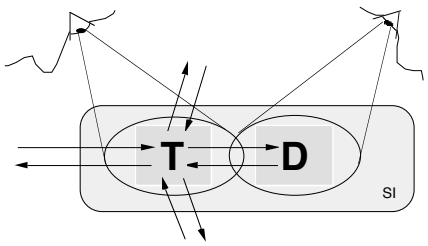


Figure 2.10 : Analyse séparée données-traitements.

Cette alternance de préoccupations, cette relative indépendance dans la description des données et des traitements, nécessitera des confrontations pour valider la cohérence entre données et traitements, qui demeurent indissociés au niveau du système d'information naturel.

La mise en œuvre des analyses et spécifications successives des données et des traitements sera précisée par la démarche de la méthode.

Système d'information naturel ou artificiel et évolution de l'entreprise

Nous avons vu précédemment que tout corps social organisé, en particulier les entreprises ou les administrations, pouvait être considéré comme un objet que l'on peut modéliser au travers d'un système de niveau 9 de complexité. Ce système décide de ses objectifs et adapte son organisation pour les atteindre ; il fait donc naturellement évoluer ses trois sous-systèmes : système opérant (SO), système de pilotage (SP) et système d'information (SI).

Nous avons ensuite vu que l'informatisation d'un système d'information, finalité de la méthode Merise, conduisait à la réalisation d'un système artificiel (SII) greffé au sein d'un système naturel (SIO).

Dans la conception de ce système artificiel, l'état actuel des connaissances ne permet pas de proposer une méthode pratique et efficace de conception de systèmes d'information qui s'appuie sur des modèles de systèmes de niveaux 8 et 9, c'est-à-dire qui intégrerait la prise en compte de l'adaptation à sa propre évolution (système d'information auto-adaptatif ou auto-évolutif).

Aussi, il ne nous est pas actuellement possible de concevoir des systèmes d'information dotés de capacités d'auto-adaptation leur permettant d'accompagner l'évolution de l'entreprise et de son système d'information naturel. Cela revient à dire que les méthodes de conception actuellement proposées (dont Merise) modéliseront l'entreprise, seulement au niveau 7 de complexité, c'est-à-dire à organisation stabilisée.

Cette hypothèse a une conséquence fondamentale. La modification de l'organisation de l'entreprise (niveau 8) et a fortiori la modification de ses objectifs (niveau 9) entraînent une évolution relativement continue de son système d'information naturel ; évolution qui nécessitera en conséquence, des re-conceptions successives, externes et ponctuelles, du système d'information artificiel le supportant.

Ainsi, à chaque évolution structurelle de l'environnement ou de l'entreprise, le système d'information naturel va immédiatement tenter de s'adapter, alors que le système d'information artificiel devrait être modifié par ses concepteurs. Au cours du temps, l'accumulation des évolutions du système d'information naturel va induire un écart avec le système d'information artificiel ; le rattrapage de ce décalage nécessitera une action de *maintenance* (voir figure 2.11).

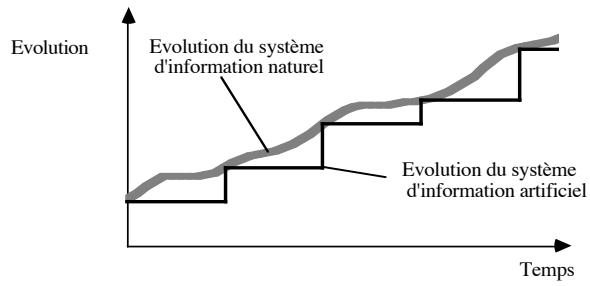


Figure 2.11 : Evolutions comparées des systèmes d'information naturel et artificiel.

L'existence de cette dérive entre les deux systèmes d'information et les actions consécutives de maintenance nous suggère quelques réflexions.

- Il importe avant tout de faciliter les actions de maintenance. Les progrès enregistrés dans les outils de développement de logiciel (langages, SGBD, AGL) rendent plus facile et rapide la mise en œuvre des évolutions. Les méthodes de génie logiciel (modularité, structuration, approche objet) œuvrent dans le même sens. Enfin la recherche d'éléments stables sur lesquels la conception initiale pourra s'appuyer doit être une préoccupation des méthodes de conception. Dans la méthode Merise, cette recherche des éléments les plus stables (ou invariants) se traduira par la mise en évidence de niveaux d'abstraction (voir chapitre 3, Les raisonnements ou cycle d'abstraction).
- La fréquence et l'importance de la maintenance ainsi que la proportion de la partie informatisée dépendent totalement de la vitesse d'évolution du système d'information naturel (voir figure 2.11). Une entreprise dont l'activité est stabilisée aura un système d'information naturel à évolution lente ; on pourra ainsi en informatiser une grande partie ; les opérations de maintenance seront espacées. Une entreprise soumise à un environnement perturbé, dont l'activité est en constante adaptation, aura un système d'information naturel à évolution rapide ; les opérations de maintenance seront quasi continuelles et inciteront certainement les concepteurs à limiter la part du système d'information informatisé, voire à concevoir des systèmes d'information " jetables ".

Cette succession d'évolutions pour le système d'information artificiel (ou informatisé) rythmera son " cycle de vie " : des évolutions mineures se traduiront par des révisions puis de nouvelles versions de l'application, des remises en cause profondes donneront naissance à de nouvelles générations.

3

Les trois composantes de Merise

Comme nous l'avons présenté en introduction, une méthode de conception de système d'information s'inscrit dans trois dimensions (voir figure 3.1) exprimant :

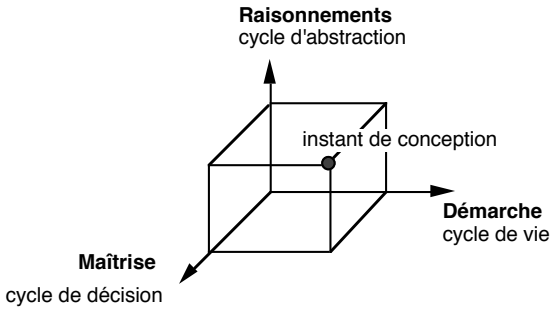
- la démarche ou cycle de vie,
- le raisonnement ou cycle d'abstraction,
- la maîtrise ou cycle de décision.

La mise en œuvre de la méthode Merise doit toujours se repérer par rapport à ces trois dimensions. Tout instant de la conception doit pouvoir se situer dans ce référentiel.

La démarche ou cycle de vie

La dénomination de ce cycle traduit le caractère « vivant » du système d'information, présentant une conception, une gestation, une naissance, une croissance, une évolution, et une mort... puis une renaissance. Dans le cas d'un système d'information, on peut distinguer trois grandes périodes : la conception, la réalisation et la maintenance.

La méthode Merise propose, pour le déroulement du cycle de vie, le découpage de ces grandes périodes en différentes étapes (voir figure 3.2).



Fgure 3.1 : Repérage dans les trois dimensions de Merise.

La période de conception se découpe en trois étapes : le schéma directeur,

l'étude préalable et l'étude détaillée. La période de réalisation se décompose , elle aussi, en trois étapes: l'étude technique et la réalisation logicielle, la mise en service.

Nous allons décrire succinctement ces différentes étapes définies par la méthode Merise, ces étapes seront étudiées en détail dans la quatrième partie.

Le schéma directeur

Première étape de conception, le schéma directeur définit le cadre général du développement des systèmes d'information principalement en termes d'objectifs et de contraintes. Il détermine, pour les systèmes d'information :

- le découpage en domaines,
- les orientations d'informatisation,
- les axes organisationnels,
- les options socio-personnelles,
- la politique matérielle et logicielle,
- la planification globale du développement,
- les cadres budgétaires.

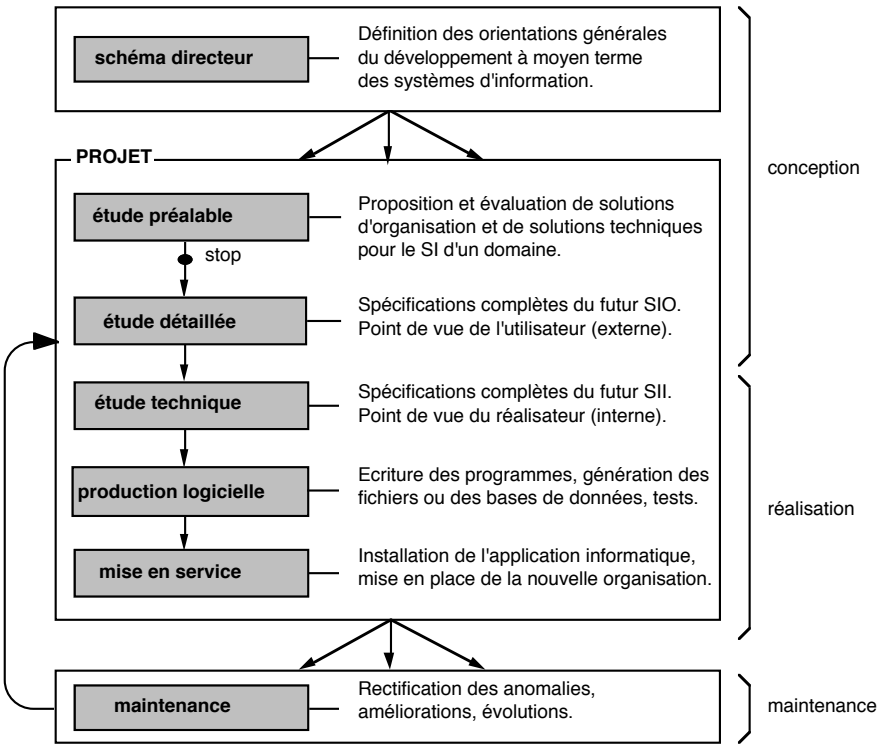


Figure 3.2 : Le cycle de vie de Merise.

L'étude préalable

Dans la ligne schéma directeur, l'étude préalable est une étape fondamentale de Merise. Elle permet, avant de se lancer complètement dans un projet, d'élaborer globalement différentes solutions et d'en évaluer les diverses conséquences. Cette étape est confrontée à deux exigences contradictoires :

- une durée relativement courte (quelques mois au plus),
- une analyse suffisamment complète pour permettre une évaluation raisonnable.

Elle portera en conséquence sur un sous-ensemble représentatif du domaine étudié. L'étude préalable permet de proposer des solutions en précisant pour chacune :

- le processus de fonctionnement du domaine,
- le degré et le type d'automatisation,
- la perception des informations,
- le coût des moyens à mettre en œuvre (informatique en particulier),
- les délais et étapes transitoires,
- les avantages et contraintes de la solution,
- la situation par rapport au schéma directeur.

L'étude détaillée

Elle permet, à partir des choix issus de l'étude préalable, de spécifier complètement le futur système d'information. Cette conception comporte deux phases :

- La *conception générale*, dont l'objet est d'étendre à l'ensemble du domaine les principes de fonctionnement retenus sur le sous-ensemble représentatif. Les différentes spécifications sont complétées et validées.
- La *conception détaillée*, qui produit, au niveau de chacune des tâches à automatiser, une description complète en termes de support (dessin écran, imprimé), d'algorithme (règles de calcul, de contrôle...), d'action sur les données (mise à jour, consultation).

L'étude détaillée permet d'obtenir, pour l'utilisateur, une description complète et contractuelle du futur système d'information organisationnel. Elle permet également de réajuster les évaluations de moyens, coûts et délais estimés dans l'étude préalable.

L'étude technique

Elle est la traduction informatique des spécifications issues de l'étude détaillée. Elle permet de déterminer :

- la structure informatique de la base de données,
- l'architecture des programmes (transactionnel et batch),
- la structure de chaque programme et des accès aux données.

La position de cette étape est souvent ambiguë. Demeurant étape d'étude, elle peut être considérée comme la partie informatique de l'étude détaillée. Toutefois, son aspect fortement technique la rend très proche de la réalisation, s'assimilant à la spécification de celle-ci.

La production de logiciel

Elle consiste à traduire, dans des langages appropriés, les spécifications exprimées dans les étapes précédentes. Cette production comprendra, entre autres :

- l'écriture des programmes dans un langage de programmation,
- la génération des fichiers ou des bases de données,
- les tests de mise au point.

A l'issue de cette étape, une recette du logiciel est effectuée, prononçant la conformité aux spécifications.

La mise en service

Elle consiste à installer les logiciels réalisés, et à mettre progressivement l'ensemble du système d'information au service des utilisateurs. Au cours de cette étape, on procède à :

- la mise au point d'un planning d'installation tenant compte des phases transitoires,
- la création et le chargement des informations de base,
- la formation des utilisateurs,
- la vérification du bon fonctionnement du logiciel,
- la mise en place progressive de la nouvelle organisation.

A l'issue de cette période de lancement, on pourra procéder, suivant les événements, à la recette provisoire puis définitive du système d'information.

La maintenance

Elle consiste à prendre en compte les évolutions apparaissant après le lancement opérationnel. Il faudrait, en fait, distinguer une étape supplémentaire, antérieure à la maintenance : le fonctionnement opérationnel.

Cette étape, qui demeure la plus importante de la vie d'un projet, ne devrait pas se manifester autrement que par des tâches d'exploitation. Les évolutions conduisant à une modification de l'application initiale proviennent des progrès

technologiques, de la modification de l'environnement et des utilisateurs.

Cette maintenance se traduit par un « rebouclage » du cycle de vie :

- étude de l'impact de la modification,
- spécification des modifications à effectuer,
- réalisation,
- mise en service.

La remise en cause

Cette étape est essentiellement un constat d'évolutions trop importantes pour relever d'une simple maintenance. Ces évolutions peuvent trouver leur origine dans l'ancienneté de l'application, l'obsolescence technologique prétexte à une révision totale de la conception du système d'information ou un changement important dans l'activité ou dans les principes d'organisation.

Si le constat conclut à une remise en cause nécessaire du système d'information, le cycle de vie reprendra soit à une nouvelle étude préalable, soit plus radicalement à partir d'un nouveau schéma directeur.

Les raisonnements ou cycle d'abstraction

Lors de la conception d'un système d'information, différents problèmes peuvent se présenter, par exemple :

- la description du fonctionnement de l'activité,
- la définition de règles de gestion,
- la définition des informations,
- la répartition des traitements entre l'homme et la machine,
- l'organisation physique des fichiers,
- le découpage en transactions,
- le choix du matériel,
- la répartition des responsabilités au sein de la structure.

Ces problèmes conduisent à faire des choix de natures différentes (gestion, organisation, techniques, matériels, etc.). Aussi est-il nécessaire d'effectuer une hiérarchisation, de rassembler des préoccupations en niveaux d'intérêts homogènes.

Cette nécessité d'aborder successivement les différents types de préoccupations a conduit à proposer différents niveaux d'abstraction, ou de hiérarchisation des préoccupations.

Le découpage en niveaux a été confirmé par la communauté internationale

[ANSI-X3-SPARC 75]. Nous retiendrons pour Merise quatre niveaux d'abstraction (voir figure 3.3) :

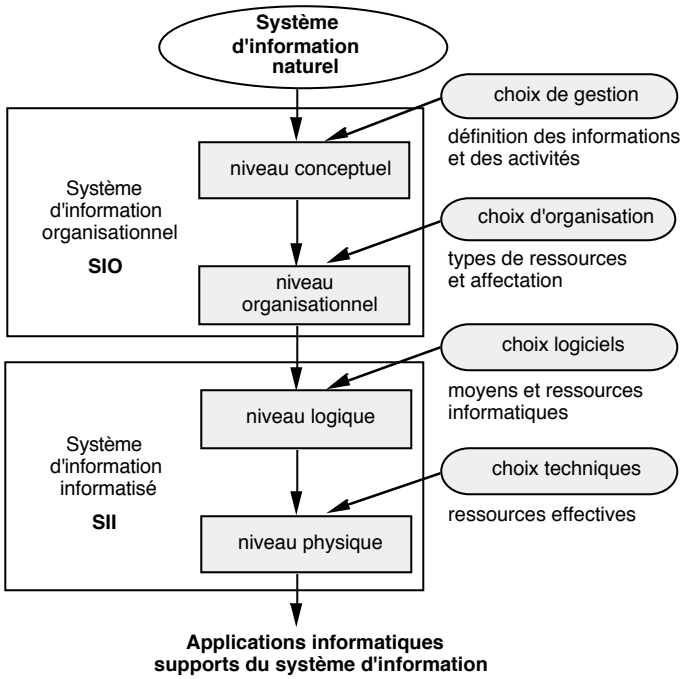
- niveau conceptuel ,
- niveau organisationnel ,
- niveau logique ,
- niveau physique.

Les deux premiers niveaux sont adaptés à la conception du système d'information organisationnel (SIO), les deux derniers à la conception du système d'information informatisé (SII). Nous aurons l'occasion de revenir plus précisément sur chacun de ces niveaux tout au long de cet ouvrage. Dans cette présentation générale, nous les définirons ainsi :

- *Système d'information organisationnel (SIO) :*

Le *niveau conceptuel* exprime les choix fondamentaux de gestion (recherche des éléments stables indépendamment des moyens à mettre en œuvre, de leurs contraintes et de leur organisation).

Le *niveau organisationnel* exprime les choix d'organisation de ressources humaines et matérielles, au travers notamment de la définition de sites,



de postes de travail.

Figure 3.3 : Les quatre niveaux d'abstraction de Merise.

- *Système d'information informatisé (SII) :*

Le *niveau logique* exprime les choix de moyens et de ressources informatiques,

en faisant abstraction de leurs caractéristiques techniques précises.

Le *niveau physique* traduit les choix techniques et la prise en compte de leurs spécificités.

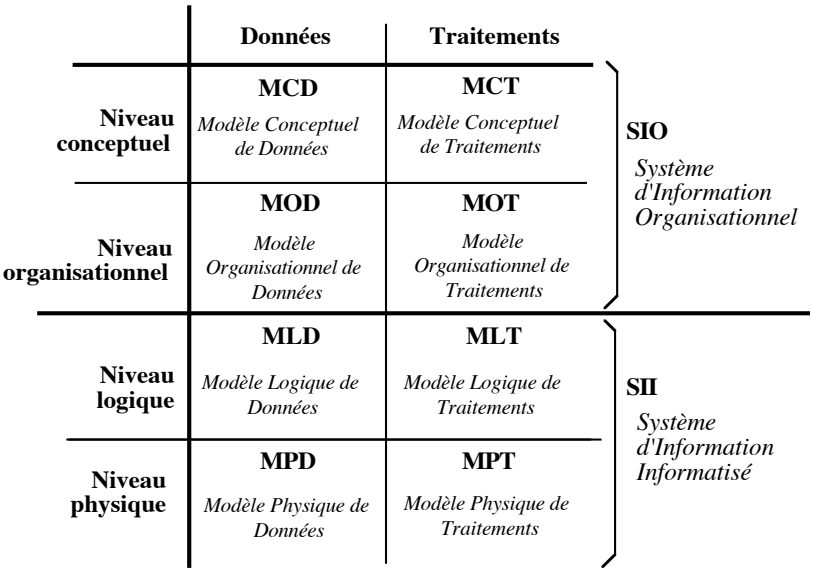


Figure 3.4 : Les différents modèles du système d'information.

A chaque niveau d'abstraction (conceptuel, organisationnel, logique, physique), pour chaque volet (données, traitements), le système d'information est représenté par un modèle (voir figure 3.4). Chaque modèle est exprimé dans un formalisme utilisant des concepts adaptés.

Au niveau conceptuel, le modèle conceptuel des données (MCD) formalise la signification des informations sur lesquelles repose le système d'information, sans contrainte technique ni économique. Le modèle conceptuel de traitement (MCT) formalise l'activité du domaine abordé, sans préciser les ressources ni leur organisation.

Au niveau organisationnel, le modèle organisationnel de traitements (MOT) décrit le fonctionnement du domaine en précisant les ressources humaines et matérielles mobilisées ainsi que l'organisation de ces ressources dans le temps et dans l'espace. Le modèle organisationnel des données (MOD) précise quelles sont parmi les données définies au niveau conceptuel (MCD) celles qui sont prises en compte par le futur système informatisé, où ces données sont localisées (répartition par site organisationnel), leur confidentialité pour chaque intervenant de l'entreprise.

Au niveau logique, le modèle logique de données (MLD) fournit une description des données tenant compte des moyens informatiques de mémorisation et de leurs conditions d'utilisation par les traitements. Le modèle logique de traitement (MLT) décrit comment les tâches informatisées définies dans les MOT précédents sont conçues en termes de logiciel.

Au niveau physique, le modèle physique de données (MPD) est une description de la ou des bases de données ou de l'ensemble des fichiers, exprimée dans la syntaxe du système de gestion de bases de données (SGBD) ou système de gestion de fichiers (SGF) adoptés. Enfin, le modèle physique de traitements (MPT) précise, pour la réalisation, les spécifications techniques des différents modules définis au niveau du MLT. Ces modules pourront être réalisés soit en langages de quatrième génération, soit de façon plus traditionnelle en langage de troisième génération (Cobol, C…).

La construction d'un système d'information, relativement aux seuls raisonnements, se traduit par un enchaînement des différents raisonnements basés sur l'utilisation des modèles et formalismes : le *cycle d'abstraction.* Ce processus doit permettre de répondre aux questions suivantes :

- Comment élaborer et exprimer les différents modèles ?
- Comment passer d'un niveau d'abstraction au suivant et transformer les différents modèles ?
- Comment confronter données et traitements pour assurer la cohérence du système ?

Ce parcours du cycle d'abstraction mettra en jeu des compétences et intérêts diversifiés. En particulier, gestionnaires - utilisateurs et informaticiens seront tour à tour concernés par l'élaboration des différents modèles.

Nous avons déjà vu cette différence de culture et de préoccupation dans la distinction entre le système d'information organisationnel (SIO) et le système d'information informatisé (SII). Sans instaurer une barrière entre partenaires gestionnaires et partenaires informaticiens, l'implication et la contribution de chacun seront très différentes selon la partie (SIO ou SII) :

- L'élaboration du système d'information organisationnel exige l'implication du gestionnaire - utilisateur ; les problèmes abordés et les solutions à proposer relèvent essentiellement de son choix. Les formalismes utilisés pour l'expression des modèles devront en tenir compte.
- L'élaboration du système d'information informatisé relève uniquement du savoir-faire des informaticiens. Les formalismes utilisés dans les modèles feront largement appel à des concepts du métier informatique.

Cette répartition des pôles d'intérêts entre gestionnaires et informaticiens ne préjuge pas de l'organisation des groupes de travail ; la répartition des rôles et des contributions sera réglée par le *cycle de décision.*

La manière dont ces préoccupations, pôles d'intérêts et de compétences émergent dans le cycle d'abstraction est illustrée dans la figure 3.5.

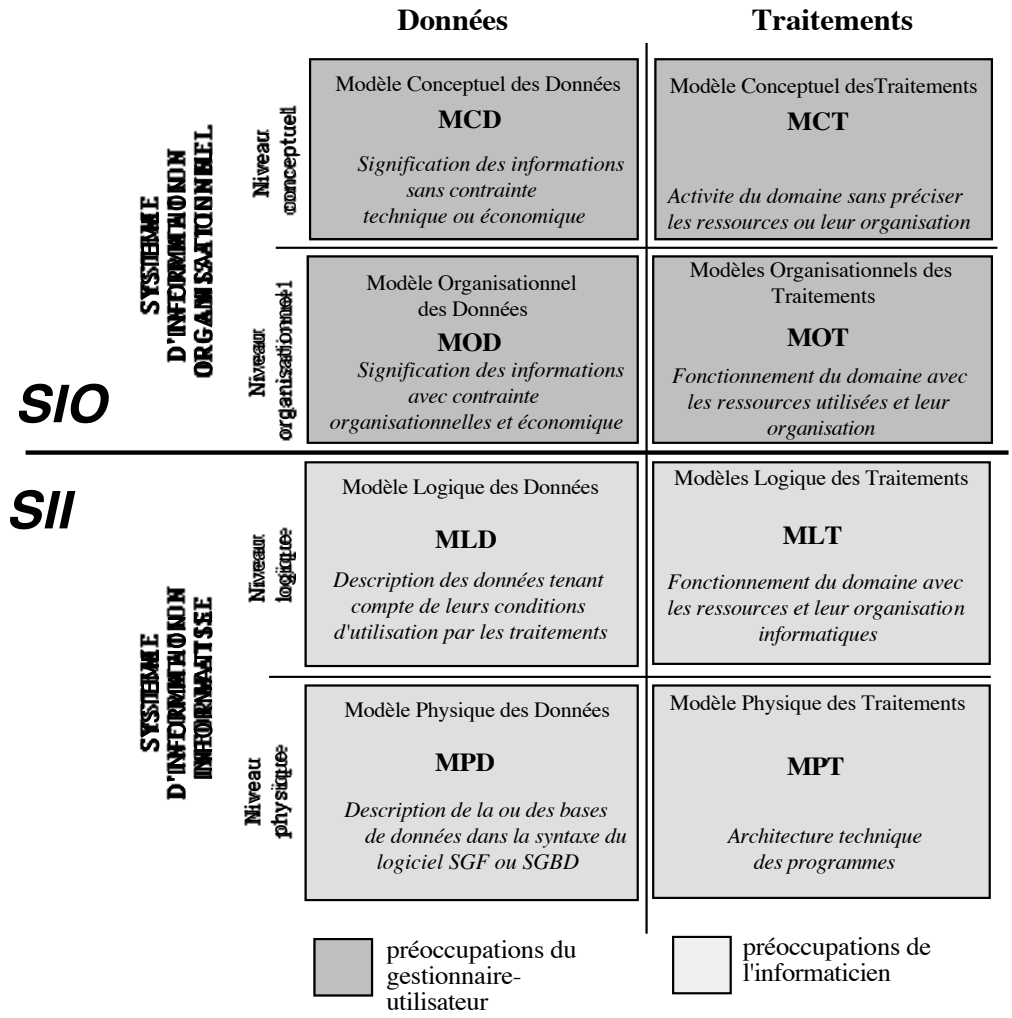


Figure 3.5 : La répartition des préoccupations entre le gestionnaire et l'informaticien pour les huit modèles de Merise.

La maîtrise du projet ou cycle de décision

Le déroulement simultané de la démarche et des raisonnements doit être maîtrisé. Dans chaque modèle, à chaque étape, des choix doivent être effectués. Vers quel projet veut-on aller ? Quels moyens veut-on lui affecter ?

La mise en œuvre de la méthode Merise se traduit, en plus, par une succession de choix permettant, d'une part, de contrôler la durée globale de la conception-réalisation, d'autre part, de définir un système en harmonie avec les objectifs généraux de l'entreprise.

Notons que la maîtrise comprend également l'ensemble des décisions d'arbitrage relatives aux coût, délai et niveau de gamme associés au projet.

Nous avons tenté d'illustrer le caractère antagoniste de ces trois composantes par un triangle Coût-Délai-Niveau de gamme (voir figure 3.6). Il traduit qu'un projet ne peut à la fois être de coût réduit, de niveau de gamme élevé et avec des délais serrés ; privilégier une composante (se rapprocher d'un sommet) se fait au détriment des autres composantes.

Ainsi dans le cadre d'un projet P, des préférences différentes peuvent conduire à deux variantes P1 et P2 :

- P1 met en priorité un niveau de gamme élevé, accompagné d'un coût élevé sans trop de contrainte sur le délai,
- P2 met en priorité un coût raisonnable en réduisant le niveau de gamme, toujours sans trop de contrainte sur le délai.

La responsabilité de ces différents choix incombe à un troisième partenaire. Après l'utilisateur - gestionnaire et l'informaticien intervient le décideur (ou direction).

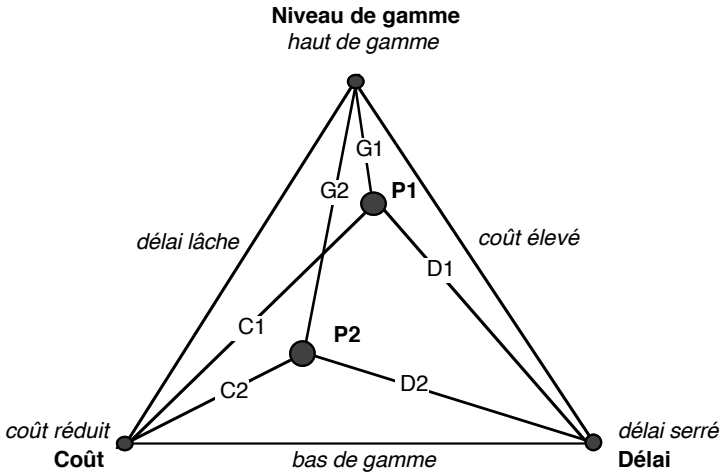


Figure 3.6 : Délai, coût et niveau de gamme liés à la maîtrise d'un projet.

La répartition des rôles entre ces différents partenaires, les diverses décisions à prendre au fur et à mesure de l'avancement du projet, la hiérarchisation des choix et les résultats à produire constituent les principaux éléments de cette maîtrise du projet.

L'organisation générale du projet s'articule autour de trois groupes :

- le groupe de pilotage;
- le groupe de projet;
- le groupe de validation.

La composition, le mode d'intervention et le rôle de ces groupes seront détaillés dans la partie IV de l'ouvrage. Dans la pratique, le cycle de décision est intégré dans le cycle de vie. Cela se traduit par des résultats types à l'issue de

chaque étape et par des décisions attendues, comme le montre la figure suivante :

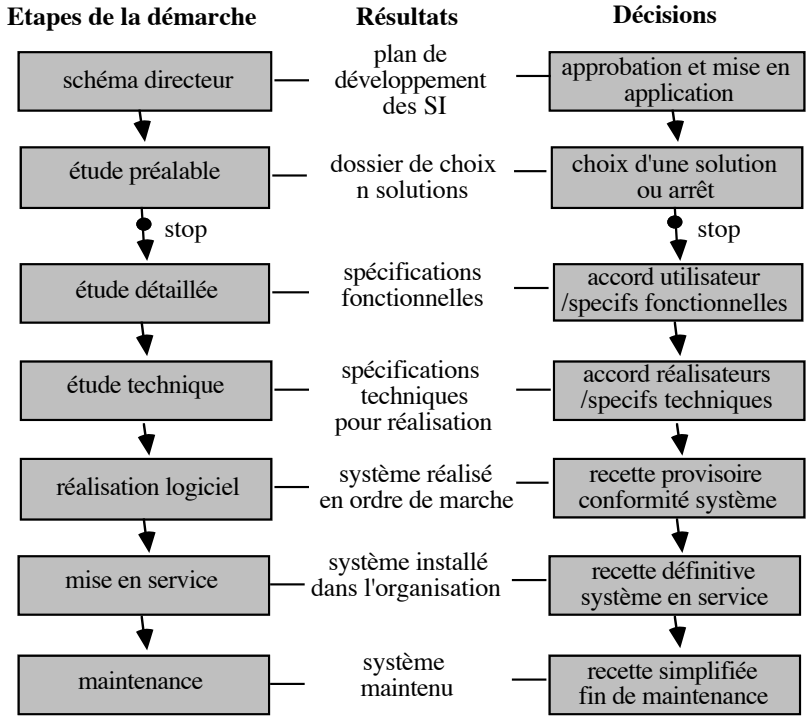


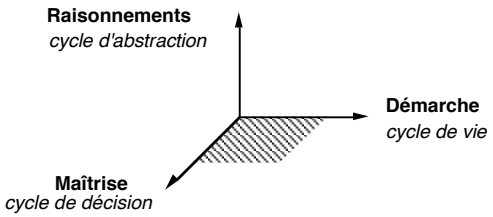
Figure 3.7 : Le cycle de décision de Merise.

Enfin, dans cette dimension, il importe de laisser une grande latitude de personnalisation par l'entreprise. Des petits projets ne mobilisent pas nécessairement les mêmes moyens de contrôle que ceux présentant un rôle stratégique pour l'entreprise.

Trois dimensions indispensables

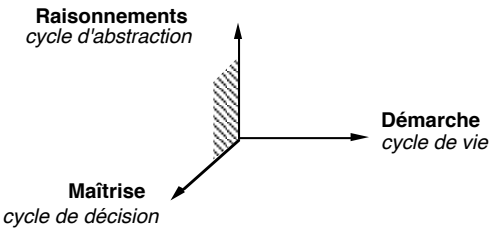
Les trois dimensions de la méthode Merise, le cycle d'abstraction, le cycle de vie et le cycle de décision sont fondamentales et indispensables. Pourrions-nous concevoir des méthodes de conception de systèmes d'information négligeant une de ces trois dimensions ?

Méthodes sans cycle d'abstraction ?



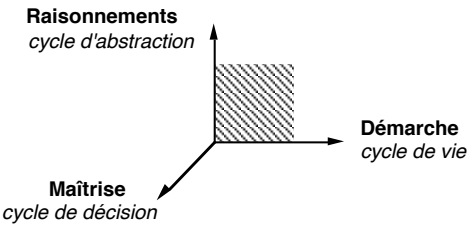
De telles méthodes sont trompeuses. Elles présentent une démarche, une succession de décisions et une organisation ; elles se limitent à la conduite du projet. Elles laissent cependant le concepteur sans technique ni instruments (modèles) lui permettant de formaliser ses raisonnements de conception et de sérier ses problèmes.

Méthodes sans cycle de vie ?



- De telles méthodes se réduisent à un ensemble de modèles et omettent le cycle de vie ; le cycle de décision est alors aussi très pauvre. Elles sont souvent intellectuellement séduisantes, mais n'indiquent au concepteur aucune démarche de mise en œuvre.
- De telles méthodes ne sont pas plus envisageables, car cela postulerait que le cycle d'abstraction est suffisamment naturel et acceptable par les décideurs de l'organisation pour conduire le déroulement de la conception. Par exemple, on ne se préoccuperait d'aspects ergonomiques ou des matériels qu'en fin de conception, sous prétexte que des réflexions de conception ne doivent pas être « polluées » par de telles considérations. Cela est bien entendu théorique.
- A cette occasion, on peut regretter que de trop nombreux ouvrages sur la méthode Merise se cantonnent à une présentation des modèles (les raisonnements du cycle d'abstraction) et escamotent la démarche qui seule propose un guide de mise en œuvre.

Méthodes sans cycle de décision ?



- Les décisions sont prises sans qu'une hiérarchie n'ait été fixée. De telles méthodes ne sont pas envisageables, car en ne se préoccupant pas du déroulement du cycle de décision, la conception du système d'information devient rapidement incontrôlable en temps et en coût. Il est nécessaire de pouvoir définir des étapes de conception et de réalisation de contenu et de durées fixés, qui permettront une maîtrise du projet, ainsi qu'une certaine répartition du travail.
- Sauf à envisager un projet en solitaire, l'absence de cycle de décision prive le projet d'une organisation. Le projet se déroule sans objectif, les décisions sont prises « par défaut », dans une absence de responsabilité.

Cheminement du processus de conception

Comme nous l'avons déjà indiqué, la méthode Merise se préoccupe principalement de la conception du système d'information organisationnel et informatisé. Une des grandes originalités de la méthode Merise est le cheminement du processus de conception qu'elle préconise, illustrée par la courbe dite du soleil (figure 3.8).

La méthode Merise préconise, lorsque cela est possible, de partir d'une compréhension du système d'information actuel, compréhension qui ne peut se faire, la plupart du temps, qu'aux niveaux physique et logique/organisationnel ; on se situe alors dans la partie gauche de la courbe de la figure 3.8. Cette compréhension se construira, par exemple, en étudiant le fonctionnement du domaine (ses activités, son organisation), en observant les différents formats des fichiers, l'architecture de la base de données, des différents documents manuels ou informatisés, etc., ainsi qu'au travers d'entretiens avec divers acteurs de l'organisation, de la direction générale aux utilisateurs directs ou indirects du système d'information actuel.

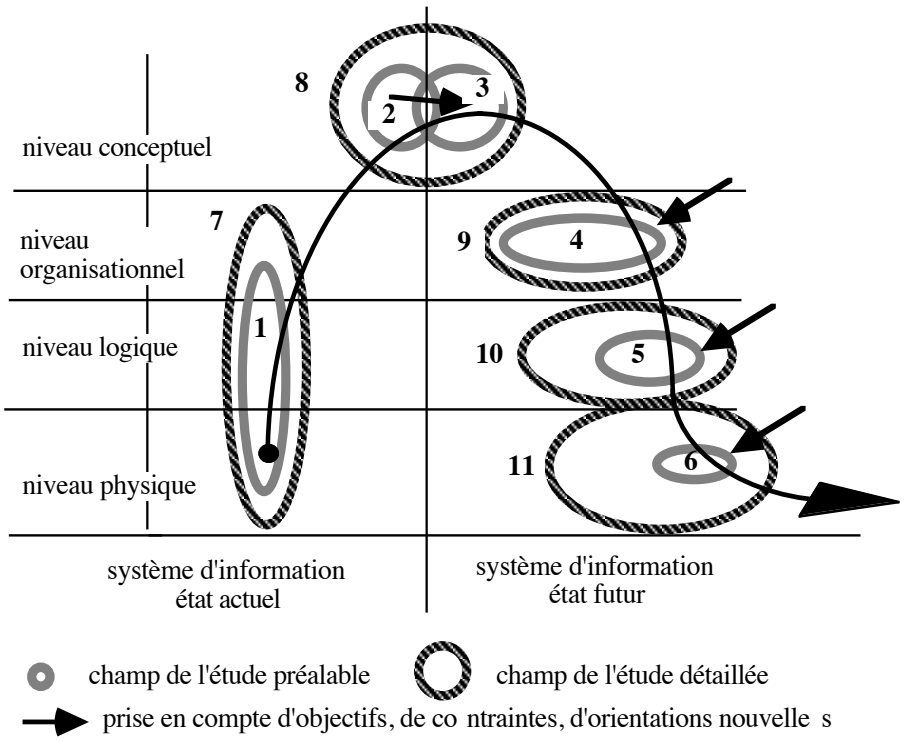


Figure 3.8 : Le cheminement de conception de Merise.

Cette étude doit permettre de faire émerger une compréhension du système actuel de niveau conceptuel, ne conservant que les grandes finalités de l'organisation, les grands choix de gestion qu'elle a pu faire consciemment ou non, abstraction faite des choix organisationnels et physiques précédemment retenus. Cette compréhension du système d'information actuel ne conduit pas obligatoirement à l'élaboration soignée de modèles de données et de traitements.

Cette modélisation conceptuelle actuelle du système d'information permet de mieux appréhender les nouveaux choix conceptuels que l'entreprise souhaite faire pour son développement futur et ainsi de concevoir les modèles conceptuels que devrait respecter le futur système d'information. On se situe alors dans la partie droite de la courbe de la figure 3.8 concernant la conception du futur système d'information. A partir de cette définition conceptuelle du futur système, des modélisations nouvelles prenant en compte des choix d'organisation et des choix techniques permettront de définir plus complètement pour sa réalisation le système futur.

En fait, le parcours de cette courbe du soleil représente la projection de la démarche (cycle de vie) sur les raisonnements. Comme l'indique la courbe, la conception avec la méthode Merise s'effectue suivant un parcours itératif des modèles du cycle d'abstraction. L'étude préalable permet de comprendre le système actuel et d'élaborer des propositions de solutions, du niveau

conceptuel au niveau technique, avec un degré de détail suffisant pour permettre les évaluations quantitatives et qualitatives nécessaires au choix d'un système futur. L'étude détaillée et l'étude technique, sur les niveaux qui les concernent, généraliseront et détailleront la solution préalable retenue.

Niveaux d'abstraction, couverture du domaine étudié et degré de détail

Le processus de conception progressif, et le parcours des niveaux selon la courbe du soleil ont, dans la pratique, corrélé la prise en compte des niveaux successifs d'abstraction et l'affinement du degré de détail. En particulier, dans la modélisation des traitements, le passage au niveau inférieur (du conceptuel à l'organisationnel, par exemple) se traduit généralement par une décomposition des activités décrites au niveau supérieur.

Nous voulons, dans ce paragraphe, expliciter les différentes dimensions qui interviennent dans la modélisation :

- les niveaux d'abstraction,
- la couverture du domaine étudié,
- le degré de détail.

La succession des *niveaux d'abstraction* traduit la prise en compte progressive de préoccupations différentes (qui, dans la réalité, sont bien évidemment mélangées...) :

- le niveau conceptuel (Que fait-on et pourquoi ? Que signifient les informations ?),
- le niveau organisationnel (Comment et avec quels moyens humains et informatiques ?), préoccupation du gestionnaire,
- le niveau logique (Comment et avec quels moyens logiciels ?), préoccupation de l'informaticien,
- le niveau physique (Comment concrétiser, compte tenu d'un environnement technique ?).

La spécification des données et des traitements s'effectue à chaque niveau par des modèles exprimés dans des formalismes.

La *couverture du domaine* exprime la part des activités et informations étudiées selon les étapes de la démarche. Elle reste partielle en étude préalable (notion de sous-ensemble représentatif) pour tendre vers l'exhaustivité en étude détaillée. Cette dimension d'étendue du champ étudié est d'ailleurs illustrée dans la courbe du soleil figure 3.8.

Le *degré de détail* dépend essentiellement de la démarche et du processus de conception : on peut analyser un système actuel avec un degré de détail moyen,

puis proposer des solutions futures avec un degré assez global, enfin affiner le détail au fur et à mesure de la progression du projet. Le degré de détail doit également s'adapter, au gré du concepteur, à l'ampleur d'un projet en fonction des contraintes de temps ; une étude préalable sur un domaine vaste et des délais réduits imposera une « maille d'analyse » assez grossière. Le concepteur doit donc pouvoir élaborer, à un niveau d'abstraction donné, un modèle global, moyen ou détaillé.

La figure 3.9 illustre le croisement de la couverture du domaine et du degré d'abstraction. On voit que, classiquement, l'étude préalable combine couverture partielle et « maille d'analyse » globale. L'étude détaillée comporte d'abord une extension de modélisation à degré de détail constant (la conception générale) puis entreprend un affinage du degré de détail (la conception détaillée).

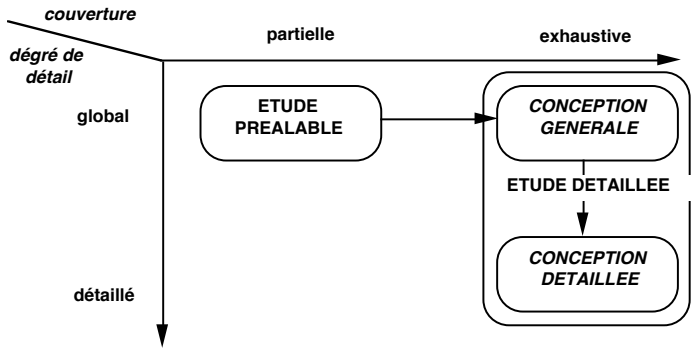


Figure 3.9 : Le croisement de la couverture du domaine et du degré de détail.

La figure 3.10 illustre le croisement du niveau d'abstraction et du degré de détail.

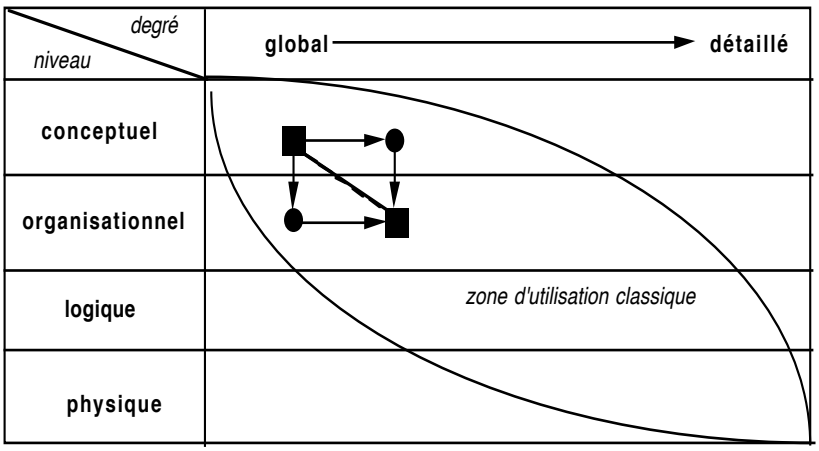


Figure 3.10 : Le croisement du niveau d'abstraction et du degré de détail.

La diagonale de la figure représente le mode de mise en œuvre classique des modèles évoqués précédemment. Ce parcours montre que le passage d'un modèle à un autre combine le changement de niveau (donc l'intégration de

nouvelles préoccupations) et l'affinage du détail. Cependant, deux cheminements sont possibles et rendent, dans chaque cas, nécessaire l'accroissement du détail à niveau constant. Dans la compréhension et la mise en œuvre des modèles et de leurs formalismes associés, il est donc important de dissocier le niveau d'abstraction et le degré de détail.

Comment permettre au concepteur de moduler le degré de détail d'un modèle selon les circonstances dans le même niveau d'abstraction ? Les techniques de décomposition/composition, inspirées des méthodes anglo-saxonnes (décomposition hiérarchique de l'« analyse structurée ») nous offrent des solutions.

Ces techniques de décomposition sont fondées sur le principe « des boîtes qui se décomposent en boîtes, qui elles-mêmes se décomposent en boîtes, qui... », apportant ainsi d'une part un effet de grossissement progressif, d'autre part une adéquation de la taille des schémas aux capacités d'analyse des lecteurs. Les méthodes SADT/Idef0 et SA appliquent ces comportements, en ne différenciant pas toutefois la notion de niveau.

Dans le cadre de la méthode Merise, nous suggérons d'adopter ce principe essentiellement pour les modèles de traitements, et nous proposons deux techniques à mettre en œuvre, à niveau d'abstraction constant : la décomposition hiérarchique à concept constant et la stratification des modèles types.

Dans la technique de *décomposition hiérarchique,* l'activité initiale et les activités issues de la décomposition sont de même type de concept. Pour illustrer cette décomposition hiérarchique, la figure 3.11 montre l'exemple de la décomposition d'une unité de traitements (UT) en unités de traitements plus fines dans un modèle logique de traitements.

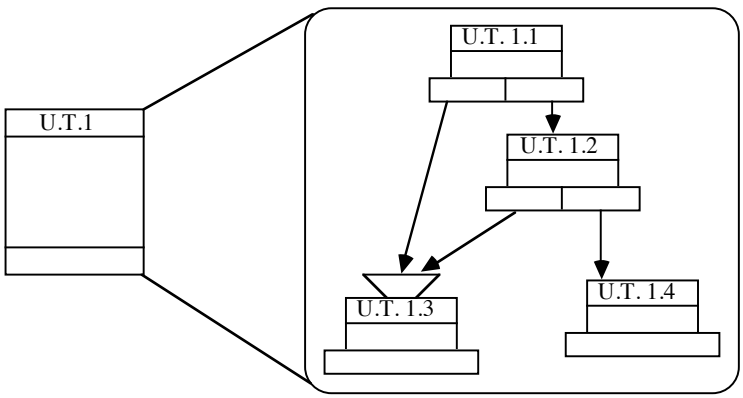


Figure 3.11 : Décomposition hiérarchique dans un modèle logique de traitements.

Dans la technique de *stratification,* la décomposition s'effectue en changeant de type de concept, tout en restant au même niveau. Pour illustrer cette stratification, la figure 3.12 montre l'exemple de la décomposition d'une phase d'un modèle organisationnel de traitements global en tâches d'un modèle

organisationnel de traitements détaillé.

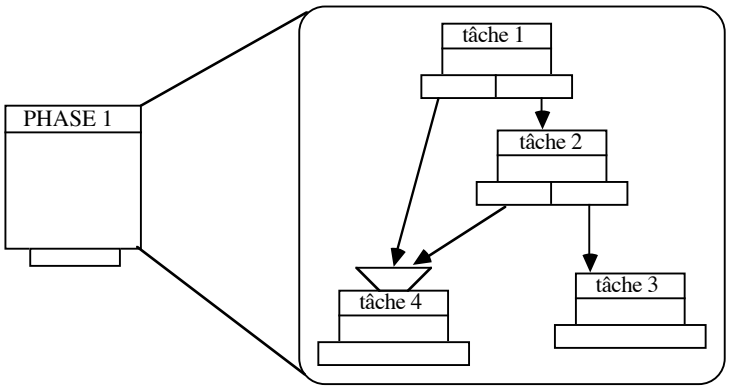


Figure 3.12 : Décomposition par stratification au niveau organisationnel de traitements.

Dans la suite de l'ouvrage, nous aborderons la technique de décomposition des traitements, pour chaque niveau de modèle, sous le terme de modularité de la modélisation des traitements.

Le recours à cette décomposition (associée à des précisions dans la description des données et des règles utilisées dans les traitements) est également préconisé dans Merise/2 [Panet, Letouche 94] sous la forme de nouveaux types de modèles de traitements dits *analytiques* .

On peut également s'interroger sur l'apport de cette technique de décomposition/composition dans le cas des modèles de données (en particulier au niveau conceptuel). Cependant nos réflexions actuelles ne sont pas suffisamment avancées pour constituer une proposition opérationnelle.

4

Différents types de systèmes d'information et la méthode Merise

La définition générale d'un système d'information que nous avons proposée au début du chapitre 2, « Contribution de la science des systèmes », et son processus d'informatisation avec la méthode Merise recouvrent en fait une certaine diversité de situations. Nous avons voulu, dans ce chapitre, prolonger nos réflexions théoriques pour analyser différents types de systèmes d'information et évaluer l'adéquation de la méthode Merise pour leur informatisation.

Le lecteur souhaitant prendre directement connaissance du contenu détaillé de la méthode Merise pourra passer directement à la deuxième partie et revenir ultérieurement sur ces développements.

Le paradigme de R. Anthony

Depuis 1965, date de sa publication, l'ouvrage de R. Anthony [Anthony 65] a fourni un cadre d'analyse des processus de planification et de contrôle en management. Au fil des ans ce cadre a été érigé en véritable paradigme. Il propose une typologie des processus de planification et de contrôle stratifiée à trois niveaux :

- *La planification stratégique*. C'est le processus de décision qui permet l'élaboration des objectifs et de la politique de l'entreprise, sur leur changement éventuel, sur les ressources utilisées pour atteindre ces objectifs, sur les politiques d'acquisition, d'utilisation et d'organisation des ressources.
- *La planification et le contrôle managérial*. C'est le processus par lequel les gestionnaires s'assurent que les ressources sont utilisées de façon efficace et rentable. La planification managériale consiste en la conception de plans à partir des objectifs définis dans la planification stratégique. Le contrôle managérial portera sur l'identification et la correction des

déviations des plans précédemment conçus.

- Le *contrôle opérationnel* ou *gestion opérationnelle*. C'est le processus qui permet de s'assurer que les tâches spécifiques sont assurées de façon efficace et rentable.

Nous retiendrons plutôt les termes plus systémiques ([Le Moigne 74] et [Mélèse 72]) de régulation, pour le contrôle opérationnel, et de pilotage pour la planification et le contrôle managérial comme l'indique la figure 4.1.

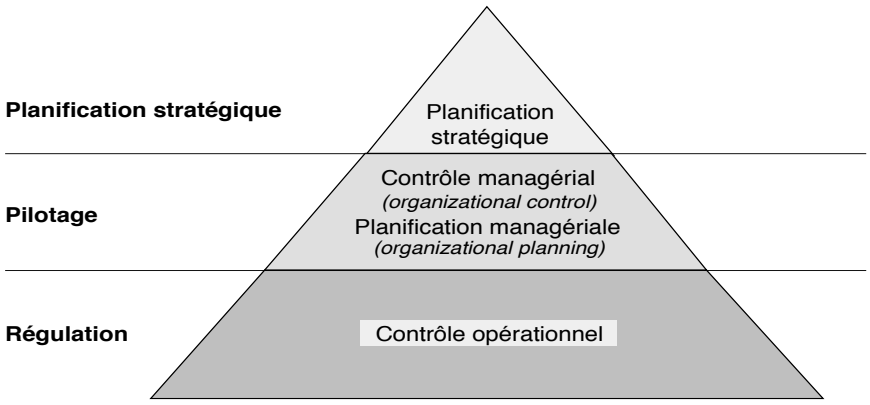


Figure 4.1: Le cadre conceptuel de R. Anthony (1965).

Nature des décisions

On peut également analyser la nature des décisions liées à ces trois types de planification (ou contrôle), en fonction de l'horizon de ces décisions :

- la planification stratégique conduit à des décisions majeures dont les conséquences sont à long terme (quelques mois à quelques années),
- le pilotage (planification et contrôle managérial) conduit en général à des décisions dont les conséquences sont à moyen terme (quelques semaines à quelques mois),
- la régulation (contrôle opérationnel) concerne principalement des rythmes inférieurs au mois et conduit à des décisions dont les conséquences sont à court ou très court terme (quelques heures à quelques jours).

On peut également s'intéresser au champ couvert par ces décisions :

- la planification stratégique conduit à des décisions dont la portée est très globale car elles peuvent engager l'avenir de toute l'entreprise,
- la régulation, contrôle opérationnel, conduit à des décisions dont la

portée est limitée, par exemple à une partie d'une chaîne de montage d'une entreprise,

- le pilotage conduit en général à des décisions dont la portée est intermédiaire.

Les caractéristiques échéance et champ couvert des décisions types associées à la planification stratégique, le pilotage et la régulation peuvent s'illustrer comme sur la figure 4.2 [Le Moigne 74].

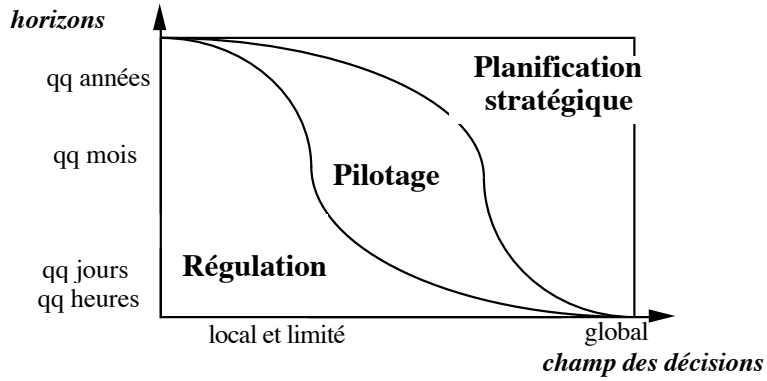


Figure 4.2: Echéances et champs couverts des décisions [Le Moigne 74].

On peut aussi s'intéresser aux problèmes et aux méthodes de résolution associés à ces décisions comme l'a fait notamment H.A. Simon dans ses recherches sur la compréhension des processus décisionnels. Pour lui, une décision sera *programmable* (sur ordinateur) si « elle est répétitive et routinière, dans la mesure où une procédure précise a été établie de façon que la décision ne soit pas réexaminée à chaque fois » [Simon 60]. Sinon une décision est *non programmable.*

H.A. Simon distingue, dans les décisions non programmables, les *décisions structurées,* celles pour lesquelles, une fois le problème à peu près identifié, le décideur peut faire appel à un certain nombre d'algorithmes, de structures de raisonnements lui permettant d'avancer vers sa résolution et les *décisions non structurées* (peu ou pas structurées), lorsque le problème est si complexe que le décideur a du mal à le formaliser.

Gorry et Scott Morton ont proposé un nouveau modèle combinant le paradigme de R. Anthony avec les types de décisions définis par H.A. Simon. Ce modèle met en évidence, pour la régulation, le pilotage et la planification stratégique, la nature des décisions associées. Pour la régulation, les décisions associées sont principalement des décisions fortement structurées conduisant à des décisions de choix de type tables de décisions. Ces décisions sont fortement informatisables. Pour la planification stratégique, les décisions associées sont principalement peu, voire très peu, structurées et sont en

conséquence difficilement automatisables. En ce qui concerne le pilotage, les problèmes à résoudre sont moins structurés que dans la régulation, mais les décisions sont en partie automatisables.

Typologie des systèmes d'information

Plusieurs typologies de systèmes d'information ont été proposées. Notre typologie distinguera tout d'abord une première famille : les *systèmes d'information de production,* où l'information est destinée au système opérant (SO). Ils sont souvent très proches du système opérant de l'entreprise.

Une seconde famille sera celle des *systèmes d'information destinés au système de pilotage de l'entreprise (SP)* avec ses sous-systèmes de décision, d'imagination-conception et de finalisation. En nous référant au paradigme de R. Anthony présenté en début de ce chapitre, nous distinguerons trois sous-types de systèmes : les *systèmes d'information opérationnels* associés au contrôle opérationnel, les *systèmes d'information de pilotage* associés à la planification et au contrôle managérial (pilotage), et enfin les *systèmes d'information stratégiques* associés à la planification stratégique.

Nous présentons ces différents types de systèmes d'information en mettant tout d'abord en évidence à quels sous-systèmes du système de pilotage ils sont plus particulièrement destinés (sous-systèmes de décision, d'imagination-conception et de finalisation). Puis nous nous intéressons à la nature des couplages, notamment SI-SO et SI-SP, qu'ils supportent. Enfin, nous évoquons leurs caractéristiques principales et nous indiquons si la méthode Merise nous semble (dans ce cas, sous quelles réserves) ou ne nous semble pas adaptée à leur conception.

Systèmes d'information de production

Les *systèmes d'information de production* constituent une première famille de systèmes d'information. Dans ces systèmes, l'information est presque exclusivement destinée au système opérant (SO) de l'entreprise, au point que ces systèmes sont très souvent assimilés au système opérant (SO) même de l'entreprise, comme l'illustre la figure 4.4.

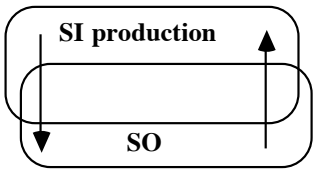


Figure 4.4 : Système d'information de production et système opérant.

Ce type de systèmes d'information se retrouve principalement dans le secteur tertiaire, car les flux transformés dans le système opérant sont en quasi-totalité des informations. C'est le cas par exemple des secteurs de l'assurance, de la

banque, des administrations, etc.

Citons aussi d'autres types de systèmes d'information de production :

- les systèmes bureautiques (traitement de texte, courrier…),
- les guichets électroniques (cartes de crédit, banques, points de ventes…).

La structure de ces systèmes d'information de production est d'une *très grande stabilité*. Ce type de systèmes d'information est un terrain privilégié de la méthode Merise.

Systèmes d'information opérationnels

Les *systèmes d'information opérationnels* sont très directement tournés vers une représentation, une coordination opérationnelle de l'activité du système opérant. Notons que ce système opérant peut être un système d'information de production comme présenté précédemment. Il y aura alors un recouvrement très important entre ces deux types de systèmes d'information.

Ces systèmes d'information opérationnels, parfois appelés *systèmes d'information primaires,* assurent ainsi un couplage très fort système d'information-système opérant pour le système de pilotage afin que son sous-système de décision puisse conduire la régulation générale du système opérant, comme l'illustre la figure 4.5.

Ces systèmes d'information opérationnels se caractérisent par un volume important de données à mémoriser. Les informations sont élémentaires (ou primaires) du fait qu'elles représentent des transactions de différents flux primaires traités par le système opérant de l'entreprise, ou représentent des états associés à l'activité de cette dernière. Ces informations sont d'abord destinées à être réutilisées par le système opérant ; lorsqu'elles sont utilisées par le système de pilotage, elles sont destinées à des décisions à court terme (niveau opérationnel). Les traitements effectués sont très formalisés (règles précises) et fortement répétitifs.

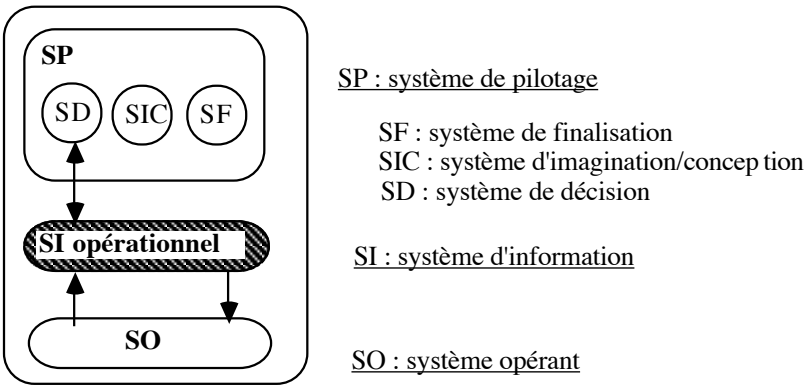


Figure 4.5 : Les couplages des Systèmes d'Information Opérationnels.

Dans ce type de systèmes d'information, on trouvera entre autres les diverses

automatisations des processus administratifs, qui furent les premiers objectifs de l'informatisation : gestion du personnel (*paie…),* gestion du matériel (*tenue des stocks…),* gestion commerciale (*suivi des commandes…),* comptabilité (*comptes, journaux, grands-livres…),* gestion de production (*GPAO),* etc.

Enfin, notons que la structure de ces systèmes d'information opérationnels est d'une assez *grande stabilité,* liée à la nature de leur système opérant.

Ce type de systèmes d'information est le terrain privilégié de la méthode Merise, c'est d'ailleurs pour ces systèmes que la méthode Merise a été conçue à l'origine.

Systèmes d'information de pilotage

Les *systèmes d'information de pilotage* sont plutôt destinés à fournir les informations nécessaires à la prise de décisions de type planification et contrôle managérial ; ils privilégient les échanges avec le système de pilotage. On les appelle aussi des systèmes d'information *secondaires.*

Ces systèmes assurent ainsi un couplage très fort SI-SP principalement avec les sous-systèmes de décision et d'imagination/conception du SP, comme l'illustre la figure 4.6.

Notons que, si la part et la spécificité des traitements décisionnels présents dans ces systèmes est importante, ces systèmes de pilotage sont parfois appelés systèmes d'information d'aide à la décision (SAD), ou systèmes interactifs d'aide à la décision (SIAD) si une forte interactivité et la mise en œuvre de modèles résolutoires complexes sont nécessaires.

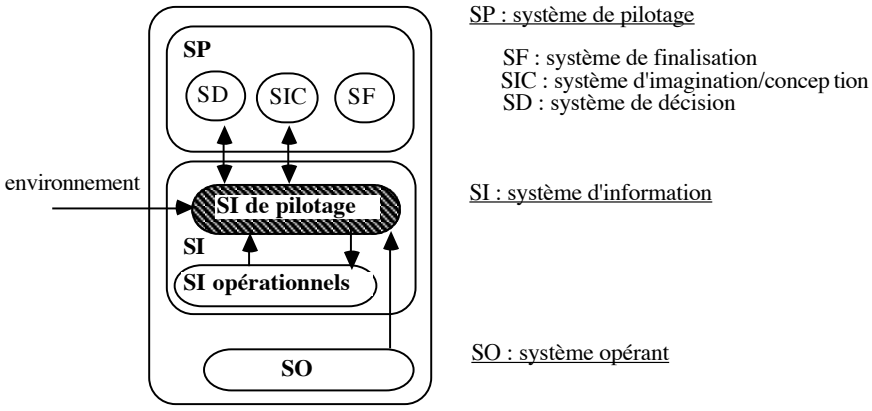


Figure 4.6 : Les couplages des Systèmes d'Information de pilotage.

Ces systèmes d'information de pilotage se caractérisent par une très grande variété d'informations, mais de volumes généralement limités ; les informations sont, pour la plupart, élaborées à partir d'informations primaires gérées par le système d'information opérationnel et parfois prises sur l'environnement ou directement sur le système opérant. Les traitements sont souvent moins formalisés et plus aléatoires (à la demande). Ces systèmes d'information de

pilotage sont souvent assez *évolutifs*.

La méthode Merise est aussi adaptée à ce type de systèmes d'information. Cependant, si la complexité du processus décisionnel et de l'interaction avec le décideur est importante, on lui préférera par exemple des méthodes de prototypage ou des méthodes issues du génie cognitif comme KOD [Vogel 88] ou KADS [KADS 89].

Systèmes d'information stratégiques

La définition des systèmes d'information stratégiques est ambiguë. Nous distinguerons deux grands types de systèmes stratégiques : les systèmes d'information à portée stratégique que nous appellerons systèmes stratégiques d'information (SSI) et les systèmes d'information de support à la stratégie que nous appellerons systèmes d'information stratégique (SIS). Cette différenciation est assez voisine de celle proposée par H. Tardieu et B.Guthmann [Tardieu & Guthmann 91] qui distinguait les systèmes d'information stratégiques (SI-S) et les systèmes d'information stratégique (S-IS).

Systèmes stratégiques d'information (SSI)

Les systèmes stratégiques d'information (SSI) peuvent en fait être assimilés à des systèmes d'information opérationnels ou à des systèmes d'information de pilotage permettant d'informatiser une activité de l'entreprise qui apparaît comme stratégique, « au sens où cette activité permettra, dans le cadre de la stratégie retenue par l'entreprise, de procurer un avantage concurrentiel durable ». On peut citer l'exemple, maintenant classique, de l'American Hospital Supply, fournisseur d'hôpitaux américains qui passa un partenariat avec ses clients en installant chez eux des terminaux leur permettant de consulter un catalogue de plus de 100 000 articles et, bien sûr, de passer commande. Le système réduit les stocks chez le fournisseur, différencie le service, et rend le client captif.

Comme le souligne Tardieu et Guthmann, ces systèmes doivent souvent être construits très rapidement, principalement afin de ne pas être imités trop rapidement par un autre concurrent, et s'inscrire « *dans une fenêtre de temps qui ne mette pas en péril le coup stratégique que l'entreprise est en train de jouer* ».

La conception de ces systèmes stratégiques d'information (SSI) nous semble pouvoir être faite en utilisant la méthode Merise, sachant qu'il est nécessaire de moduler la démarche de mise en oeuvre. En effet, ces systèmes doivent souvent être conçus et réalisés dans des délais souvent très courts pour la raison précédemment évoquée. La modulation de la démarche (cycle de vie) consécutive à cet état de fait revient principalement à ajuster la dimension maîtrise (ou cycle de décision) de la méthode. Dans le cadre de la conception d'un système stratégique d'information, l'arbitrage privilégiera par exemple le

délai au détriment du coût et/ou du niveau de gamme comme l'illustre la figure 4.7.

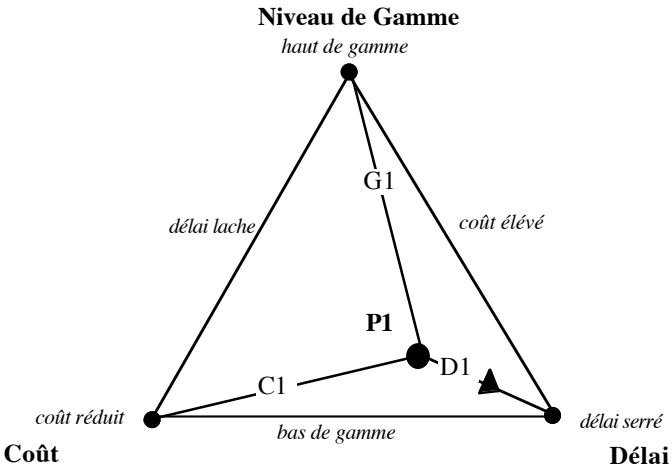


Figure 4.7 : Délai, coût et niveau de gamme lié à la maîtrise d'un projet.

Systèmes d'information stratégique (SIS)

Ces systèmes sont plutôt destinés à fournir les informations nécessaires à la prise de décisions relevant de la planification stratégique. Ces systèmes assurent ainsi un couplage très fort SI - SP principalement avec les sous-systèmes de décision, d'imagination/conception et surtout de finalisation du SP, comme l'illustre la figure 4.8.

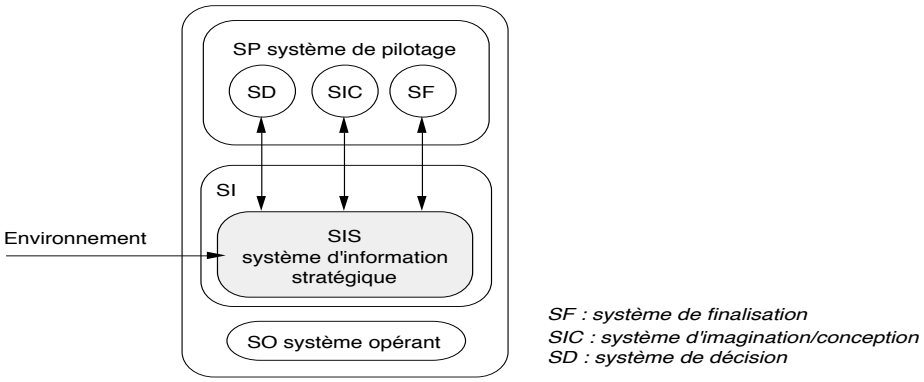


Figure 4.8 : Les couplages des systèmes d'information stratégiques.

C'est au début des années 80 qu'a émergé le concept de systèmes d'information de direction (*Executive Information Systems*). Pour Rockart et Treacy [Rockart 81a], ces systèmes doivent satisfaire les besoins en information des responsables dirigeants de l'entreprise. Les facteurs critiques de succès introduits par Rockart [Rockart 81b] permettent de définir ces « *informations*

pertinentes, limitées aux domaines dans lesquels les résultats, s'ils sont satisfaisants, assureront la performance et la compétitivité de l'entreprise ». Les systèmes d'information de direction se présentent alors comme des systèmes d'aide à la décision destinés à des dirigeants de haut niveau, leur permettant de se concentrer sur ces facteurs critiques de succès.

Les systèmes d'information stratégique (SIS) se rapprochent des systèmes d'information de direction définis plus haut. Ce sont des systèmes manipulant l'information stratégique pour aider à la prise de décisions et à l'examen des stratégies mises en œuvre. Ces systèmes se présentent comme des systèmes d'aide à la décision permettant d'une part l'élaboration d'une stratégie (sous la forme d'un plan) et d'autre part son suivi. Nous renvoyons le lecteur intéressé aux travaux réalisés en stratégie concurrentielle par M. Porter [Porter 82 & 86], sur la théorie des coups stratégiques de C. Wiseman [Wiseman 88], ainsi que sur la théorie des plans d'action de R. Wilenski [Wilenski 83].

En ce qui concerne la conception des systèmes d'information stratégique (SIS ou *Executive Information Systems*), la méthode Merise n'est pas vraiment adaptée, tant au niveau des raisonnements que de la démarche. De nouvelles méthodes restent à élaborer, elles devraient, tout en retenant certains aspects de Merise, prendre de nouvelles racines d'une part dans l'art de la stratégie concurrentielle et d'autre part dans le génie cognitif. Une démarche qui se veut compatible avec Merise est proposée par Tardieu et Gutmann [Tardieu & Guthmann 91]. Si Merise a su concilier organisation et informatique, ces nouvelles méthodes devront réunir stratégie et informatique.

Stratégies de développement d'entreprise et système d'information

Dans un environnement de plus en plus concurrentiel, les entreprises doivent appliquer les meilleures techniques de gestion disponibles. Durant les années 70, la planification stratégique, technique de gestion, a été l'outil clé pour le succès de l'entreprise. Il était généralement acquis qu'une entreprise appliquant un plan stratégique efficace atteindrait d'excellents résultats. Au cours des années, on s'est aperçu qu'une stratégie judicieuse ne suffisait pas, à elle seule, à garantir le succès.

Le problème majeur résidait en ce que ces stratégies, aussi bien conçues fussent-elles, n'étaient pas correctement appliquées dans l'entreprise (inertie, résistance au changement...). Ces difficultés de mise en œuvre de stratégies peuvent être considérablement réduites en utilisant des outils, des techniques, systèmes de gestion s'appuyant sur des systèmes d'information « artificiels » spécifiques.

Pour examiner comment les systèmes d'information artificiels d'une entreprise doivent être conçus, il est donc nécessaire de savoir quels systèmes de gestion, et donc quels systèmes d'information naturels, sont adaptés à la stratégie de développement de l'entreprise.

Un des facteurs déterminants est le degré de maturité de l'entreprise. La plupart des entreprises tendent à suivre une évolution, plus ou moins rapide, en plusieurs étapes : stade de démarrage, puis stades de croissance et de maturité, et enfin stade de vieillissement, comme l'illustre la figure 4.9.

Les entreprises performantes en stade de démarrage se caractérisent par leur esprit d'entreprise, elles excellent en recherche-développement et dans la connaissance de leur marché [Olivier 84]. Elles mettent en œuvre des technologies nouvelles avant leurs concurrents. Leurs systèmes d'information sont plutôt informels.

Les entreprises leaders en phase de croissance ne sont pas toujours à la pointe des technologies, mais se caractérisent plutôt par la force de leur marketing et de leur organisation de vente. Leur planification reste assez flexible et leurs systèmes d'information sont plus formels.

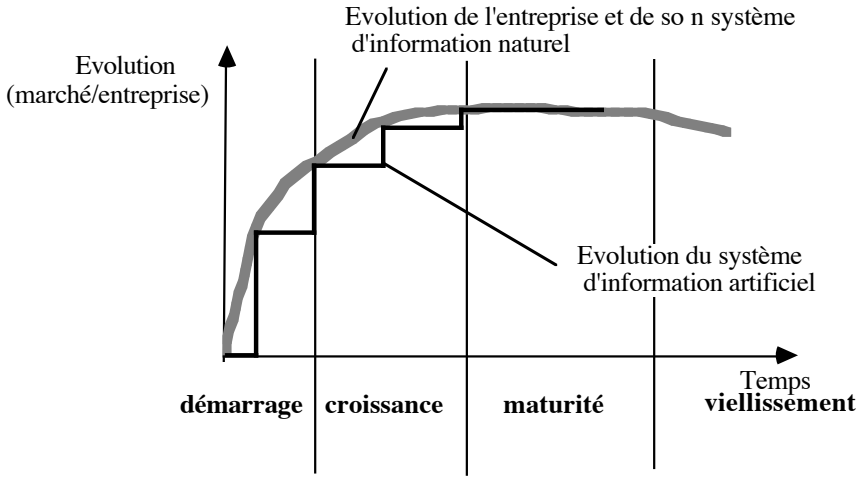


Figure 4.9 : Les stades de développement d'une entreprise.

Les entreprises en stade de maturité se caractérisent par la qualité de leur gestion opérationnelle, c'est le souci croissant du contrôle. Leurs systèmes d'information ont pour objectif principal d'assurer ce contrôle.

Enfin, la réussite des entreprises en phase de vieillissement dépend pour une large part d'une exploitation opportuniste et d'un contrôle financier étroit, les besoins de contrôle sont à leur maximum. Le degré de certitude de l'entreprise étant plus grand, la complexité de ses systèmes d'information va en se réduisant.

Les spécificités des systèmes d'information accompagnant ces quatre stades d'évolution de l'entreprise sont illustrées par la figure 4.10 [Olivier 84].

	Démarrage	**Croissance**	**Maturité**	**Vieillissement**
Objectif principal	rapidité, flexibilité	planification	coordination, contrôle	contrôle
Caractère	informel, ad-hoc, oral	plus formel, sur mesure, oral et écrit	formel, uniforme, écrit	formel, uniforme, réduit
Contenu	qualitatif, orienté vers le marché, peu systématique	qualitatif et quantitatif, couvrant toutes les fonctions, systématique	quantitatif, orienté vers les opérations, systématique	quantitatif, financier, systématique
Directives	peu nombreuses	plus nombreuses	nombreuses	nombreuses
Procédures	inexistantes	peu nombreuses	nombreuses	nombreuses

Figure 4.10 : Spécificités des systèmes d'information accompagnant l'évolution de l'entreprise.

Les systèmes d'information ouverts

De l'entreprise-système au système d'entreprises

Les *systèmes d'information ouverts ou globaux* constituent une nouvelle famille de systèmes d'information non plus associés à une seule entreprise, mais à plusieurs entreprises devenant partenaires. Ce passage de l'entreprise-système au système d'entreprises (voir la figure 4.11) sera associé à la définition de stratégies d'ouverture, de partenariat, de coopération-coalition de plusieurs entreprises, voire de secteurs économiques entiers.

Ainsi les entreprises productrices de biens ou de services sont confrontées à un marché global et à la prise en compte d'une production centrée client. Ce contexte de production se traduit par une demande de produits et de services de plus grande qualité, disponibles dans des délais minimaux, moins chers et personnalisés [Vernadat 94] [De Terssac 92].

Pour faire face à ces exigences, les entreprises adoptent de nouveaux types d'organisation leur permettant d'accroître leur flexibilité et leur réactivité ; citons notamment l'entreprise étendue (extend enterprise) [Browne 94], l'entreprise virtuelle (virtual enterprise) [Rolstadas 94], l'ingénierie concourante [Molina & al. 94], les systèmes holoniques de manufacture (holonic manufacturing systems) [Herath 94] [Deen 94] et enfin le D-CIM (Distributed-

CIM) [Skjellaug & al. 90].

Ces nouveaux types d'organisation présentent de nombreuses similitudes en mettant notamment l'accent sur deux points fondamentaux : (i) la distribution de responsabilités et de capacités décisionnelles de l'entreprise conduisant à la caractérisation d'unités autonomes internes ou externes à celle-ci et (ii) le développement de politiques de coopération entre ces unités. Ainsi, une entreprise, ses différents départements, ses sous-traitants, ses clients et ses fournisseurs peuvent être appréhendés comme un réseau dont les nœuds sont des "unités autonomes" qui coopèrent et constituent l'entreprise distribuée. La coopération entre ces unités autonomes nécessite la coordination de processus décisionnels eux-mêmes distribués.

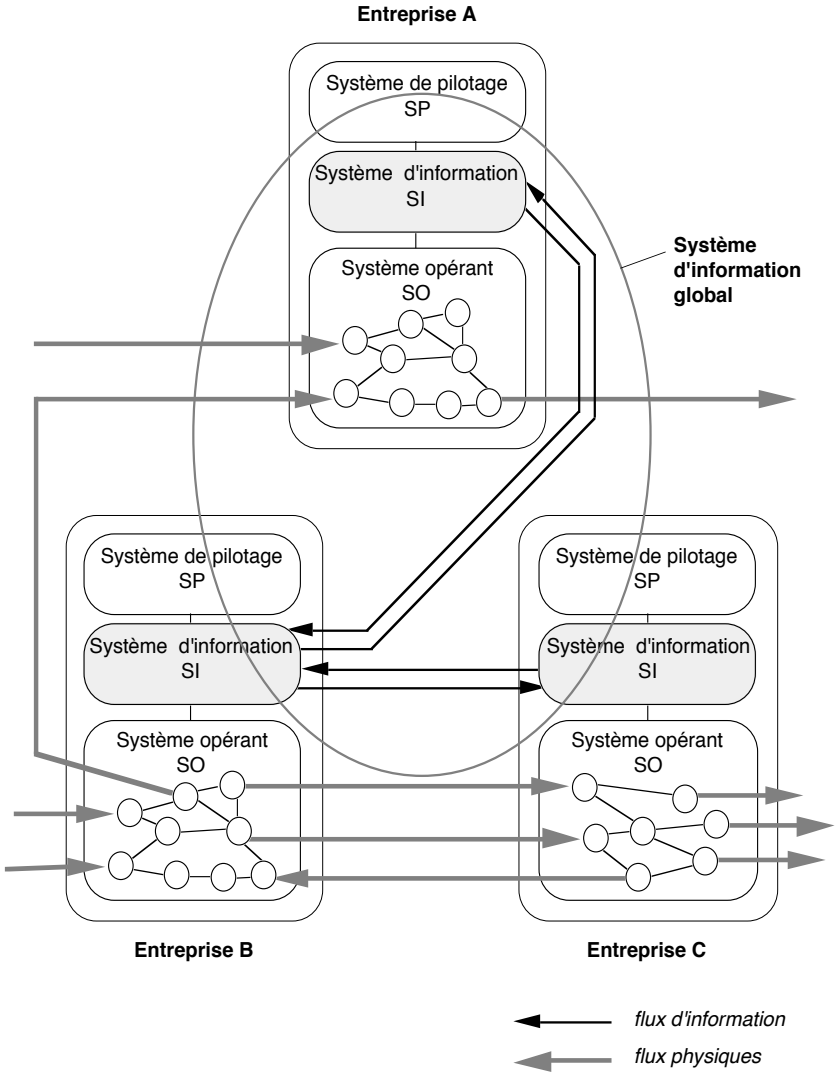


Figure 4.11 : De l'entreprise-système au système d'entreprises.

Les possibilités offertes par les nouvelles techniques de traitements de l'information et des connaissances, mais aussi et surtout le développement d'Internet et des intranets, apportent de nouveaux challenges, notamment celui de développer de nouveaux systèmes d'information permettant d'une part d'accéder à des données et connaissances multimédia de sources d'information localisées partout de par le monde et d'autre part de supporter la coordination d'activités organisationnelles humaines et/ou artificielles associées aux nouveaux types d'organisation en permanente évolution que nous venons d'évoquer.

Comme nous l'avons vu précédemment, les systèmes d'information stratégiques de l'entreprise avaient pour objectif clairement affirmé de faire gagner à l'entreprise des parts de marché en assistant le système de pilotage dans la planification et le suivi de ses stratégies. C'est ainsi la recherche de performance concurrentielle dans une certaine fermeture : la recherche du "competitive advantage" de l'entreprise-système.

Le développement de l'EDI (échanges de données informatisés) a ouvert cette recherche de performance concurrentielle et conduit les entreprises à raisonner maintenant non plus seulement en termes d'entreprise-système mais plutôt en termes de systèmes d'entreprise auxquels sont associés des systèmes d'information ouverts ou globaux. Cette ouverture associée à l'élaboration de partenariats est devenu dans la compétition mondiale une nécessité pour les entreprises (par exemple entre un centre de grande distribution et ses fournisseurs), elle caractérise le passage du "competitive advantage" au "competitive necessity".

L'EDI (échange de données informatisé)

L'EDI (échange de données informatisé) constitue un enjeu majeur de l'informatique stratégique dépassant le simple fait de s'échanger des données. Enjeu majeur qui se situe aussi bien au niveau de partenaires entreprises qu'au niveau d'une communauté de partenaires socio-économiques, voire par le biais de l'informatique au niveau d'un ou de plusieurs secteurs économiques.

En management, d'importants moyens humains et matériels sont mobilisés pour le montage d'un dossier commercial, le suivi d'une opération de logistique, l'enregistrement d'événements sur les plans comptable, statistique, fiscal et douanier. L'ensemble des coûts de traitements et d'échanges d'informations est estimé en moyenne à 7 % de la valeur des marchandises vendues. Ces coûts, répartis entre l'émetteur et le récepteur, peuvent être considérablement réduits par la mise en œuvre de l'EDI. L'EDI permet d'échanger des informations de système informatique à système informatique sans ressaisie.

Au niveau partenarial, un premier type d'avantages est lié à la dématérialisation

des documents, c'est-à-dire à la suppression de la chaîne traditionnelle du support papier. L'EDI présente aussi des avantages liés au management pratiqué par chacun des partenaires. Tout d'abord il apporte une amélioration de la qualité des transferts d'informations, sur le plan de la sécurité et de la fiabilité. Il peut conduire à un recouvrement des créances plus rapide permettant d'améliorer la trésorerie. L'EDI renforce les liens entre les partenaires pouvant conduire ainsi à une certaine fidélisation. Enfin, il permet de réconcilier les flux physiques et les flux d'information [Stoven 89], apportant ainsi une meilleure réponse à l'événement, un suivi logistique de bout en bout et la possibilité d'instaurer le « juste à temps » (JAT) [Sandoval 90a]. L'EDI conduit à une évolution en profondeur de l'entreprise. En effet, rien ne sert de recevoir des commandes en temps réel si la production et la logistique ne suivent pas. Il faudra à terme adapter voire reconcevoir le système d'information de l'entreprise et améliorer les fonctionnements internes. L'entreprise est un tout, dont il faudra repenser un certain nombre de fonctionnements afin de maximiser les potentialités de l'échange électronique. Ainsi, l'amélioration du processus de commande (commande client chez le fournisseur) par l'utilisation de l'EDI devra être resituée dans un contexte plus global, en repensant le fonctionnement des flux physiques associés.

Au niveau communautaire, l'enjeu de l'EDI est fondamental [C.C.E. 89, C.C.E 90, Stoven 89...]. Tout d'abord il peut consister en la fédération de tout un secteur d'activité de l'économie autour de quelques entreprises (EDI sectoriels) et conduire à améliorer de façon considérable les services, augmenter les gains, offrir une meilleure image, etc. Il peut aussi dépasser le cadre d'un secteur particulier de l'économie pour améliorer les communications entre plusieurs secteurs, en augmentant les communications en aval et en amont de l'entreprise. De façon générale, l'objectif principal est d'améliorer la productivité globale de la communauté en permettant une gestion communautaire se rapprochant des flux tendus (report des stocks chez les fournisseurs…). Enfin, l'EDI conduit aussi à des améliorations importantes dans les relations entre les entreprises et les pouvoirs publics.

Deuxième Partie

Les raisonnements de la méthode Merise : conception du Systèmes d'Information Organisationnel (SIO)

Partie 2
Principes généraux et fondements théoriques de la méthode Merise

Préambule

La première partie a présenté les principes généraux et les fondements théoriques de la méthode Merise ainsi que les types de systèmes d'information pour lesquels elle est plus particulièrement adaptée.

Cette deuxième partie traite des raisonnements mis en œuvre par le concepteur pour l'élaboration des modèles nécessaires à la compréhension et à la conception du système d'information organisationnel (SIO).

Elle précise comment élaborer et exprimer les différents modèles de données et de traitements de niveau conceptuel et organisationnel qui spécifient ce SIO. Cette partie se compose des chapitres suivants :

- chapitre 5 : découpage en domaines et analyse des flux ;
- chapitre 6 : modélisation conceptuelle des traitements ;
- chapitre 7 : modélisation conceptuelle des données ;
- chapitre 8 : modélisation organisationnelle des traitements ;
- chapitre 9 : cycle de vie des objets et objets métier ;
- chapitre 10 : modélisation organisationnelle des données ;
- chapitre 11 : confrontation données/traitements.

Notons que dans cette partie, la présentation des différents modèles associés au SIO concerne essentiellement la dimension du cycle d'abstraction de la méthode (les raisonnements). Cette présentation est faite sans référence aux autres dimensions de la méthode : les cycles de vie et de décision.

Une telle démarche est théorique et n'a de justification que pédagogique.

Dans la pratique, comme nous le verrons dans la quatrième partie de l'ouvrage, le parcours du cycle d'abstraction devra toujours se situer par rapport à une tape du cycle de vie, en particulier les études préalables, détaillée ou technique.

5

Découpage en domaines et analyse des flux

Découpage en domaines

L'analyse systémique nous a fourni une modélisation de l'entreprise échangeant et transformant des flux (cf. chapitre 2). L'approche par les niveaux de complexité nous a montré que l'entreprise-système, dès le niveau 7, résultait d'une fédération de processeurs actifs élémentaires coordonnés: le système opérant. Le système d'information est la représentation de l'activité de ce système opérant.

Cette vision unitaire d'un système d'information général, même si elle traduit justement la complexité et les interdépendances internes d'un tel système, est difficilement exploitable; il faudrait aborder l'informatisation du système d'information d'une entreprise d'un seul tenant...

Pour tenter de réduire la complexité de modélisation d'une entreprise, et surtout pour obtenir des tailles de projet maîtrisables, on cherche à découper l'entreprise *en domaines d'activité* (voir figure 5.1). Ce découpage s'effectue généralement sur la base des grandes fonctions ou activités de cet organisme : vendre, stocker, acheter, gérer du personnel, etc.

Chaque domaine est considéré comme « quasi autonome » avec son propre système opérant, son propre système de pilotage et son propre système d'information. Le système d'information général de l'entreprise n'est alors défini que comme la réunion des systèmes d'information de chaque domaine.

Ce découpage s'opère généralement au regard des activités du système opérant et/ou des finalités du système de pilotage. Cependant, les systèmes d'information résultant du découpage en domaines ne sont pas disjoints. Comme ils entretiennent entre eux des flux, et qu'ils partagent des perceptions sur l'environnement, certaines représentations figurent dans plusieurs systèmes d'information. Il faudra alors être vigilant à la cohérence interdomaine.

Ce délicat problème de découpage en domaines, apparaissant comme pré-

requis à l'informatisation d'un système d'information, est normalement abordé, dans la démarche de la méthode, lors du schéma directeur. Toutefois, en pratique, un tel schéma est souvent absent, ou ne comporte pas de découpage en domaines.

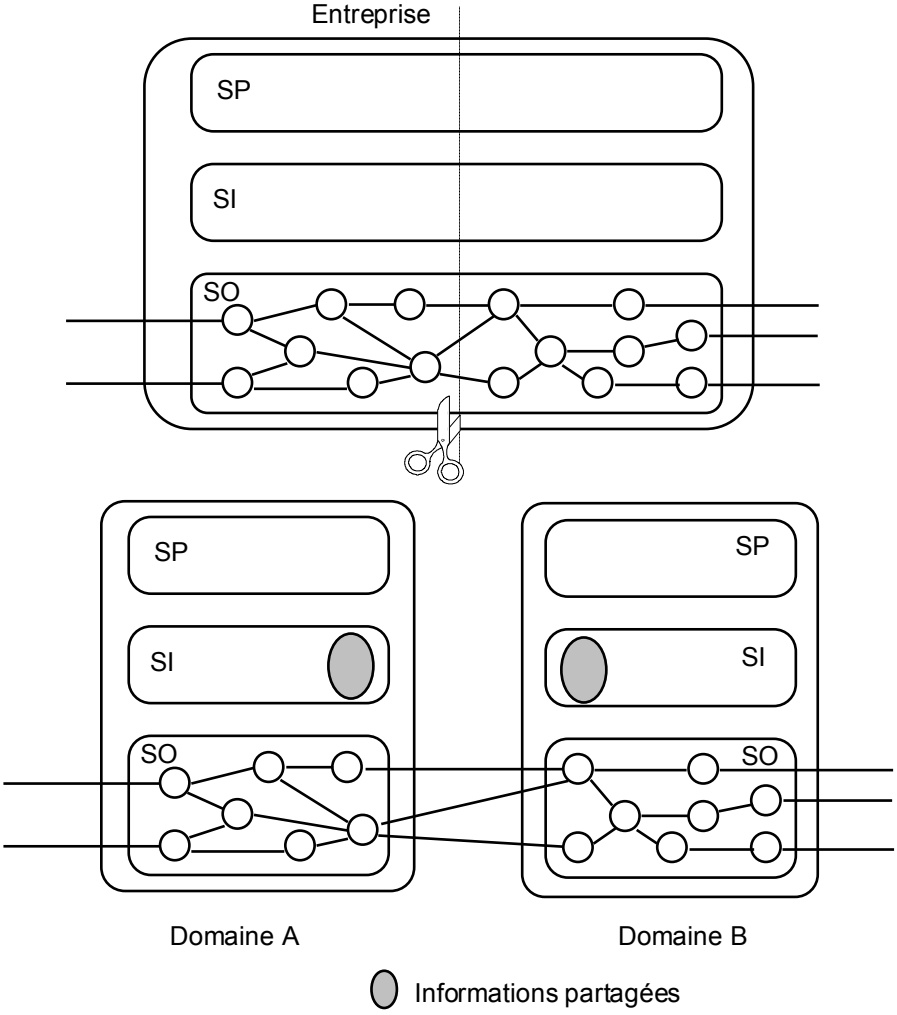


Figure 5.1 : Le découpage en domaines.

En conséquence, le concepteur doit, dans la première phase de l'étude préalable, procéder à la délimitation du domaine dont il va élaborer le système d'information informatisé. Pour ce faire, il peut utiliser la technique d'*analyse des flux*.

Analyse des flux

L'approche systémique évoquée précédemment nous suggère d'analyser le

système opérant de l'entreprise comme un ensemble coordonné d'« unités actives» échangeant des flux entre elles, avec le système de pilotage et l'environnement. L'analyse des flux permet d'appréhender simplement le fonctionnement global de l'entreprise, en se focalisant éventuellement sur un ensemble d'activités concernées par l'étude, sans chercher à identifier l'origine et la stabilité de ce découpage en unités actives; la prise en compte des niveaux d'abstraction (conceptuel, organisationnel, logique, physique) s'effectuera dans les autres modèles.

Acteurs et flux

L'analyse des flux s'exprime avec deux concepts : l'acteur et le flux.

L'acteur

L'*acteur* représente une unité active intervenant dans le fonctionnement du système opérant. Stimulé par des flux, il les transforme, les renvoie ; un acteur « fait quelque chose », il est actif.

Dans la pratique, un acteur peut modéliser :

- un partenaire extérieur à l'entreprise (client, fournisseur...);
- un domaine d'activité de l'entreprise précédemment identifié (la comptabilité, la gestion du personnel...);
- un ensemble d'activités ou processus (liquidation, contrôle...);
- un élément structurel de l'entreprise (service, unité géographique, unité fonctionnelle...);
- le système de pilotage, ou pilote, dans ses interactions avec le système opérant ou le système d'information.

Le flux

Le flux représente un échange entre deux acteurs. Les flux peuvent être classés en cinq catégories :

- Matière (qui est transformée ou consommée)
- Finance
- Personnel
- Actif (matériel ou savoir-faire nécessaire pour exercer l'activité)
- Information

Un flux est émis par un acteur à destination d'un autre acteur.

Dans l'utilisation de l'analyse des flux par la méthode Merise, on s'intéressera principalement aux flux d'informations. Les autres types de flux qui présenteraient un intérêt majeur devront donc être explicités en flux d'informations (information-représentation).

Dans l'utilisation de l'analyse des flux par la méthode Merise, conformément à l'analyse séparée données/traitements, nous ne représentons pas les flux entre un acteur et la mémoire ; un fichier n'est pas modélisé comme un acteur, bien qu'il subisse des échanges d'informations. Si l'on souhaite faire apparaître de tels échanges, nous suggérons de faire appel à un troisième type de concept qui s'apparenterait au « dépôt de données » des formalismes Diagrammes de flux de données de l'analyse structurée [Gane & Sarson 79].

Diagramme des flux

C'est une représentation graphique (une « cartographie ») des acteurs et des flux échangés.

En l'absence de norme d'usage de symbolisation, les acteurs peuvent parfois être représentés par différents symboles selon leur nature:: partenaire extérieur, domaine, processus, unité organisationnelle, ...

Le flux est représenté par un lien orienté (fléché); le nom du flux étant porté par ce lien. Un diagramme des flux, s'inspirant d'un thème sur l'assurance automobile et que nous retrouverons tout au long des différentes modélisations, est présenté figure 5.2

L'aspect visuel et la simplicité du symbolisme font du diagramme des flux un support efficace pour le dialogue avec l'utilisateur, en particulier lors des premiers entretiens.

Pour préserver la lisibilité de tels diagrammes et éviter de trop nombreux croisements, on peut « dupliquer » les symboles de certains acteurs.

Le diagramme des flux peut parfois, dans la phase de l'analyse de l'existant (cf. partie IV), se substituer au modèle organisationnel des traitements actuel dans le cas où les aspects organisationnels sont simples ou limités.

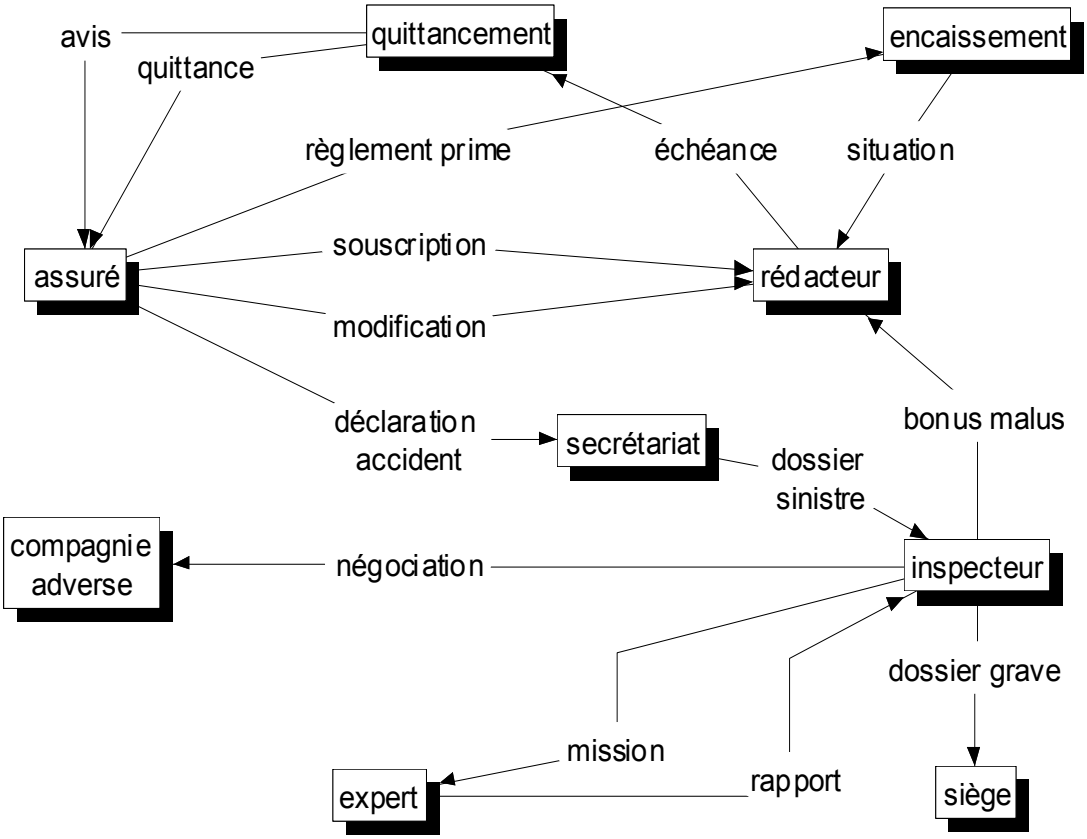


Figure 5.2 : Diagramme des flux.

Matrice des flux

C'est une représentation matricielle des acteurs et des flux échangés.

Les acteurs forment les lignes et colonnes du tableau. Situé en ligne, l'acteur a un rôle d'émetteur de flux ; situé en colonne, il a un rôle de destinataire de flux.

Les flux sont indiqués dans les « cases » du tableau de la figure 5.3, à l'intersection de la ligne de l'acteur-émetteur et de la colonne de l'acteur-destinataire.

La matrice des flux est plutôt destinée à l'usage du concepteur pour recenser les flux d'une manière systématique, en s'interrogeant à chaque case.

vers de	assuré	cie adverse	Quittancement	encaissement	rédacteur	secrétariat	inspecteur	expert	siège
assuré				règlement prime	souscription modification	déclaration			
cie adverse									
quittancement	quittance avis								
encaissement					situation				
rédacteur			échéance						
secrétariat							dossier		
inspecteur		négociation						mission	dossier grave
expert							rapport		
siège									

Figure 5.3 : Matrice des flux

Utilisation de l'analyse des flux pour le découpage en domaines

Diagramme brut des flux

Ce diagramme est élaboré lors des premières interviews; le concepteur découvre le champ de l'étude et va utiliser l'analyse des flux pour débroussailler le domaine. La notion d'acteur s'applique à toute unité active ; le diagramme des flux ressemble à celui de la figure 5.2.

La visualisation des acteurs et des flux permet alors au concepteur de regrouper les différentes unités actives qui constitueront le domaine d'activité étudié. On met ainsi en évidence deux frontières qui permettent de répartir les acteurs :

- l'entreprise,
- le domaine étudié.

On peut également percevoir certains acteurs comme l'expression d'autres domaines par ailleurs définis, ou regrouper des acteurs extérieurs au domaine étudié qui peuvent être considérés comme des domaines potentiels.

On peut faire figurer ces frontières et regroupements sur un diagramme des

flux (voir figure 5.4) ; on s'efforcera alors de mettre en évidence le domaine étudié.

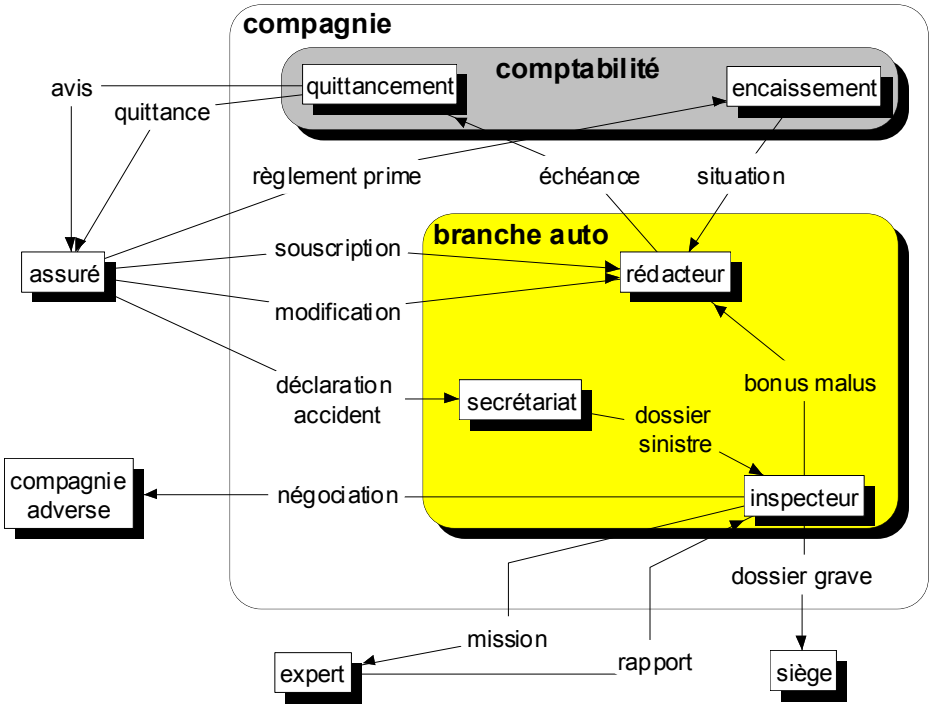


Figure 5.4 : Diagramme des flux avec frontière

Acteurs internes, acteurs externes, domaines

L'établissement de la frontière du domaine induit une distinction entre les acteurs internes et les acteurs externes au domaine étudié.

Les *acteurs internes* traduisent fréquemment une répartition des activités, au sein du domaine, selon des choix d'organisation. Ultérieurement, nous verrons que l'abstraction nécessaire à la modélisation conceptuelle des traitements nous conduira à ne plus prendre en compte ces acteurs internes. Ils réapparaîtront éventuellement lors de la modélisation organisationnelle des traitements sous la forme de postes. En résumé, les acteurs internes n'ont qu'une existence éphémère et limitée à l'analyse des flux.

Les *acteurs externes* conservent une importance capitale dans l'étude et la modélisation du système d'information. Le domaine (vu comme système ouvert) ne « vivra » qu'avec ses échanges avec les acteurs externes. Les flux qu'ils émettent vont être des stimuli déclenchant l'activité du domaine; les flux

qu'ils reçoivent sont les réponses des activités du domaine. Ces flux entre les acteurs externes et le domaine sont un des éléments stables que nous recherchons pour la conception d'un système d'information.

Certains de ces acteurs externes représentent souvent d'autres domaines de l'entreprise, l'étude engagée étant rarement la première pour l'informatisation des systèmes d'information. Si ces domaines sont explicitement déterminés, il est préférable de les identifier et de les représenter en tant que tels sur le diagramme des flux. En effet, les flux inter-domaines mis ainsi en évidence contribuent très tôt au maintien de la cohérence inter-domaine et à la détermination des futures interfaces.

Diagramme conceptuel des flux

A la différence du diagramme brut des flux présenté précédemment, le diagramme conceptuel des flux (voir figure 5.5) tient compte des conséquences de la détermination du domaine. N'apparaîtront sur un tel diagramme que :

- le domaine étudié,
- les acteurs externes,
- les autres domaines.

Un tel diagramme peut ainsi avoir une existence plus permanente que le diagramme brut des flux, s'enrichir régulièrement et représenter l'articulation générale des systèmes d'information. En conséquence, au fur et à mesure de l'étude des systèmes d'information de l'entreprise, les acteurs externes ne représenteront plus que des acteurs extérieurs à l'entreprise.

Un tel diagramme peut ainsi avoir une existence plus permanente que le diagramme brut des flux, s'enrichir régulièrement et représenter l'articulation générale des systèmes d'information. En conséquence, au fur et à mesure de l'étude des systèmes d'information de l'entreprise, les acteurs externes ne représenteront plus que des acteurs extérieurs à l'entreprise.

Notons que ce diagramme des flux conceptuel est également désigné comme diagramme des flux contextuel du domaine et qu'il correspond aussi globalement au modèle conceptuel de communication proposé dans certaines versions de la. méthode Merise.

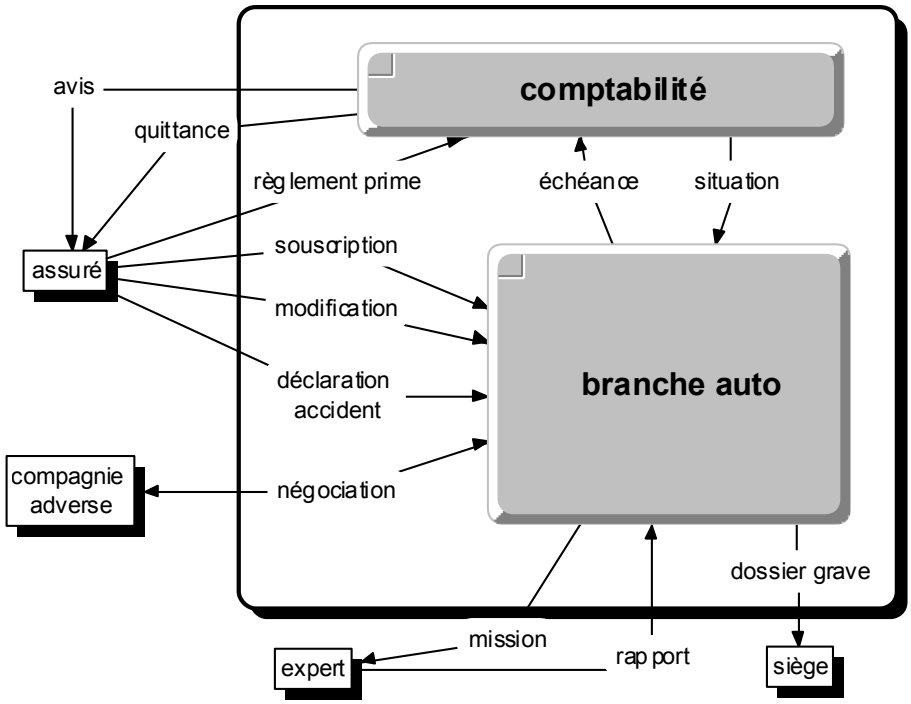


Figure 5.5 : Diagramme conceptuel des flux

6

Modélisation conceptuelle des traitements

Considérations méthodologiques

Notions de traitement dans Merise

Il importe d'abord de lever une ambiguïté sur la compréhension du terme traitement. Qu'il provienne des premières approches de l'informatique ou de l'informatique scientifique et technique, le terme *traitement* a été souvent limité à la seule transformation de données (voir figure 6.1).

Décrire le traitement revient alors à décrire l'algorithme (organigramme, arbre programmatique...)

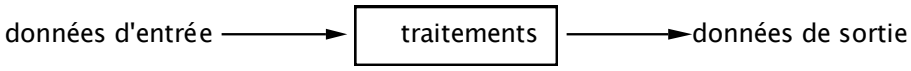


Figure 6.1 : Le traitement informatique.

Dans Merise, le terme traitement est plus général; il s'assimile au fonctionnement du système d'information perçu à travers ses couplages avec le système opérant et le système de pilotage. Décrire les traitements, c'est décrire les processus mis en oeuvre dans le domaine (vu comme un système) en interaction avec son environnement (voir figure 6.2).

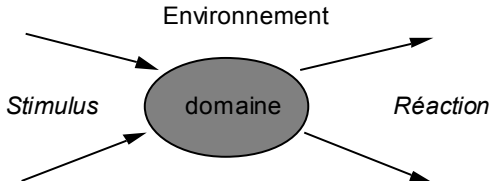


Figure 6.2 : Les traitements dans Merise.

Problématique de la modélisation conceptuelle des traitements

La modélisation conceptuelle des traitements a pour objectif de représenter formellement les activités exercées par le domaine, activités dont la connaissance est la base du système d'information. Elle est tournée vers la prise en compte des échanges du domaine avec son environnement (autres domaines, extérieur de l'entreprise, système de pilotage). C'est avant tout l'identification de ces échanges et des activités induites qui va contraindre et structurer le fonctionnement du domaine.

La modélisation de ces activités s'effectue en faisant abstraction de l'organisation, c'est-à-dire des moyens et ressources nécessaires à l'exécution de ces activités. Un modèle conceptuel de traitements (MCT) exprime ce que fait le domaine, et non par qui, quand, où et comment ces activités sont réalisées.

Dans ses principes généraux, la méthode Merise lie les niveaux de préoccupation aux degrés de stabilité. Au niveau conceptuel des traitements, cette stabilité se traduit par les flux échangés et les activités associées; la définition des interactions du domaine avec son environnement prime sur la manière dont s'exerceront ces activités.

L'élaboration d'un MCT permet ainsi de préciser les frontières du domaine en décrivant les activités qui lui sont associées et les échanges avec son environnement. Son utilisation dans la démarche viendra d'ailleurs le confirmer.

Formalisme de modélisation des traitements

La modélisation des traitements dans la méthode Merise s'exprime dans un formalisme spécifique [Heckenroth, Tardieu, Espinasse 80] [Tardieu, Rochfeld, Coletti 83], élaboré pour permettre de représenter le fonctionnement d'activités aux différents niveaux de préoccupations (conceptuel, organisationnel, logique, physique). Il conserve ainsi une unicité de structure qui évite la multiplication des formalismes. L'adaptation aux différents niveaux se fait par la dénomination des concepts types. Ce formalisme propose une représentation graphique destinée à faciliter le dialogue entre concepteur et utilisateur.

Il repose sur des bases théoriques solides permettant une vérification formelle des modèles, dans la mesure où il s'inspire du formalisme des réseaux de Pétri [Brams 83] [Techniques et sciences informatiques 85]. Il permet une simulation de l'activité du système d'information : fonctionnement pas à pas, mise en évidence de conflits et parallélismes.

Pour décrire le niveau conceptuel, le formalisme des traitements comporte les concepts suivants :

- L'acteur
- L'événement/résultat-message

- L'état
- L'opération

L'acteur

Comme nous l'avons déjà vu plus haut, le modèle conceptuel de traitements (MCT) formalise d'abord les activités du domaine consécutives aux échanges avec l'environnement. Les acteurs pris en compte dans un MCT sont donc uniquement les acteurs externes au domaine (à l'exception du système de pilotage). Les acteurs internes au domaine mis en évidence dans l'analyse des flux traduisent un découpage organisationnel dont on doit faire abstraction au niveau conceptuel.

L'acteur est formalisé graphiquement de la façon suivante :

L'événement/résultat-message

Les flux reçus (stimuli) et émis (réactions) par le domaine sont respectivement modélisés en événements et résultats (voir figure 6.3). Un événement est la formalisation d'un stimulus par lequel le domaine, puis son système d'information, prend connaissance de comportements de son environnement (interne ou externe à l'entreprise). Un événement est donc émis par un acteur à destination du domaine. Un résultat est la formalisation d'une réaction du domaine et de son système d'information. Un résultat est donc émis par une activité du domaine à destination d'un acteur.

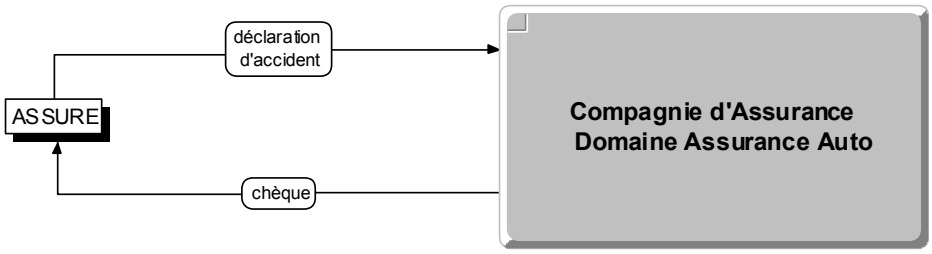


Figure 6.3 : Evénement et résultat.

On distingue plusieurs catégories d'événements/résultats :

- *Externes*, modélisant des flux avec un acteur.
- *Décisionnel*, représentant les échanges avec le système de pilotage.
- *Temporel*, représentant des échéances.

Ces deux dernières catégories seront plus particulièrement utilisées dans la modélisation organisationnelle des traitements (Chapitre 8)

Seuls les événements et résultats externes traduisent des échanges avec l'environnement du domaine et constituent de "vrais" événements et résultats.

Ces événements et résultats, pour être acceptés par le système, sont regroupés, modélisés en *types*. Par exemple, la déclaration de M. Dupont du 13/01 et la déclaration de M. Martin du 15/02 peuvent être considérées comme 2 *occurrences* de l'événement type déclaration d'accident. De même trois chèques émis par la compagnie d'assurance à destination d'assurés constituent 3 occurrences d'un même résultat type (figure 6.4).

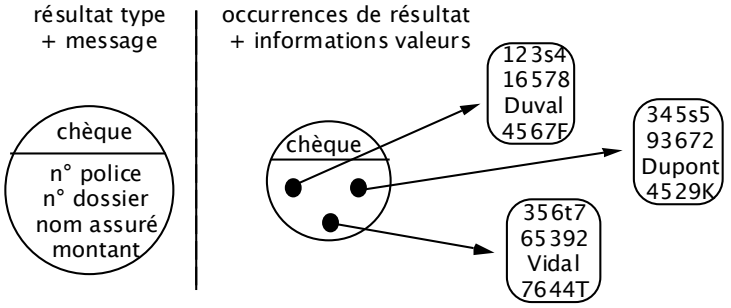


Figure 6.4 : Type, occurrence et message d'un événement/résultat.

Dans un MCT, on ne représente que des types d'événements et de résultats.

A un événement ou résultat est éventuellement associé un ensemble d'informations appelé *message*. Un message est un ensemble structuré d'informations décrivant un événement/résultat type (figure 6.4). Une occurrence d'événement/résultat doit être distinguable des autres par le contenu de son message associé, ainsi que par l'instant et l'endroit où il se produit.

Il faut parfois faire la distinction entre le message et son support. Par exemple, un dossier qui s'enrichit au fur et à mesure des activités représente différents messages successifs sur un même support.

L'état

Cette notion a été introduite dans la modélisation des traitements avec la deuxième génération. L'état modélise une situation du système d'information.

Dans la description du fonctionnement des activités d'un domaine, le concepteur peut constater que l'exécution de certaines activités d'une part dépend d'une situation préalable du système d'information (un dossier doit être ouvert avant d'instruire le sinistre), d'autre part peut produire des changements

d'état (après le règlement du sinistre, le dossier est clos).

L'état peut s'exprimer par:

- une valeur prise par une information (statut dossier = en cours),
- le fait qu'une activité à été réalisée (calcul des pénalités effectué),
- une règle de traitement (délai de règlement dépassé de 15 j.)

Ces situations ou états du système d'information qui conditionnent le fonctionnement des activités doivent être mémorisés. On peut envisager trois solutions :

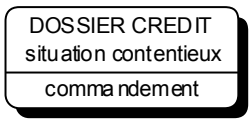
- Le fonctionnement du système d'information est sous la surveillance d'un "moniteur", véritable système d'information sur le fonctionnement du système d'information qui prend acte des activités réalisées et des états obtenus, et autorise ou interdit certaines activités conditionnées par des états.
- La mémorisation des états est prise en charge dans les traitements.
- La mémorisation des états est traduite dans les données par des informations spécifiques (par exemple, statut du dossier).

Compte tenu des possibilités actuelles d'implémentation technique, la mémorisation des états relève généralement de la troisième solution. En conséquence, la notion d'état s'applique essentiellement à des objets et associations entre objets modélisés dans les données. La prise en compte des états dans la modélisation des traitements établi ainsi un lien avec la modélisation des données.

Pour la description d'un état d'un objet de données, on précisera:

- le nom de l'objet,
- le nom de l'information décrivant le type d'état,
- la valeur de l'état,
- éventuellement la règle permettant de déterminer l'état.

L'état est formalisé graphiquement de la façon suivante :



Dans la modélisation conceptuelle des traitements, les états ont un rôle vis à vis des activités assez proche de celui des événements et résultats. Comme un événement, un état est une condition préalable à l'exécution d'une opération; comme un résultat, un état est la conséquence conditionnelle d'une opération. Cela explique pourquoi, dans le passé, les états ont été modélisés comme des

événements, qualifiés *d'internes*.

L'opération

L'opération est la description du comportement du domaine et de son système d'information par rapport aux événements types. Elle est déclenchée par la survenance d'un (ou plusieurs) événement(s) et/ou d'un (ou plusieurs) états synchronisés. L'opération comprend l'ensemble des activités que le domaine peut effectuer à partir des informations fournies par l'événement, et de celles déjà connues dans la mémoire du système d'information.

La segmentation en plusieurs opérations ne se justifie que par l'attente d'informations complémentaires en provenance d'événements nécessaires à la poursuite des activités.

Synchronisation

La synchronisation représente une condition de présence d'évènements et/ou d'états préalables au démarrage de l'opération..

Elle se traduit par une expression logique s'appliquant sur la présence (ou l'absence) des occurrences des événements et/ou des états. L'expression logique de la synchronisation utilise les opérateurs classiques ET, OU, NON, ou toute combinaison admise par la logique.

Si la condition est vérifiée, l'opération peut démarrer et les occurrences déclencheuses (ainsi que les messages associés) sont considérées comme consommées par l'opération. Si la condition n'est pas vérifiée, la synchronisation et les occurrences d'événements présents restent en attente jusqu'à ce qu'elle soit vérifiée.

Description d'une opération

L'opération est décrite par un ensemble d'activités ou fonctions élémentaires à assurer et peuvent comporter :

- des décisions,
- des règles de gestion,
- des actions sur les données mémorisées,
- des traitements sur les données,
- des actions quelconques.

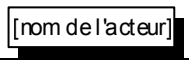
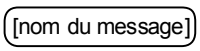
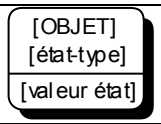
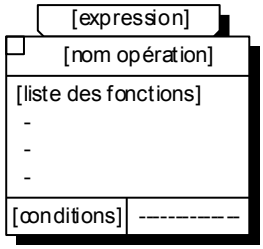
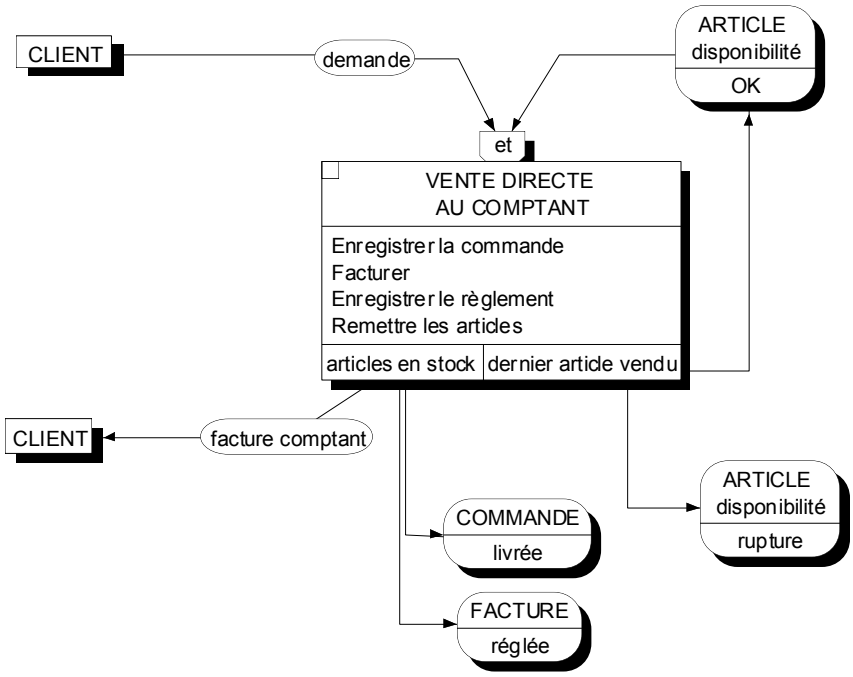
Acteur	
Evénement/Résultat	
Etat	
Opération	

Figure 6.5 a : Graphisme du formalisme des traitements de Merise.

Certains auteurs désignent le contenu d'une opération par tâches. Nous réserverons l'appellation tâche au niveau organisationnel, même si les fonctions décrites deviennent éventuellement des tâches dans le modèle organisationnel des traitements (MOT). Au niveau normal de modélisation des opérations d'un MCT, il n'est pas nécessaire de représenter explicitement les actions sur les données mémorisées et les règles de traitement ou de gestion. Cependant, dans un MCT *analytique* (voir « Modularité des modèles conceptuels de traitement »), les actions sur les données seront explicitement formalisées par la notion de sous-schéma, ainsi que les règles de traitement. Ces concepts sont présentés dans le chapitre «Modélisation organisationnelle des traitements»



6.5 b : Illustration d'une opération

L'ordre dans lequel les fonctions sont présentées au sein de l'opération n'est pas significatif; ces fonctions expriment l'ensemble des activités réalisables à partir de la survenance de l'événement. Le passage au MOT permettra ultérieurement d'organiser ces activités en termes de tâches.

Conditions d'émission

L'opération produit des résultats et/ou des états. L'émission de ces résultats et/ou états est soumise à des conditions traduites par des expressions logiques.

Bien que représentée graphiquement dans la partie inférieure de l'opération, une condition peut être vérifiée à partir de toute fonction de l'opération. La présence d'une condition (un test) dans le déroulement d'activités consécutives à un ou plusieurs événements ne justifie pas, au niveau conceptuel, la segmentation en différentes opérations.

Plusieurs résultats de nature et destination différentes, ainsi que plusieurs états d'objets ou d'états-types différents peuvent être émis par une même condition.

Le processus

Le processus est un ensemble structuré d'événements, opérations et résultats consécutifs qui concourent à un même but. Il représente généralement un sous-ensemble d'activités de l'entreprise dont les événements initiaux et les résultats

finaux délimitent un état stable du domaine.

Par exemple, dans le domaine assurance auto, on pourrait distinguer 3 processus :

- La prospection
- La gestion des contrats
- La gestion des sinistres

L'analyse en termes de processus est une vision macroscopique des activités du domaine que l'on pourra modéliser spécifiquement, comme nous le verrons plus loin (voir « Modularité des modèles conceptuels de traitement »).

Notions complémentaires pour la formalisation conceptuelle des traitements

Les concepts de base présentés suffisent généralement pour construire un modèle conceptuel de traitements. Toutefois, certaines situations à modéliser rendent nécessaires des éléments complémentaires tels que :

- *La durée de l'opération,* temps passé entre le déclenchement de l'opération et la production de résultat. Cette durée peut être variable suivant les conditions d'émission des résultats. Rarement utilisée.
- *La duplication d'un résultat*, nombre d'occurrences identiques d'un résultat émis par une opération. Par défaut, cette valeur est 1.
- *La participation d'un événement à une synchronisation*, nombre d'occurrences différentes de l'événement nécessaires au déclenchement de la synchronisation. Par défaut cette valeur est 1, une valeur typique est tous.

Un exemple

La figure 6.6 montre un exemple de modélisation conceptuelle de traitements représentant le processus de gestion d'une déclaration amiable d'accident.

On remarquera la double modélisation dans le cas d'un accident considéré comme trop grave :

La transmission du dossier grave au siège modélisé comme un message (flux),

L'indication dans le système d'information que le dossier a été transmis, modélisé comme un état.

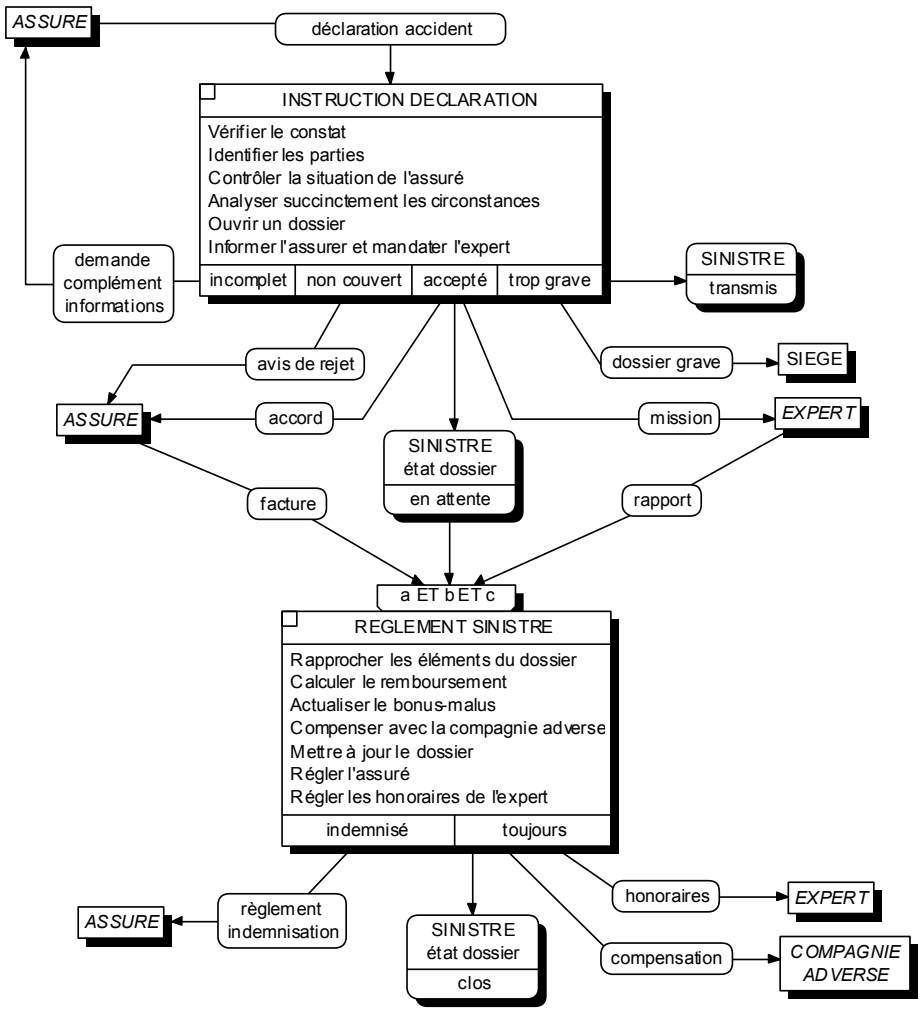


Figure 6.6 : Exemple de modélisation conceptuelle de traitements.

Règles de vérification

Comme tout modèle, un MCT doit représenter fidèlement le système étudié, actuel ou futur :

dans sa description (choix des acteurs, des opérations, des événements-résultats) ;

dans son fonctionnement (fonctionnement répétitif sans blocage, dans les différents cas de figures possibles, bonne circulation des occurrences d'événements et de résultats).

La description, tout à fait essentielle, sera validée par les gestionnaires et les futurs utilisateurs du domaine, au cours de présentations prévues à cet effet

Le fonctionnement correct du MCT peut être apprécié par une simulation manuelle.

Description et fonctionnement sont plus faciles à vérifier, lorsque le MCT respecte quelques règles simples de syntaxe.

Règles de syntaxe

a) Un acteur émet au moins un événement, ou reçoit au moins un résultat.

b) Un événement externe provient d'au moins un acteur.

c) Un résultat provient d'au moins une opération.

d) Tout résultat a au moins une destination : acteur ou opération.

e) Une opération est déclenchée soit directement par un événement ou un état, soit par une synchronisation unique.

f) Une synchronisation lie au moins deux événements ou états par une expression logique.

g) Une expression logique associée à une synchronisation ou à l'émission d'un résultat ne peut être toujours fausse ; il doit y avoir au moins une situation où cette expression logique est vraie, sinon l'opération ne sera jamais déclenchée, ou le résultat jamais émis.

Les règles a, b et e imposent la fermeture du modèle sur les acteurs :

- Le MCT ne vit que par ses échanges avec l'environnement.
- Les événements externes ne naissent pas spontanément.
- Les résultats produits sont utilisés.

Règles de fonctionnement

Nous retiendrons ici trois règles simples :

Un fonctionnement cyclique doit pouvoir être contrôlé.

Il y a cycle dans un MCT lorsqu'un événement ou un état contribue à une synchronisation-opération qui produit, directement ou à travers plusieurs opérations, ce même événement ou état.

Pour modéliser un fonctionnement répétitif, un MCT peut comporter des cycles. Il faut alors s'assurer que chaque cycle est contrôlé, en précisant clairement les conditions de son démarrage et de son arrêt.

Tout résultat ou état du MCT doit pouvoir être produit (résultat atteignable).

Un résultat ou un état est dit atteignable si l'on peut trouver une séquence d'activation de synchronisations et de conditions d'émission qui permettent de produire ce résultat ou cet état.

Pour un résultat ou un état déterminé, son atteignabilité dépend de plusieurs conditions :

- Existe-t-il dans le schéma représentant le modèle un chemin entre les événements et/ou états initiateurs du processus et ce résultat ?
- Les conditions de déclenchement des synchronisations et les conditions d'émission des résultats et/ou états présents sur ce chemin sont-ils compatibles ?

Les situations de conflit doivent être analysées

Il y a situation de conflit sur un événement/résultat s'il contribue à plusieurs synchronisations ou s'il est destiné à plusieurs acteurs.

Le conflit peut être résolu si les conditions de participation aux synchronisations sont exclusives, en ajustant la duplication du résultat, ou par une décision explicite du pilote.

Le fonctionnement devient alors déterministe.

Si un conflit n'est pas résolu, le fonctionnement du MCT n'est pas prévisible ; il est dit non déterministe.

La figure 6.7 illustre cette résolution de conflits.

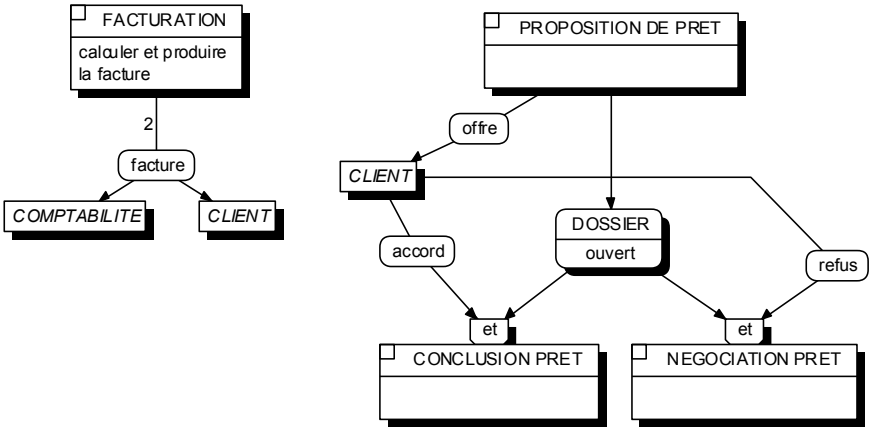


Figure 6.7 : Exemple de conflits résolus

Construction d'un modèle conceptuel des traitements

Rappelons que le concepteur procédera deux fois à l'élaboration d'un MCT : lors de l'étude préalable, puis lors de l'étude détaillée. Les résultats produits seront certes différents dans leur approche et dans leur ampleur ; cependant, l'on retrouvera des principes généraux de construction que nous résumons ci-dessous :

Recenser les acteurs et les flux échangés. L'analyse des flux (voir chapitre 5) et sa représentation par le diagramme des flux, en particulier sous sa forme de diagramme des flux conceptuels, permet de mettre en évidence le domaine, les acteurs et les flux échangés. Un effort d'abstraction sera fait pour identifier ces échanges par des événements/résultats.

Identifier les principaux processus, au sein du domaine, liés aux flux précédents.

Découper chaque processus en opérations, c'est-à-dire en une succession d'événements et de résultats. Ce découpage suscite quelques conseils :

- On regroupera dans une même opération toutes les activités qui peuvent être effectuées, dès la survenance de l'événement, sans tenir compte des éventuelles attentes qui ne seraient dues qu'à l'organisation interne. En conséquence, deux opérations consécutives, s'enchaînant directement ou uniquement par un état, ne présentent aucune attente et devraient de ce fait être fusionnées.
- Au niveau conceptuel, l'on ne cherche pas à expliciter l'enchaînement des fonctions élémentaires de l'opération, ni les moyens nécessaires à leur exécution (qui sont supposés illimités et immédiatement disponibles). Leur présentation se fait fréquemment sous la forme d'une liste. Il suffit de décrire ce que fait l'opération.
- Dans la description d'un processus, seule l'attente d'événement complémentaire devrait justifier le découpage en plusieurs opérations. Quand une opération s'achève, le domaine perd le contrôle de la poursuite du processus.
- A chaque survenance d'événement, rien n'oblige à ce que toutes les fonctions de l'opération soient à effectuer. Une condition peut se trouver vérifiée dès les premières fonctions d'une opération et conduire à la fin de l'opération.
- L'ensemble des conditions de sortie d'une opération n'est pas obligatoirement dichotomique; leur expression peut être considérée comme vraie ou fausse à n'importe quelle étape du déroulement de l'opération et plusieurs peuvent avoir la valeur « vraie » à l'issue d'une opération.
- Plusieurs résultats peuvent être émis par la même condition de sortie.

- Il n'est pas obligatoire de représenter comme consécutives (ou liées) des opérations dont l'état résultant de l'une est l'état préalable de l'autre

Modélisation orientée processus ou orientée état

L'introduction de la notion d'état dans la modélisation conceptuelle des traitements que nous avons vu se substituer au rôle d'événement interne entraîne une importante évolution dans la technique, l'approche et la représentation des MCT.

Dans une approche orientée processus, l'analyse et la construction du MCT sont conduites par l'enchaînement chronologique des activités, à partir d'un ou plusieurs événements initiaux. On déroule ainsi le processus, en respectant toutefois les règles de modélisation des opérations (on regroupe dans une opération toutes les activités réalisables à partir des informations fournies par le ou les événements sans avoir besoin d'autres informations nécessitant des événements complémentaires).

La description des processus est alors un enchaînement d'opérations, cet enchaînement étant assuré par les états; la similitude graphique (voulue) entre événement interne et état permet de conserver l'aspect général et la présentation des MCT de la première génération. Dans cette modélisation, une même opération peut être représentée dans plusieurs schémas de processus. Le MCT présenté en figure 6.6 est une illustration de cette approche orientée processus.

Dans une approche orientée état, chaque opération est analysée indépendamment des autres opérations; ses conditions de déclenchement étant seulement soumises à la présence d'événement (au moins un) et d'état. Pour les états, le concepteur ne cherche pas à modéliser, voire à connaître, les opérations qui ont antérieurement produit ces états. L'opération ainsi modélisée regroupe un ensemble d'activités dont l'exécution ne demande pas d'échange avec l'environnement du domaine ; on formalise essentiellement ses entrées et ses sorties.

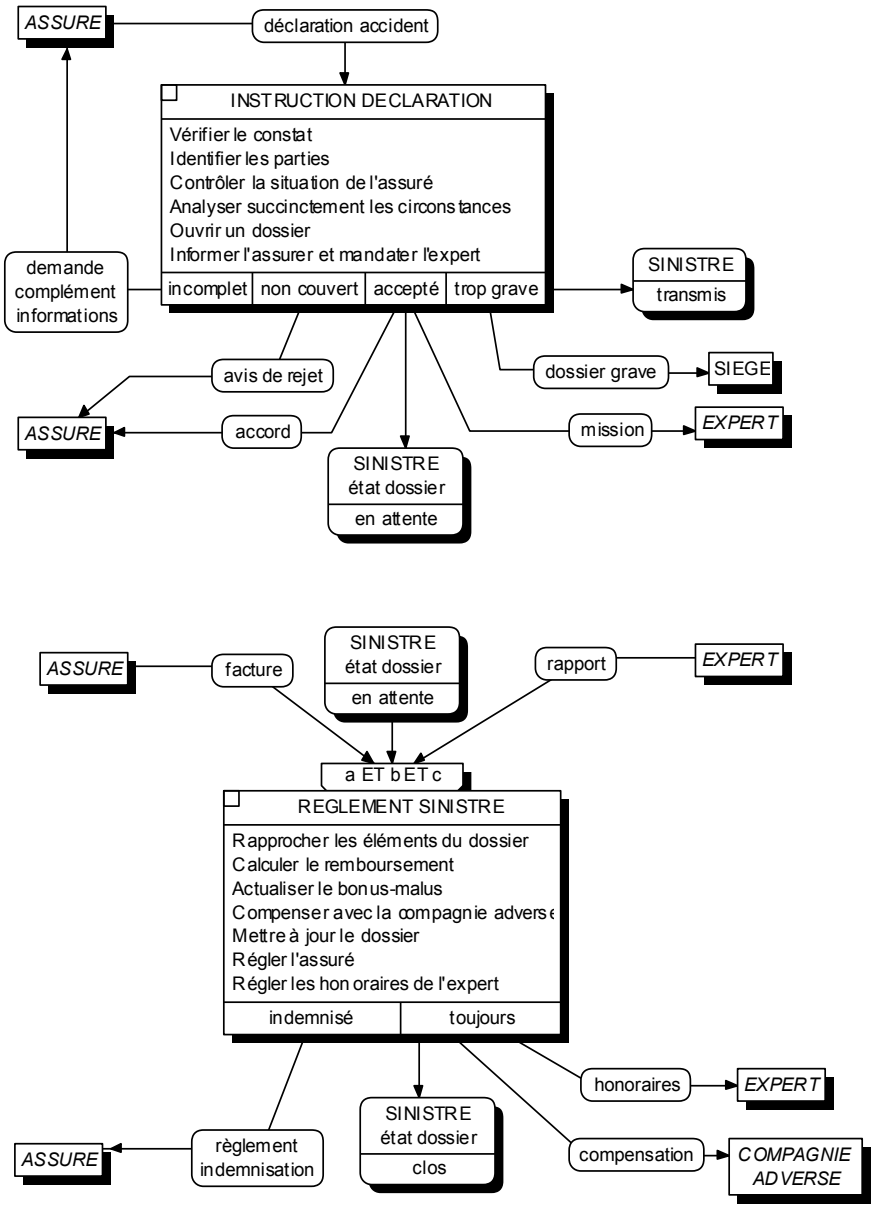


Figure 6.8 : Exemple de modélisation conceptuelle orientée état

La représentation graphique des MCT est alors une simple juxtaposition d'opérations, sans indication d'enchaînement entre opérations. Dans cette modélisation, une opération n'est représentée qu'une seule fois. La figure 6.8 illustre cette approche orientée état.

Ces deux approches sont complémentaires, voire duales. A partir d'un MCT orienté processus, on peut obtenir un MCT orienté état en dupliquant puis en segmentant tous les états qui matérialisent des enchaînements d'opération. A

partir d'un MCT orienté état, on peut reconstituer l'ensemble des processus possibles en reliant les opérations qui partagent le même état (état résultat, opération précédente – état préalable, opération suivante). Les figures 6.6 et 6.8 l'illustrent partiellement (le nombre limité d'opérations, d'états et de processus ne permet pas de mettre en évidence les différences de présentation).

L'approche processus peut paraître plus « naturelle » aux utilisateurs ; on la retrouve d'ailleurs dans le B.P.R. (Business Process Reengineering). L'approche par état permet d'isoler les opérations, facilitant ainsi l'analyse ultérieure de son contenu ; les informaticiens y retrouveront une certaine « approche objet ».

Modularité des modèles conceptuels de traitements

Au niveau normal, le MCT détaille les processus en opérations comme nous l'avons vu précédemment. Sur des projets importants ou complexes, il peut se révéler pratique, au niveau conceptuel, d'élaborer un modèle de degré de détail plus global (voir Niveau d'abstraction et degrés de détail à la fin du chapitre 3)

Au niveau global, la modélisation des traitements s'effectue alors en termes de processus qui se représente par le même symbole que l'opération. Seuls figurent dans le MCT les événements déclencheurs du processus, et les résultats finaux ; des événements intermédiaires qui fractionnent le processus en le laissant dans un état inachevé ne sont pas pris en compte. Ce modèle, de niveau conceptuel, peut être appelé *modèle général des processus.*

On peut également élaborer des modèles conceptuels de traitement plus détaillés dans lesquels on décompose les opérations en opérations plus élémentaires et homogènes, essentiellement autour de structures de données simples. Dans ce degré de détail de modélisation, on exprime complètement et formellement les actions sur les données, les règles de traitements et les états. Ce type de modélisation est appelé *Modèle Conceptuel des Traitements Analytique* dans Merise/2 [Panet, Letouche 94]

Cette stratification des modèles conceptuels des traitements s'inspire des principes de composition/décomposition (refinement) largement utilisés dans des méthodes anglo-saxonnes [SSADM [Longworth, Nicholls 86], SADT 77...]. Nous les utiliserons dans Merise pour faire varier le degré de détail, tout en restant au même niveau de préoccupation.

Pour illustrer cette stratification, la figure 6.9 présente un exemple de modèle général des processus.

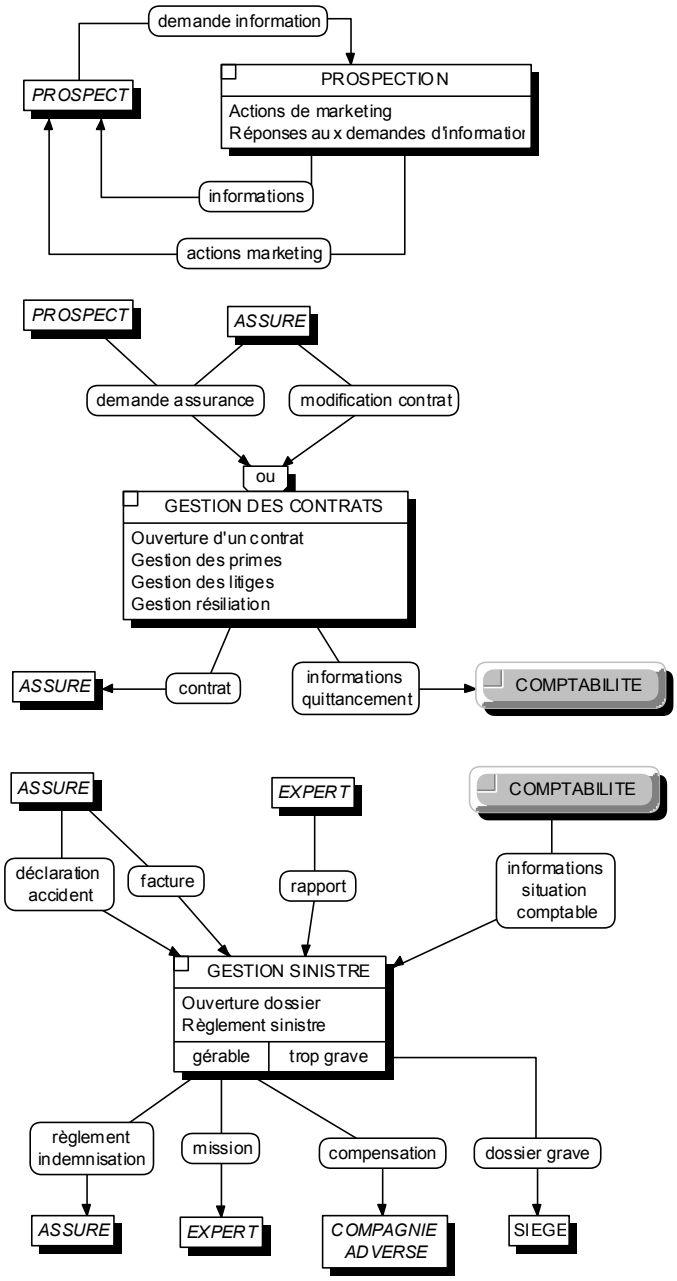


Figure 6.9 : Un exemple de modèle général des processus.

Expression d'un modèle conceptuel des traitements

Bien que les présentations puissent être adaptées au contexte d'utilisation, évoquons les principaux éléments composant la présentation d'un modèle conceptuel des traitements :

Liste descriptive des acteurs.

Graphiques :

- diagramme des flux ;
- schéma d'enchaînement des événements, états, opérations, résultats (appelé schéma du MCT), présenté généralement par processus ;
- schéma d'enchaînement des processus.

Pour chaque opération, description (succincte ou détaillée suivant le niveau d'étude) :

- des événements contributifs et du contenu du message associé ;
- des états préalables à l'opération :
- des conditions liées à la synchronisation ;
- du contenu de l'opération en termes de fonctions ;
- des données utilisées,
- des règles de traitement appliquées,
- des résultats produits et du contenu du message associé ;
- des états résultants,
- des conditions de production de ces résultats.

7

Modélisation conceptuelle des données

Problématique du modèle conceptuel de données (MCD)

Le modèle conceptuel de données (MCD) est la représentation de l'ensemble des données du domaine, sans tenir compte des aspects techniques et économiques de mémorisation et d'accès, sans se référer aux conditions d'utilisation par tel ou tel traitement.

Dans un système d'information en fonctionnement, données et traitements apparaissent intimement liés (surtout du point de vue de l'utilisateur). L'ensemble des informations utilisées, échangées constitue l'univers du discours du domaine. Dans cet univers du discours, on fait référence à des objets concrets ou abstraits (l'assuré, le contrat) et à des associations entre ces objets (le contrat comporte des garanties). L'objectif du modèle conceptuel de données est d'identifier, de décrire par des informations et de modéliser ces objets et associations.

Dans la démarche de construction d'un modèle conceptuel de données, on distingue deux attitudes, correspondant en fait à la connaissance de l'univers du discours acquise par le concepteur :

Une démarche déductive qui s'appuie sur l'existence préalable d'une liste d'informations à structurer; le discours est décortiqué en informations élémentaires.

Une démarche inductive qui cherche à mettre rapidement en évidence les différents concepts évoqués dans le discours, puis à les décrire par des informations.

Ces deux approches ne sont nullement antagonistes et coexistent alternativement dans la pratique. Précisons toutefois que la démarche déductive est plus lourde à mettre en œuvre, et donc difficilement opérationnelle en étude préalable. Par ailleurs notre expérience nous incite à préférer la démarche

inductive qui s'avère plus créative et efficace. En résumé :

- Si le concepteur opte pour une démarche déductive, il doit d'abord constituer une liste d'informations.
- Si le concepteur choisit la démarche inductive, il peut directement, à l'aide du formalisme, construire le modèle conceptuel de données.

Dans les deux cas, la base essentielle reste le discours (parlé ou écrit) de l'utilisateur ou du gestionnaire, exprimé en langue naturelle.

- Les mots utilisés comprennent les termes usuels de la langue, mais aussi des termes spécialisés du domaine.
- Les phrases fournissent, après une analyse pseudo- grammaticale, les principaux objets et les associations entre ces objets.

Bien entendu, on est ici très proche des techniques d'extraction des connaissances [Vogel 88] utilisées avant de construire un système expert.

Constitution d'une liste d'informations

Cette liste d'informations est le résultat d'un recueil d'informations circulant dans le domaine. Elle se présente sans aucune structure de regroupement a priori, tout au plus un classement alphabétique.

Pour constituer cette liste, le concepteur peut procéder de deux façons :

- Ratisser, au gré des entretiens, les informations présentes sur quelques documents.
- Exprimer les messages associés aux événements et résultats, et spécifiés dans le modèle conceptuel de traitements ou le modèle organisationnel de traitements (figure 7.1). Notons que, dans ce cas, confronté à des événements/résultats traduisant un flux physique ou monétaire, le concepteur doit le traduire en terme d'informations (d'où un choix de représentation).

Pour chaque information que le concepteur recueille dans son environnement, avant de l'ajouter à la liste déjà établie, il doit répondre aux questions suivantes :

- La nouvelle information n'a-t-elle pas déjà été répertoriée ? Il est, par exemple, fort probable que l'information n° police apparaisse dans de nombreux messages et documents. Dans ce cas, on considère la « nouvelle » information comme déjà connue.

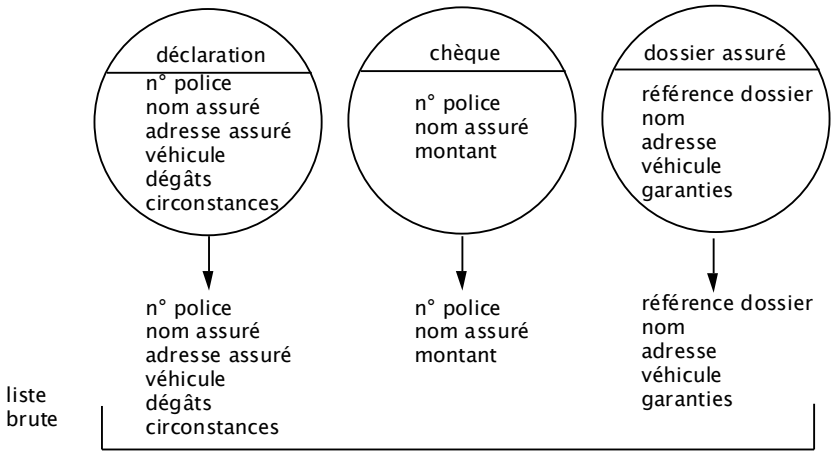


Figure 7.1: Exprimer les messages associés aux événements/résultats.

- La nouvelle information a été déjà répertoriée mais sous une appellation différente. Le concepteur est en présence d'un synonyme. Par exemple, référence contrat et n° police. Après s'être assuré de cette synonymie, le concepteur peut soit prendre en compte les deux informations en notant cette synonymie, soit ne retenir qu'une appellation.
- Une appellation identique existe déjà pour la nouvelle information mais associée à une signification différente. Le concepteur est en présence d'un homonyme. Par exemple, date de livraison (demandée) et date de livraison (effective). Il doit impérativement lever l'ambiguïté en modifiant les appellations des informations.

A la fin de ce travail, le concepteur dispose d'une liste d'informations sans redondance, sans synonyme et sans homonyme. Il prend soin, par ailleurs, d'associer à chaque information une description sous la forme d'un texte libre et éventuellement de mots clés, afin de constituer un catalogue (ou dictionnaire) d'informations.

Formalisme de description des données au niveau conceptuel

Le formalisme utilisé dans Merise est désigné par entité-relation. En dehors du contexte de la méthode, il a été reconnu internationalement par l'ISO [ISO TC97 SC5 WG3 1982], et fait l'objet de nombreux développements. Sa diffusion lui a valu plusieurs appellations : formalisme individuel [Tardieu, Heckenroth, Nanci 75][Tardieu, Nanci, Pascot 79], formalisme entity-

relationship [Chen 76], formalisme entité-relation qui recouvrent, parfois avec quelques nuances, les mêmes idées.

Ce formalisme comporte quatre concepts types de base. Deux concepts sont structuraux, l'*entité type* et la *relation type*; le troisième concept est descriptif, c'est la *propriété type*; le quatrième qualifie la liaison entre entité-type et relation-type, c'est la *cardinalité*. Ce formalisme possède une représentation graphique présentée à la figure 7.2.

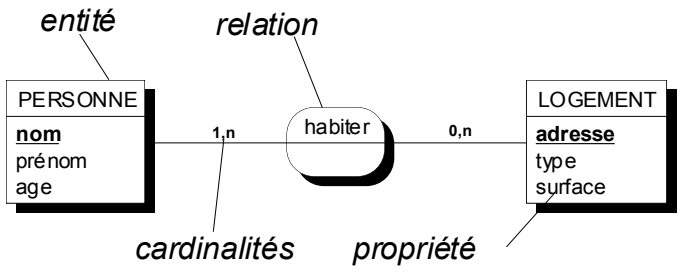


Figure 7.2 : Les concepts du formalisme entité - relation

La propriété type

- Une propriété type est la modélisation d'une information élémentaire présente dans le discours. Elle peut prendre des valeurs; par exemple :

Nom de client : Dupont, Durand, Martin

Date de naissance : 23/07/52, 02/03/63, 25/10/75

Montant du chèque : 250 000 F, 1 392,75 F, 31 745 F

Pourquoi distinguer information et propriété ? La propriété est une manière de modéliser une information, mais toutes les informations ne seront pas systématiquement traduites par une propriété. La propriété subit, pour sa modélisation, un certain nombre de règles que l'information matière première n'a pas à respecter.

La propriété est l'élément descriptif de l'entité type ou de la relation type. Pour prendre sa signification, une propriété est obligatoirement rattachée à une entité type ou à une relation type. Une propriété est unique dans un modèle conceptuel et ne peut être rattachée qu'à un seul concept (entité type ou relation type).

Propriété composée

Dans certains cas, la signification d'une propriété peut être obtenue par la composition d'autres informations, par exemple :

Numéro INSEE : sexe + année + mois + départ + commune + chrono

Référence courrier : initiales auteur + année + n° ordre

Adresse : rue, code postal, ville

On peut opter pour deux modélisations différentes :

- Considérer cette propriété "construite" comme une propriété normale qui suit les mêmes règles que toute propriété et a sa signification intrinsèque, en précisant toutefois la règle de construction. En revanche, il est exclu de considérer une fraction de valeur d'une telle propriété comme ayant une signification propre, par ailleurs non exprimée. Exemple : les deux premiers caractères du code article représentant la famille.
- Utiliser une propriété composée en modélisant éventuellement sa composition sous forme de propriétés. Cette modélisation permet, en particulier en étude préalable, de nommer un ensemble d'informations sans systématiquement en exprimer le détail. Dans ce cas, chaque fraction de valeur d'une propriété composée peut être exprimée par la valeur de chaque propriété composante.

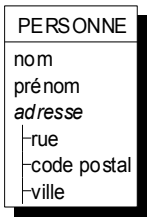


Figure 7.3 : Représentation d'une propriété composée

Propriété générique

Certaines informations, par exemple les adresses, décrivant des entités (ou des relations) différentes sont, à juste titre, modélisées comme des propriétés distinctes. Toutefois, ces propriétés possèdent en commun une signification générique et des caractères descriptifs (composition, format, contraintes de valeurs, ...). On qualifie cette information de référence de *propriété générique*. Une telle propriété n'est affectée à aucune entité ou relation, donc n'est pas représentée sur les schémas; elle ne sert qu'à supporter les caractères généraux communs à l'ensemble des propriétés qui s'y réfèrent. La notion de propriété générique est essentiellement utile dans la modélisation organisationnelle des données (Chap. 9) et lors du passage au modèle logique de données relationnel (Chap. 13).

L'entité type

L'entité type permet de modéliser un ensemble d'objets de même nature, concrets ou abstraits, perçus d'intérêt dans le discours. L'entité type exprime un type, une classe, un ensemble dont les éléments sont appelés *occurrences d'entité type*. La représentation graphique de l'entité est illustrée sur la figure 7.2.

Quelques règles régissent la modélisation en termes d'entité type.

Règle de pertinence

La définition de l'entité type est un choix du concepteur en fonction de l'intérêt qu'elle présente. A partir d'objets concrets ou abstraits du monde réel, le concepteur peut, à son gré, composer diverses modélisations en termes d'entité.

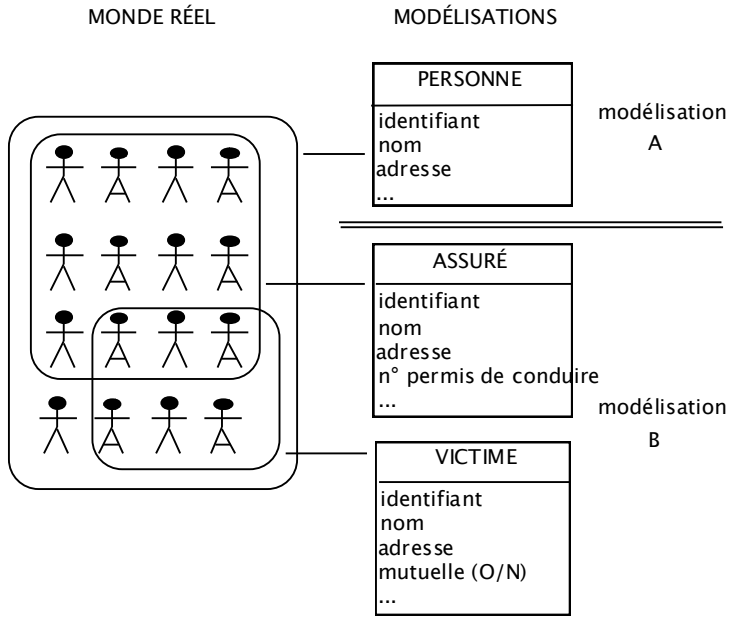


Figure 7.4 : Différentes modélisations possibles.

Par exemple, dans le cas du domaine de l'assurance auto, différentes modélisations de l'environnement perçu sont possibles (voir figure 7.4).

Il est probable que l'assureur préférera la modélisation B, qui peut mieux convenir à sa manière de percevoir son environnement en fonction de son activité. L'assureur perçoit alors douze assurés et six victimes. Sauf modélisation spécifique complémentaire, cet assureur ne saura pas que certains assurés sont également victimes; pour lui, ce sont des occurrences d'entités différentes.

Règle d'identification

L'on doit pouvoir faire référence distinctement à chaque occurrence de l'entité.

Pour cela l'entité type doit être dotée d'un *identifiant*. Cet identifiant est une propriété telle que, à une valeur de l'identifiant, corresponde une seule occurrence de l'entité type.

Cette correspondance biunivoque entre l'occurrence de l'entité type et la valeur de son identifiant doit être vérifiée au présent mais également confirmée dans le futur. Le choix d'un identifiant est un problème délicat. On peut opter pour :

- une propriété « naturelle », par exemple le nom d'un pays pour l'entité pays;
- une propriété « artificielle », inventée par le concepteur pour identifier l'entité qu'il vient de concevoir (tous les numéros, références, codes, etc., en sont l'illustration);
- une propriété composée en s'assurant que la règle de composition ne générera pas de doublons; on parle alors d'identifiant composé; par exemple nom + prénom + date et lieu de naissance;
- un identifiant relatif, par exemple n° allocataire + n° ordre (voir Identifiant relatif).

Le concepteur doit également être prudent dans la reprise d'identifiants issus de la liste d'informations car cela peut ainsi reconduire involontairement la perception initiale de l'entité type.

Enfin un identifiant d'une entité type doit être :

- *univalué* : à une occurrence correspond une seule valeur pour un identifiant donné;
- *discriminant* : à une valeur correspond une seule occurrence de l'entité;
- *stable* : pour une occurrence donnée d'entité, une fois affectée une valeur à son identifiant, cette valeur doit être conservée jusqu'à la destruction de l'occurrence;
- *minimal* : s'agissant d'un identifiant composé, la suppression d'un de ces composants lui ferait perdre son caractère discriminant.

Notons que certaines entités peuvent avoir plusieurs propriétés possédant les qualités d'identifiant; ils sont qualifiés d'identifiants *alternatif*s.

Règle de distinguabilité

Les occurrences d'une entité type doivent être distinguables. Face à deux objets du monde réel, et par rapport à la modélisation qu'il en fait, le concepteur doit pouvoir dire : C'est la même occurrence, ce n'est pas la même occurrence ! Cette distinguabilité induit la compréhension de l'entité type et se traduit par le choix de l'identifiant.

Prenons l'exemple de livres issus d'une même bibliothèque.

Un lecteur percevra trois occurrences distinctes (les deux César étant le même pour lui), tandis que le bibliothécaire percevra quatre occurrences distinctes(Figure 7.5).

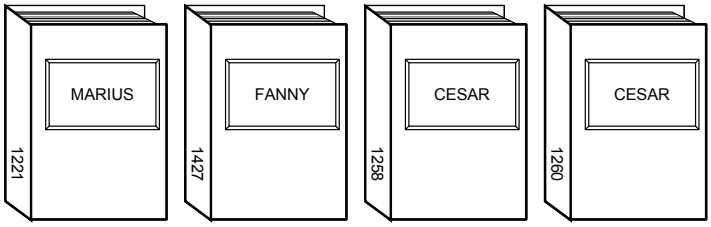


Figure 7.5 : Règles de distinguabilité

Nous attirons l'attention du concepteur sur l'extrême importance que revêt ce choix de distinguabilité, intimement lié à l'identifiant, et qui bien souvent sera la source de quiproquos ultérieurs, voire de la remise en cause du système d'information.

Règle de vérification

L'entité type est décrite par une liste de propriétés. Chaque propriété rattachée à l'entité type doit impérativement suivre la règle suivante, dite de vérification (ou de non-répétitivité) :

A toute occurrence de l'entité type, il ne peut y avoir, dans la mémoire du système d'information, au plus qu'une valeur de la propriété.

Si la réponse à cette règle est négative, la propriété concernée ne peut appartenir à l'entité type.

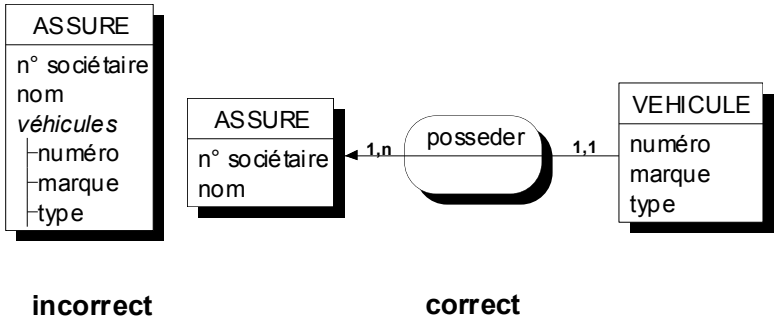


Figure 7.6 a: Règle de vérification

Si un assuré peut avoir plusieurs véhicules assurés, alors les propriétés numéro, marque et type ne peuvent appartenir à l'entité ASSURÉ. Le concepteur devra faire appel à une autre modélisation (nouvelle entité et nouvelle relation). Cette règle a d'abord pour objectif de faire modéliser explicitement les éventuels objets à l'origine de la multiplicité de valeurs des propriétés (Figure 7.6a) .

Cette règle d'unicité de valeur peut toutefois être non respectée dans les cas suivants:

- Si la multiplicité des valeurs d'une propriété est exclusivement due à la conservation des valeurs successives prises par cette propriété au cours du temps, on considérera que cette propriété, pour sa valeur présente, respecte la règle de vérification. Le problème sera abordé et résolu par la modélisation des historiques (voir Modélisation des historiques).
- Si la multiplicité des valeurs exprime une liste de valeurs sans pour autant traduire la présence d'objets d'intérêt à modéliser, on modélisera une *propriété multivaluée* .
  Exemple: une entreprise ayant une liste de n° de téléphone; l'entité téléphone n'a, en général, que peu d'intérêt (Figure 7.6b).

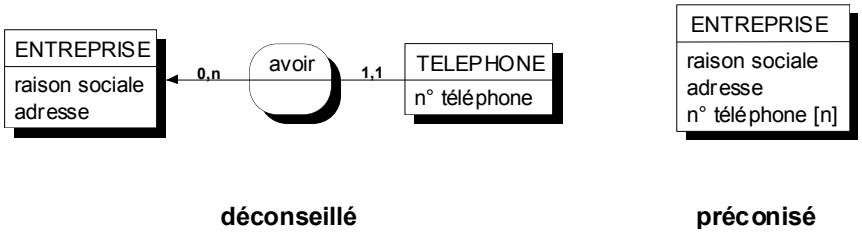


Figure 7.6 b: Exception à la règle de vérification, propriété multivaluée.

Règle d'homogénéité

Il est souhaitable que les propriétés rattachées à une entité type aient un sens pour toutes les occurrences de celle-ci. Cette règle invite le concepteur à s'assurer que, dans sa compréhension de l'entité type, il n'englobe pas plusieurs populations dont certaines ont des caractères spécifiques exprimés dans la liste des propriétés. Le concepteur peut :

- soit confirmer sa modélisation initiale et tolérer que, pour certaines occurrences, des propriétés ne soient pas pertinentes;
- soit remodéliser sa perception en plusieurs entités types.

Cette question sera ultérieurement étudiée sous les thèmes : sous-types d'entités puis liste variable de propriétés.

La relation type

La relation type modélise un ensemble d'associations de même nature entre deux ou plusieurs occurrences d'entités (de types différents ou du même type), perçus d'intérêt dans l'univers du discours. La représentation graphique de la relation est illustrée sur la figure 7.2.

Quelques règles régissent la modélisation en termes de relation type :

Le choix de la relation type

Il dépend de l'intérêt porté par le concepteur aux associations perçues. La relation type n'existe qu'à travers les entités types qui la composent.

On appelle *dimension* le nombre d'entités types composant la relation type. On appelle *collection* la liste de ces entités. L'ordre de citation des entités types dans la collection est sans importance.

Soulignons, à l'occasion, les difficultés pratiques liées à l'appellation des relations. Fréquemment, on désigne les relations types par des noms de verbes; cet usage s'explique et peut se justifier par la similitude existant entre le formalisme entité relation et la grammaire moderne (groupe nominal, groupe verbal).

Il est souhaitable d'utiliser un verbe à l'infinitif. Un verbe exprime parfois une action qui peut alors être assimilée à un traitement; nous conseillons d'utiliser un verbe plus statique plutôt qu'un verbe d'action qui a plus sa place dans les MCT et MOT. La forme active ou passive d'un verbe permet d'orienter la lecture de la relation. On peut éventuellement utiliser un substantif issu du verbe.

De nombreuses relations expriment la notion d'appartenance (appartenir, concerner...). On peut les qualifier en évoquant l'une des entités.

On peut également construire une abréviation à partir des noms des entités. Par contre, nous déconseillons de recourir à la numérotation des relations (R1, R2...).

Identification d'une relation type

Une relation type n'a pas d'identifiant propre. L'occurrence d'une relation type est déterminée par les occurrences des entités types de sa collection. A une combinaison d'occurrences d'entités types composant la collection d'une relation type, il ne peut y avoir au plus qu'une occurrence de cette relation type.

L'identification d'une relation est réalisée par la conjonction des identifiants des entités de sa collection. Dans la figure 7.5, la relation « habiter » est identifiée par la concaténation d'une valeur de *nom* de personne et d'une valeur *d'adresse* de logement.

Une occurrence de relation type ne peut exister que reliée à une occurrence de chacune des entités types de sa collection (pas de patte optionnelle possible dans la relation).

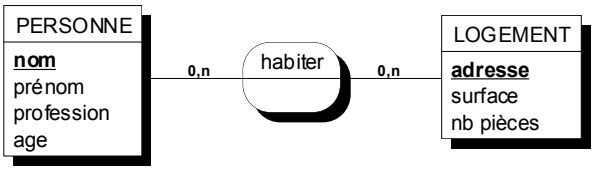


Figure 7.5 : Entités types reliées par une relation type.

Attention, il ne faut pas réciproquement en déduire qu'un concept identifié par la conjonction d'identifiants d'entités doit systématiquement être modélisé comme une

relation. L'intérêt du concepteur pour ce concept peut le conduire à le modéliser comme une entité (règle de pertinence) avec une identification relative multiple (voir Identification relative et entité faible).

Propriétés d'une relation type

La relation type peut être dotée de propriétés (cf. figure 7.6). Il s'agit d'informations qui ne peuvent prendre de sens qu'avec la présence de l'ensemble des entités constituant cette relation type.

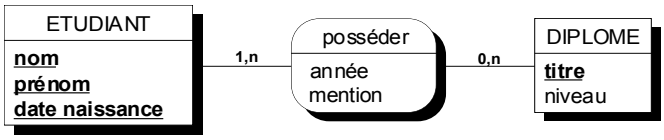


Figure 7.6 a : Relation type avec propriétés

Règles de vérification et de normalisation

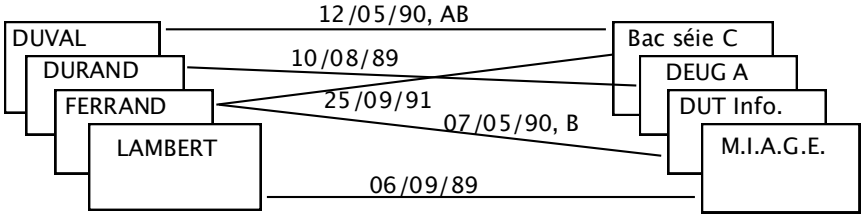
La *règle de vérification* s'applique aux propriétés rattachées à une relation type :

A une occurrence d'une relation type, il ne peut y avoir, dans la mémoire du système d'information, qu'une seule valeur pour chacune des propriétés rattachées à cette relation type.

De nombreuses relations types sont modélisées sans propriétés, on les appelle familièrement *relations vides.*

Il faut également s'assurer que la propriété est correctement affectée à la relation. La propriété a été vérifiée par rapport à l'occurrence de la relation type, c'est-à-dire par rapport à la totalité des entités formant sa collection. Cette propriété ne doit pas également être vérifiée par rapport à un sous-ensemble des entités de la collection. Cette règle s'appelle la *normalisation.* Si l'on peut vérifier une propriété par rapport à un sous-ensemble de la collection, deux cas se présentent :

- Le sous-ensemble est réduit à une seule entité type, la propriété en cause est à rattacher à cette entité type.
- Le sous-ensemble est composé de plusieurs entités types, il faut éventuellement créer une nouvelle relation type de cette dimension et y rattacher la propriété en cause.



Collection de la relation posséder : {personne, diplôme}.

Dimension de la relation posséder : 2.

Propriété propre de la relation posséder :année, mention.

Identification de la relation posséder : nom + titre.

Figure 7.6 b : Occurrences de la relation posséder.

Variété des relations types

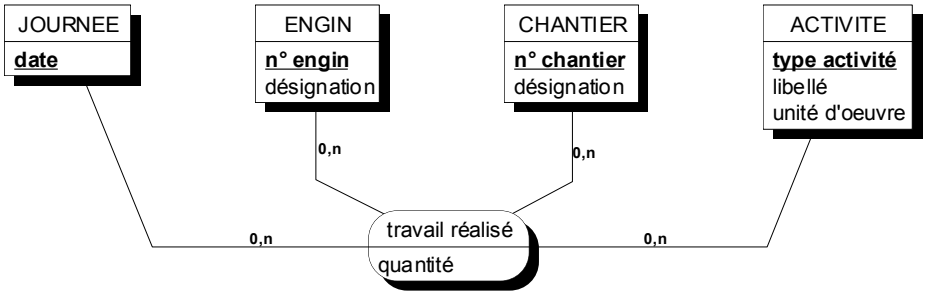
A la différence de la plupart des formalismes de représentation de données proches de l'informatique ainsi que des formalismes « Entity-Relationship » et dérivés, le formalisme entité-relation de Merise propose une grande variété d'expressions des relations. Nous allons en présenter quelques-unes.

La dimension d'une relation est non limitée

Il est possible d'exprimer des relations plus que binaires (n-aires), comme sur la figure 7.7. En pratique, nous n'avons jamais rencontré de relation type de dimension supérieure à 7 (pas plus de 7 entités types dans la collection d'une relation type).

Cela tient, selon nous, à deux facteurs :

- Le cerveau humain ne peut visualiser, manipuler ou mémoriser simultanément que 6 ou 7 éléments distincts (mémoire à court terme).
- La langue naturelle correspond à cette même complexité; ainsi, une phrase en français, sans proposition relative, ordonne autour d'un verbe, au plus, 7 éléments : sujet, complément d'objet direct, complément d'objet second, compléments circonstanciels de temps, de lieu, de manière et d'accompagnement.



Une occurrence de cette relation pourrait être : le 22 février 1995, le bulldozer n° 4589B a réalisé sur le chantier n° 1258 du pont de l'Alpe, un travail de déneigement de 300 m^3

Figure 7.7 : Une relation quaternaire

Remarquons cependant que, dans la pratique, une grande proportion des relations types modélisées est binaire.

Plusieurs relations types peuvent partager la même collection

Situation tout à fait normale où, entre deux ou plusieurs entités types, des associations de significations différentes peuvent exister. La figure 7.8 en est une illustration.

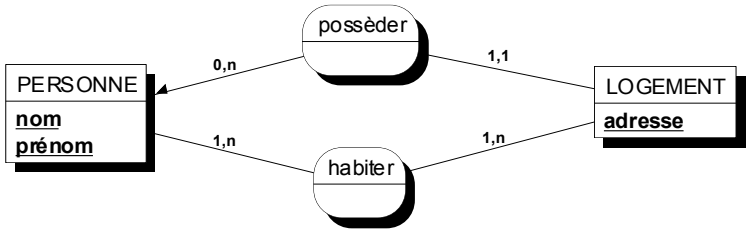
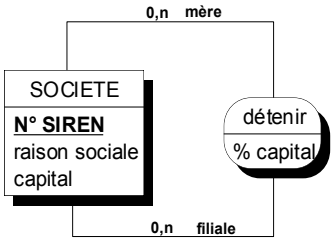


Figure 7.8 : Deux relations partageant la même collection

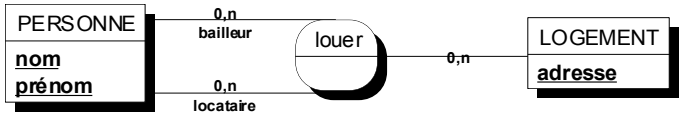
Une même entité type peut apparaître plusieurs fois dans la collection d'une relation type

Une association peut exister entre les occurrences d'une même entité (voire sur la même occurrence). Dans ce cas, il convient de préciser (sur la patte de relation) le *rôle* joué par chacune des occurrences d'entité dans le cadre de la relation; en effet, le plus souvent une relation type n'est pas symétrique (voir figure 7.9).

Ces relations sont souvent appelées *réflexives* surtout lorsque les deux pattes bouclent sur une seule entité.



Les termes mère et filiale précisent le rôle joué par chacune des occurrences de l'entité société dans la relation participer.



Les termes bailleur et locataire précisent le rôle joué par chacune des

occurrences d'entités personne dans la relation louer.

Figure 7.9 : Relations dont la collection fait apparaître plusieurs fois la même entité type.

Les cardinalités d'une entité type dans une relation type

Le terme *cardinalité,* dans le formalisme entité-relation, traduit la *participation* des occurrences d'une entité type aux occurrences d'une relation type.

Cette participation s'analyse par rapport à une occurrence quelconque de l'entité type, et s'exprime par deux valeurs : la *cardinalité minimum* et la *cardinalité maximum*. Ces cardinalités seront représentées dans un modèle conceptuel de données comme sur la figure 7.10.

Figure 7.10 : Les cardinalités.

Ce couple de valeurs se note sur la patte de la relation type concernée par l'entité type dont on qualifie ainsi la participation à la relation type. Bien que des valeurs quelconques puissent être affectées à ces cardinalités, certaines valeurs typiques caractérisent les situations les plus courantes.

Cardinalité mini = 0 : certaines occurrences de l'entité type ne participent pas à la relation; participation optionnelle.

Cardinalité mini = 1 : toute occurrence de l'entité type participe au moins une fois aux occurrences de la relation; participation obligatoire.

Cardinalité maxi = 1 : quand une occurrence de l'entité type participe à la relation, elle n'y participe au plus qu'une fois; unicité de participation.

Cardinalité maxi = n : quand une occurrence de l'entité type participe à la relation, elle peut y participer plusieurs fois; multiplicité de participation. On remarquera qu'au niveau conceptuel on ne cherche pas systématiquement à chiffrer cette multiplicité.

Ainsi, les cardinalités fréquemment utilisées sont :

Participation	Optionnelle	Obligatoire
Unique	0,1	1,1
Multiple	0,n	1,n

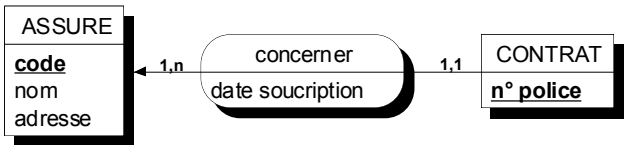
Les cardinalités mini et maxi ont une importance différente. On peut ainsi, si besoin est, laisser la cardinalité mini indéterminée; elle peut alors être notée par le symbole « ? ».

En pratique, on constatera qu'il est plus facile de déterminer les cardinalités des relations binaires que celles des relations ternaires (ou plus). En effet, dans une relation binaire, en fixant une occurrence d'entité, les occurrences de la relation sont assimilables aux occurrences de l'entité restée « libre ». Par contre, dans le cas de relation ternaire (ou plus), en fixant une occurrence d'entité, les occurrences de la relation représentent des n-uples d'occurrences des entités restées libres, ce qui est plus difficile à imaginer.

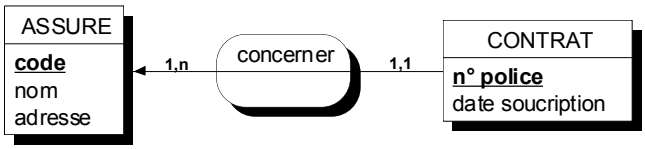
Exemple de cardinalité sur une relation binaire

Une occurrence d'assuré peut être concernée par un ou plusieurs contrats (cardinalité 1,n), tandis qu'une occurrence de contrat ne concerne qu'un et un seul assuré (cardinalité 1,1).

Toute relation binaire avec cardinalité (1,1) ne peut être porteuse de propriété. En effet une telle propriété (voir figure 7.11) migre alors obligatoirement dans l'entité portant cette cardinalité (1,1).



incorrect



correct

Figure 7.11 : Propriété d'une relation binaire avec cardinalité 1,1

Dans la figure 7.11, tout contrat n'a qu'une et une seule date de souscription.

Exemple de cardinalité sur une relation ternaire

La relation ternaire de la figure 7.12 exprime, par exemple, que l'entreprise Dupin a réalisé un montant annuel de 8 500 F de menuiserie à la copropriété Beau-Manoir. De plus les cardinalités précisées s'interprètent ainsi :

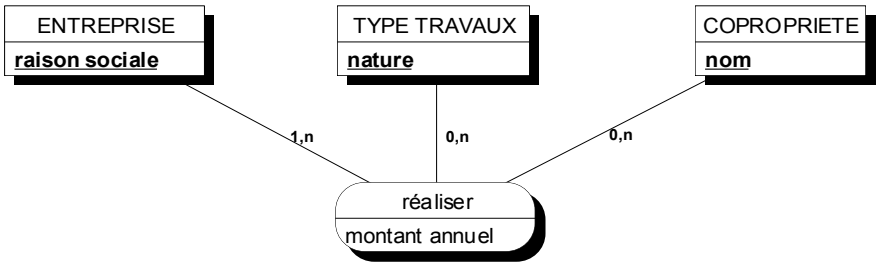


Figure 7.12 : Relation ternaire avec ses cardinalités

ENTREPRISE (1,n)

Toute entreprise a au moins un montant réalisé (le système d'information ne connaît que les entreprises qui ont réalisé des travaux sur les copropriétés).

Une entreprise peut avoir plusieurs montants réalisés (sur plusieurs copropriétés et/ou plusieurs types de travaux).

COPROPRIÉTÉ (0,n)

Une copropriété peut n'avoir fait l'objet d'aucun travail.

Certaines copropriétés peuvent faire l'objet de plusieurs travaux, de même type ou de types différents et/ou réalisés par des entreprises différentes ou la même entreprise.

TYPE TRAVAUX (0,n)

Un type de travaux peut ne faire l'objet d'aucune réalisation.

Il peut y avoir plusieurs montants réalisés pour un type de travaux, cela par plusieurs entreprises ou une seule et/ou concernant plusieurs copropriétés ou une seule.

C'est à l'occasion de la détermination des cardinalités sur des relations de dimension supérieure à 2 que l'on détectera les éventuelles dépendances fonctionnelles (voir plus loin « Contraintes intra-relation »).

Ainsi, dans l'exemple de la figure 7.12, la cardinalité maxi d'entreprise est n, c'est-à-dire qu'une entreprise peut réaliser plusieurs montants de travaux dans plusieurs copropriétés. Supposons que toute entreprise ne soit spécialisée que dans un seul type de travaux. La cardinalité maxi reste toujours égale à n, mais on a mis en évidence une dépendance fonctionnelle complétant la sémantique du modèle. Cette dépendance fonctionnelle permettra de décomposer éventuellement la relation (voir plus loin «Décomposition d'une relation type»).

Types et sous-types d'entités : spécialisation/généralisation

Ces notions de types et de sous-types ont été intégrées au formalisme entité - relation Merise, lors du congrès « Autour et à l'entour de Merise », suite aux

travaux de normalisation du groupe 135, «Conception des systèmes d'information » de l'AFCET [AFCET 91] [Tabourier 91]. Nous nous inspirerons fortement des travaux de ce groupe pour présenter ces extensions.

Spécialisation simple

La spécialisation permet de modéliser dans une population (l'ensemble des occurrences) d'une entité, des sous-populations (sous-ensembles d'occurrences) présentant des spécificités. Ces spécificités peuvent porter sur :

des propriétés,

des relations,

des appellations.

Ces sous-populations sont explicitement modélisées par des entités dites *entités sous-types* par rapport à *l'entité sur-type*. .

Soit le cas d'un cabinet d'assurance gérant des assurés. Parmi ces assurés, le concepteur souhaite distinguer deux sous-populations : les particuliers et les entreprises. En tant qu'assurés, particuliers et entreprises ont des caractéristiques communes. Ils ont en outre des caractéristiques spécifiques, par exemple, la date de naissance et la profession pour les particuliers, le SIRET et la forme juridique pour les entreprises.

La spécialisation consiste tout d'abord à modéliser une entité Assuré, qualifiée de *sur-type* et comportant les propriétés communes aux assurés. Ensuite de considérer les deux entités Particulier et Entreprise comme deux spécialisations de cette entité Assuré. Particulier et Entreprise sont alors appelés *entités sous-types de l'entité sur-type* Assuré.

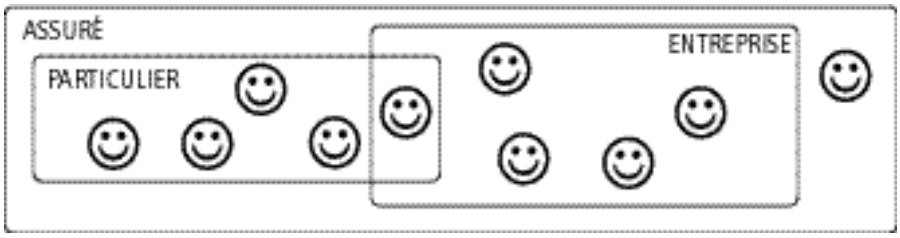


Figure 7.13 : Spécialisations de sous-populations

Les occurrences des entités sous-types sont obligatoirement et également des occurrences de l'entité sur-type. Les entités sous-type n'accueillent que les propriétés spécifiques On dira ainsi que les entités sous-types héritent des propriétés de leur entité sur-type. Ce mécanisme *d'héritage* s'applique aussi à l'identifiant du sur-type. Dans le cas de la spécialisation, les entités sous-types ne possèdent pas d'identifiant propre.

Graphiquement, les entités sous-types et sur-types se représentent comme des

entités classiques. La spécialisation (ou *héritage*) se représente sous la forme d'un triangle qui portera éventuellement une contrainte, relié aux entités concernées; l'entité sur-type est indiqué par un lien fléché. Pour notre exemple, on a la modélisation de la figure 7.13

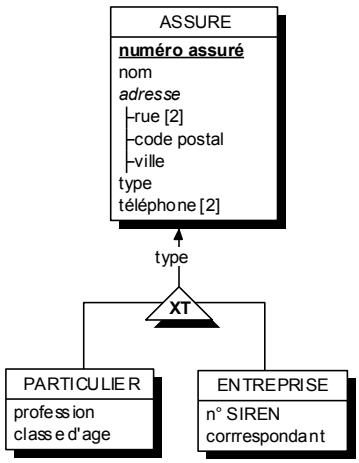


Figure 7.14 : Entités sur-type et sous-types

La spécialisation d'une entité sur-type en entités sous-types peut s'effectuer selon un *critère de spécialisation*: explicitable par :

des valeurs d'une propriété du sur-type,

une règle de gestion.

Certaines relations types peuvent également n'avoir de signification que par rapport à une entité sous-type.

Une spécialisation peut comporter un nombre quelconque de sous-types. Une entité sous-type d'une spécialisation peut être également sur-type d'une autre spécialisation; on obtient ainsi une arborescence de spécialisations (figure 7.15).

Dans la spécialisation, l'entité sur-type, porteuse de l'identifiant, préexiste par rapport aux entités sous-types qui ne sont que des déclinaisons de l'entité sur-type. Le processus de perception va du général au particulier.

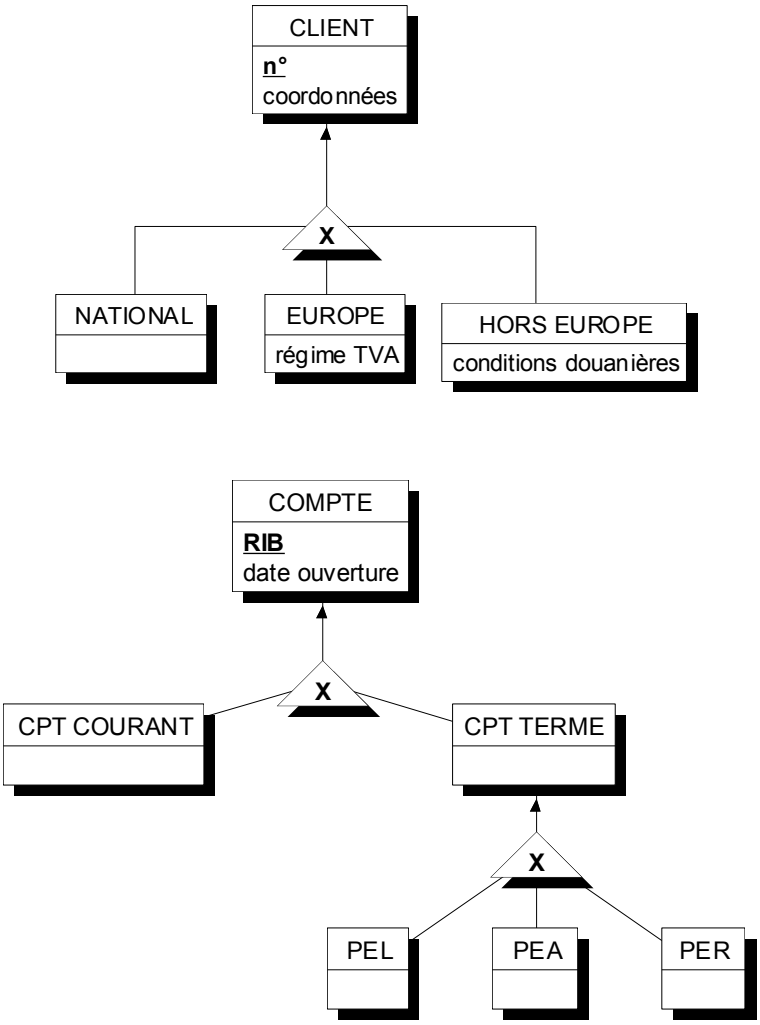


Figure 7.15 : Diversité de spécialisations

Spécialisations multiples

Le découpage d'une population en sous-populations peut s'effectuer selon plusieurs critères, chaque critère produisant une spécialisation en différents sous-types. La multiplicité des critères peut parfois conduire à une combinatoire difficile à maîtriser.

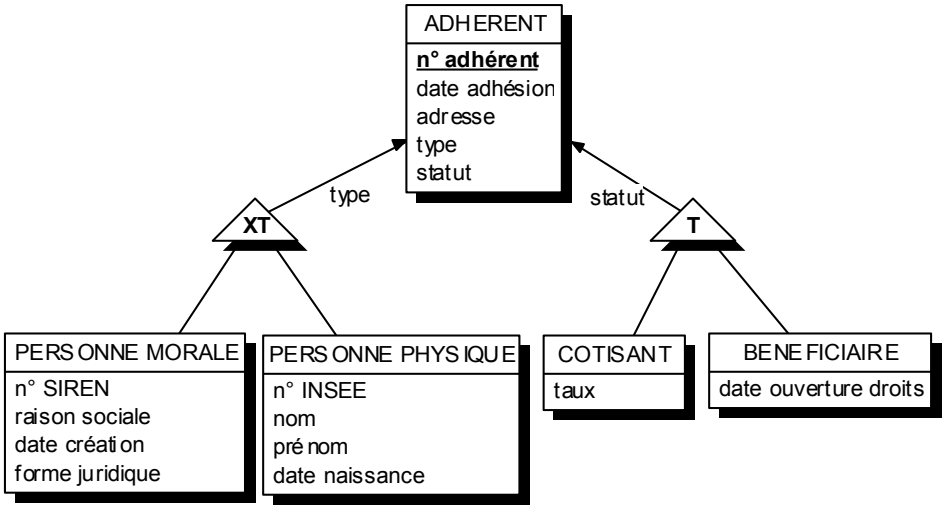
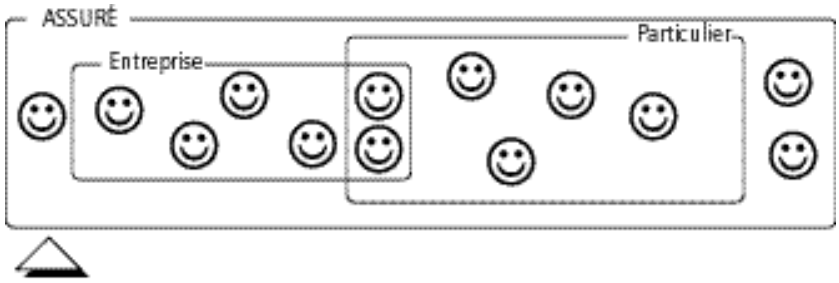


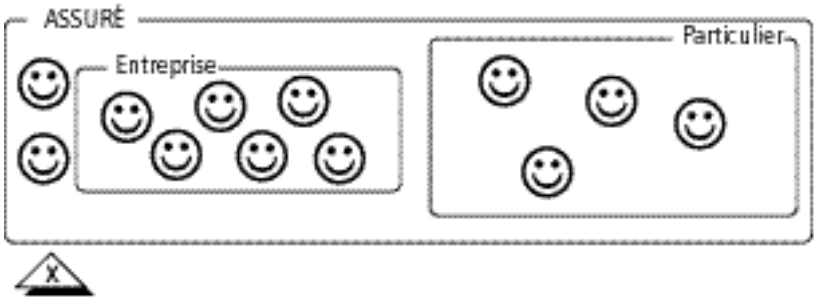
Figure 7.16: Spécialisations multiples

Contraintes sur spécialisations

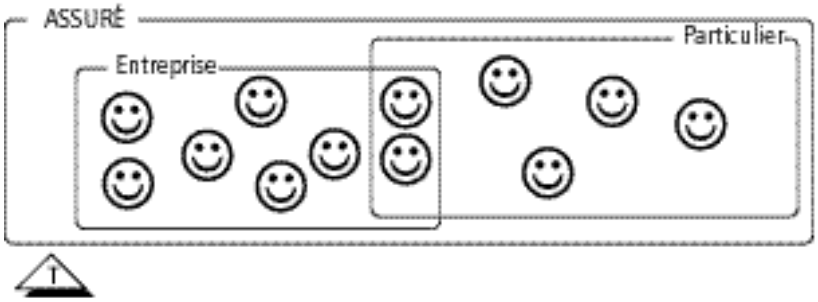
Les contraintes sur spécialisation expriment les participations des occurrences de l'entité sur-type aux entités sous- types. On retrouvera les contraintes ensemblistes classiques.



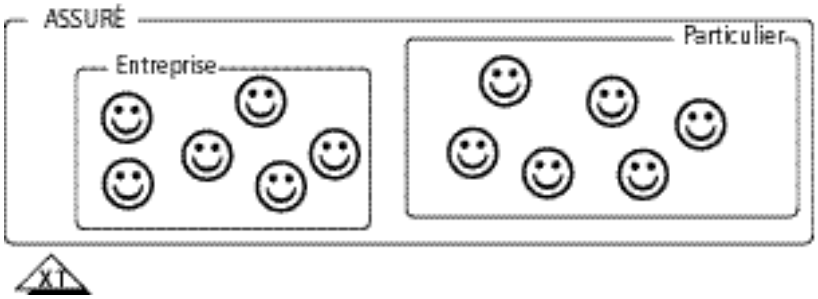
Pas de contrainte : Un assuré peut être une entreprise, un particulier, un particulier et une entreprise, ni un particulier ni une entreprise.



Exclusivité : : Un assuré peut être une entreprise, un particulier,, ni un particulier ni une entreprise, mais ne peut pas être entreprise et assuré.



Totalité : Tout assuré est un particulier, une entreprise, ou les deux

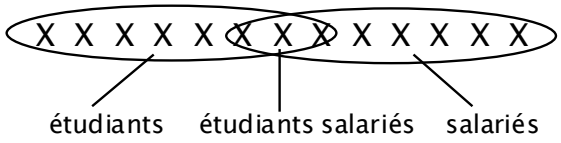


Partition : Tout assuré est soit un particulier, soit une entrepris.

Ces notions de contraintes sur spécialisation sont identiques à celles qui seront développées plus loin pour les relations dans le paragraphe "Contraintes sur la participation d'une entité à plusieurs relations"

Spécialisations à sur-types multiples

Considérons la population des étudiants et la population des salariés. On peut vouloir s'intéresser aux étudiants qui sont aussi salariés et vice versa. Ce qui revient à considérer l'intersection des populations d'étudiants et de salariés :



Cela conduit à faire émerger dans la modélisation une entité étudiant salarié, sous-type d'une part de l'entité étudiant, et d'autre part de l'entité salarié. On observe alors une double spécialisation à sur-types multiples de cette nouvelle entité étudiant salarié, modélisée sur la figure 7.17.

Dans ce cas l'entité étudiant salarié hérite des propriétés d'étudiant et de salarié.

L'identifiant de l'entité étudiant salarié est soit l'identifiant de l'entité étudiant (n° inscription), soit l'identifiant de l'entité salarié (n° matricule). On est en

présence d' *identifiants alternatifs*.

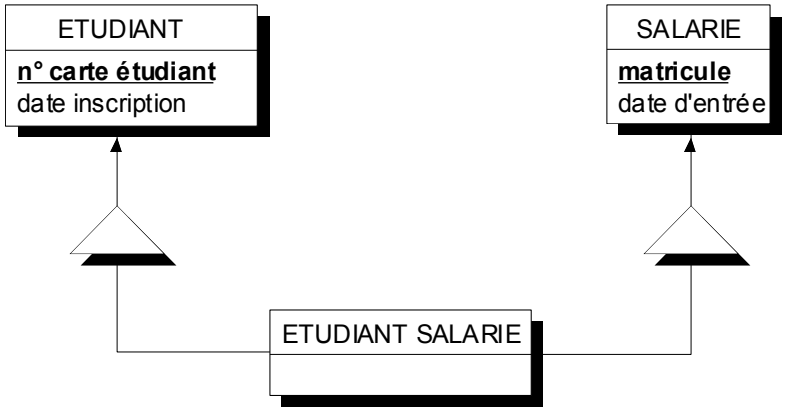


Figure 7.17 : Un exemple de spécialisations à sur-types multiples.

Dans ce cas l'entité étudiant salarié hérite des propriétés d'étudiant et de salarié.

L'identifiant de l'entité étudiant salarié est soit l'identifiant de l'entité étudiant (n° inscription), soit l'identifiant de l'entité salarié (n° matricule). On est en présence d' *identifiants alternatifs*.

Généralisation

Dans la généralisation, inversement à la spécialisation, ce sont les entités sous-types qui préexistent. En conséquence, les identifications de ces entités sous-types sont indépendantes de l'identification de l'entité sur-type. Les entités sous-types ont leurs propres identifiants comme le montre la modélisation de la figure 7.18. L'identifiant hérité du sur-type est alors un *identifiant alternatif* dans chacun des sous-types. Le processus d'héritage, la représentation graphique ainsi que les contraintes sont identiques à la spécialisation.

La généralisation apparaît comme une « mise en facteurs communs » de propriétés. Le processus de perception va du particulier au général.

En spécialisation comme en généralisation, on obtient dans les deux cas une structure de sur-type / sous-types avec héritage. La différence porte d'abord sur le processus de perception et se traduit au niveau de l'identification.

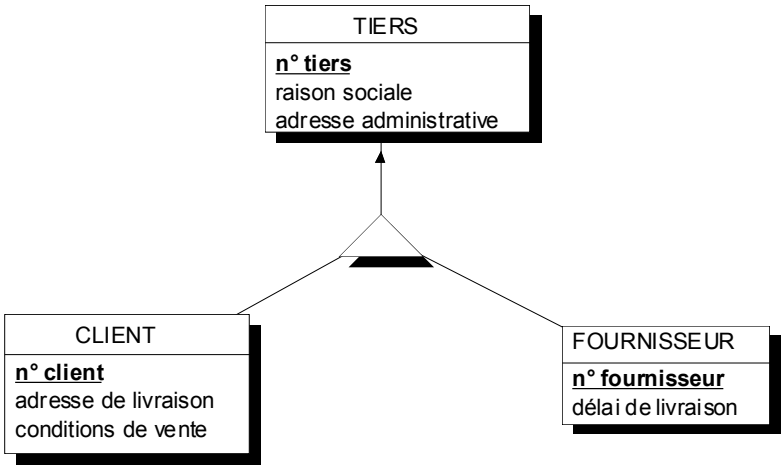


Figure 7.18 : Un exemple de généralisation sur entités sous-types.

Restrictions et sous-types de relations

Les types et sous-types de relations concernent la restriction de relations à des sous-types d'entités. Pour illustrer cette restriction, considérons le modèle suivant : une entité sur-type employé, une entité secrétaire, sous-type d'employé. Soit une troisième entité projet auquel peuvent être affectés des employés, au travers de la relation travailler. Bien qu'un projet comporte plusieurs employés, supposons qu'il y ait au plus une secrétaire qui puisse travailler sur un projet donné.

Ainsi, pour les employés secrétaires, il y a modification des cardinalités de la relation travailler. On prendra en compte une telle situation en introduisant une nouvelle relation, *restriction* de la relation travailler, appelée gérer, et dont les occurrences sont celles de travailler pour lesquelles l'employé est une secrétaire. Il est alors possible sur cette restriction de la relation de préciser de nouvelles cardinalités. On représentera cette restriction de relation comme une spécialisation de relation comme l'indique la modélisation de la figure 7.19.

Notons que la relation gérer hérite des éventuelles propriétés de la relation travailler et peut comporter des propriétés propres qui n'auraient pas de sens pour toutes les occurrences de la relation originale travailler. La restriction de relation peut être considérée comme une relation *sous-type* de la relation *originale*.

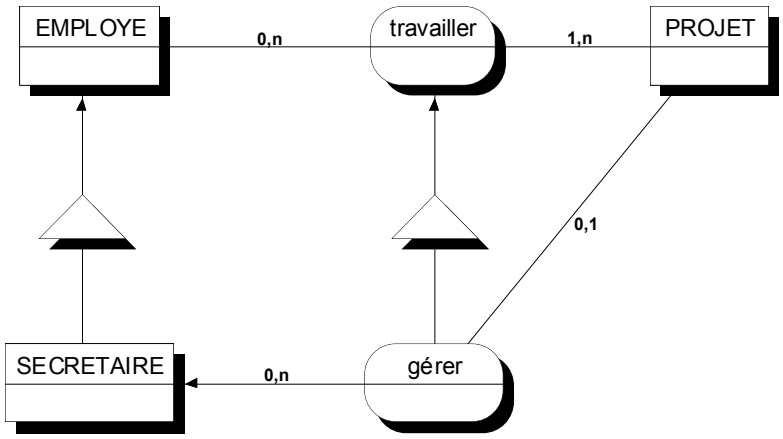


Figure 7.19 : Restriction d'une relation type.

Contraintes intra-relation

En mathématiques, la notion de dépendance fonctionnelle entre deux ensembles A et B exprime qu'à un élément a de A correspond au plus un élément b de B; on note:

A ———> B

L'ensemble de départ peut être simple, ou composé par le produit de deux ou plusieurs ensembles, à un couple (a,b) correspond un seul c :

A x B ———> C

On appelle l'ensemble (ou les ensembles) de départ l'*émetteur* et l'ensemble d'arrivée, la *cible* de la dépendance fonctionnelle.

Dans le formalisme conceptuel de données de Merise, cette notion de dépendance fonctionnelle s'applique, entre autres, au sein d'une relation type entre deux ou plusieurs entités types de sa collection. Nous allons étudier plusieurs cas pouvant se présenter.

Représentation graphique générale des contraintes intra-relation

Les dépendances fonctionnelles (ou contraintes d'intégrité fonctionnelles ou CIF) sur une relation ne peuvent pas toujours être spécifiées par les cardinalités définies sur ses pattes. Il est nécessaire d'introduire un graphisme spécifique. La représentation graphique générale (voir figures 7.20 et 7.21) pour ces dépendances fonctionnelles est la suivante :

un cercle dans lequel est indiqué CIF (éventuellement indicé);

un lien en pointillé indique la relation sur laquelle s'applique la contrainte;

un lien plein non fléché indique la (ou les) entité(s) émettrices de la

dépendance;

un lien plein fléché indique l'entité cible de la dépendance.

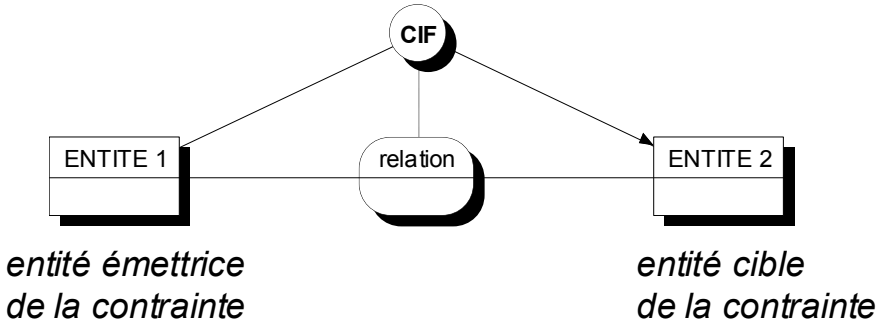


Figure 7.20 : Représentation graphique de dépendances fonctionnelles sur une relation inaire.

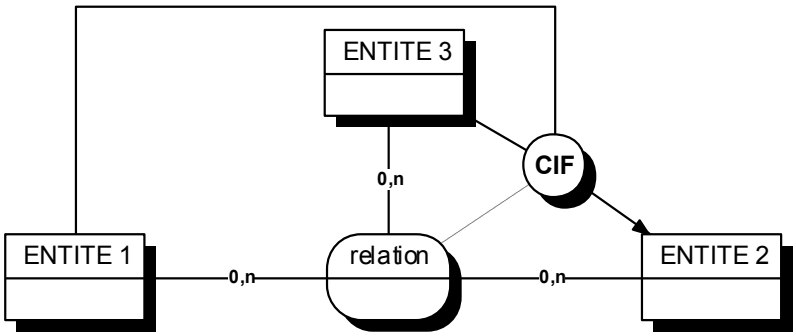


Figure 7.21 : Représentation graphique de dépendances fonctionnelles sur une relation ternaire.

Dépendance fonctionnelle sur une relation binaire

Une dépendance fonctionnelle sur ce type de relation exprime qu'à partir d'une occurrence d'une entité type lui correspond (au plus) une seule occurrence de l'autre entité type de la collection. On constate, dans ce cas, qu'il y a correspondance entre :

la cardinalité maximum = 1

et l'existence d'une dépendance fonctionnelle.

Ces relations types seront couramment appelées *binaires fonctionnelles.*

Exemple de relation binaire fonctionnelle :

appartient

voiture ——————————> personne

Une voiture n'appartient qu'à une personne au plus.

Ces relations binaires fonctionnelles sont très utiles car elles permettent de faire référence à une occurrence d'une entité par l'intermédiaire d'une autre entité et d'une relation (exemple : le propriétaire du véhicule 1234 PX 13).

Pour les dépendances fonctionnelles sur relation binaire, nous préférerons, au graphisme général, « intégrer » la dépendance fonctionnelle à la relation en fléchant la patte la reliant à l'entité cible (voir figure 7.22).

Figure 7.22 : Dépendance binaire fonctionnelle.

Dépendances fonctionnelles sur une relation n-aire

On peut avoir plusieurs catégories de dépendances fonctionnelles (ou contraintes d'intégrité fonctionnelles):

- des dépendances fonctionnelles simples (1 émetteur),
- des dépendances fonctionnelles composées (n-uple d'émetteurs) mais n'englobant pas la totalité de la collection de la relation,
- des dépendances fonctionnelles composées (n-uple d'émetteurs) englobant la totalité de la collection.

Les deux premiers cas représentent souvent une situation provisoire dans le processus de conception, et seront traités par l'opération dite de décomposition (voir plus loin, « Décomposition de relation type »). Le dernier cas sera noté graphiquement et exprimé lors de la description de la relation type.

Lors de la détermination des cardinalités, la mise en évidence de ces dépendances fonctionnelles doit être une opération systématique pour les relations de dimension supérieure à 2.

Plusieurs dépendances fonctionnelles peuvent être définies au sein d'une relation. Dans ce cas, qui reste en pratique exceptionnel, il convient d'effectuer

un certain nombre de contrôles, fondés sur la logique, et qui peuvent conduire à l'élimination de certaines dépendances fonctionnelles [Tardieu, Nanci, Pascot 79].

Exemple de dépendance fonctionnelle simple

Reprenons l'exemple déjà utilisé concernant des types de travaux réalisés par des entreprises sur des copropriétés. Supposons que chaque entreprise n'effectue qu'un seul type de travaux; soit :

travailler

entreprise ——————————> type de travaux

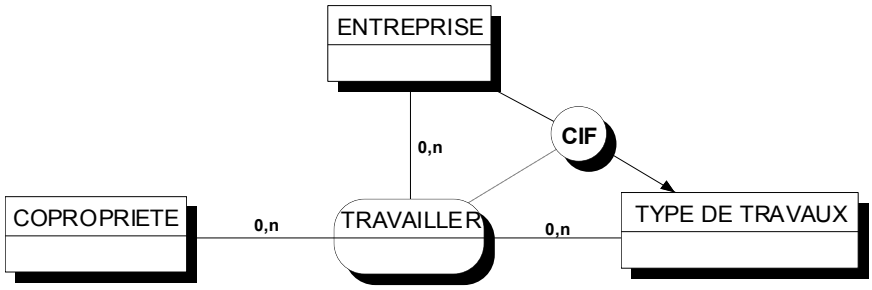


Figure 7.23 : Dépendance fonctionnelle simple sur relation ternaire.

Cette dépendance fonctionnelle, ou CIF, ne peut être représentée au travers des cardinalités de la relation. Il est nécessaire d'utiliser le graphisme spécifique déjà proposé; de plus, il est recommandé d'accompagner ce graphisme d'un texte explicatif, comme l'illustre la modélisation de la figure 7.23.

Rappelons qu'une telle dépendance fonctionnelle est provisoire et sera traitée dans la décomposition.

Exemple de dépendance fonctionnelle composée englobant la totalité de la collection

Pour tout module d'emploi du temps (exemple, le mardi de 9 heures à 10 heures), un professeur ne fait cours que dans une seule salle; soit :

faire cours

module x professeur ——————————> salle

La modélisation graphique associée est celle de la figure 7.24

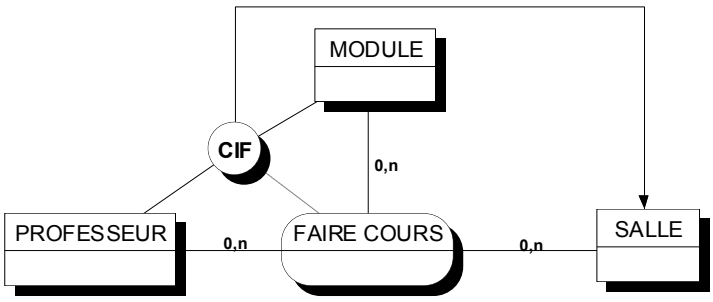


Figure 7.24 : Dépendance fonctionnelle composée.

Lorsque ce type de contrainte est la seule portée par la relation, nous suggérons de l'intégrer à la relation en fléchant la patte la connectant à l'entité cible comme sur la figure 7.25.

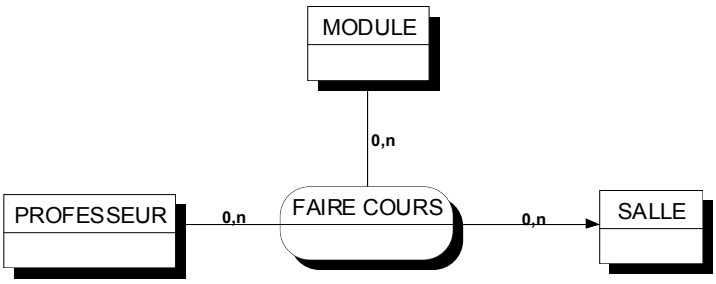


Figure 7.25 : Dépendance fonctionnelle composée sur relation ternaire.

Contraintes inter-relations

Les cardinalités et dépendances fonctionnelles précédentes s'appliquaient, sous forme de contraintes, dans le cadre d'une même relation type. D'autres situations présentent des conditions à exprimer entre deux ou plusieurs relations types.

Contraintes sur la participation d'une entité à plusieurs relations

Ces conditions concernent fréquemment la coexistence d'occurrences de relations types au départ d'une entité type commune. Nous allons étudier différentes situations pouvant se présenter, comme l'illustre la figure 7.26.

Ce type de contrainte est symbolisé (voir figure 7.27) par :

un cercle dans lequel est indiqué le type de contrainte (éventuellement indicé);

un lien en pointillé qui indique l'entité impliquée dans la contrainte;

un lien plein qui indique les relations concernées par la contrainte.

Il est conseillé d'accompagner le modèle graphique d'un bref texte explicatif.

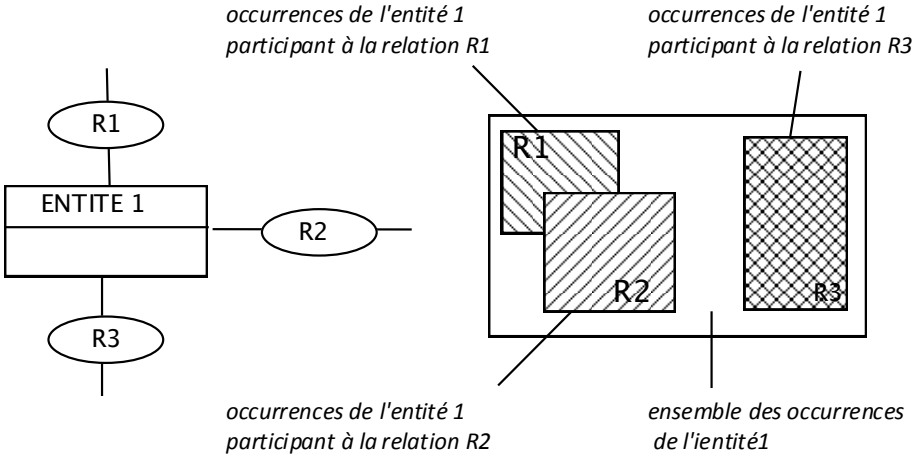


Figure 7.26 : Participation d'une entité à plusieurs relations.

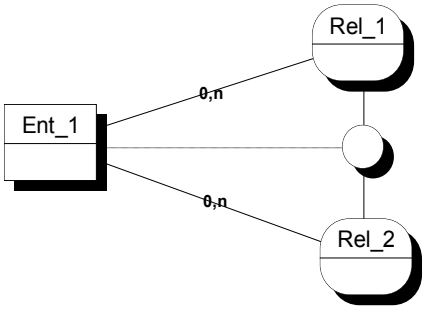


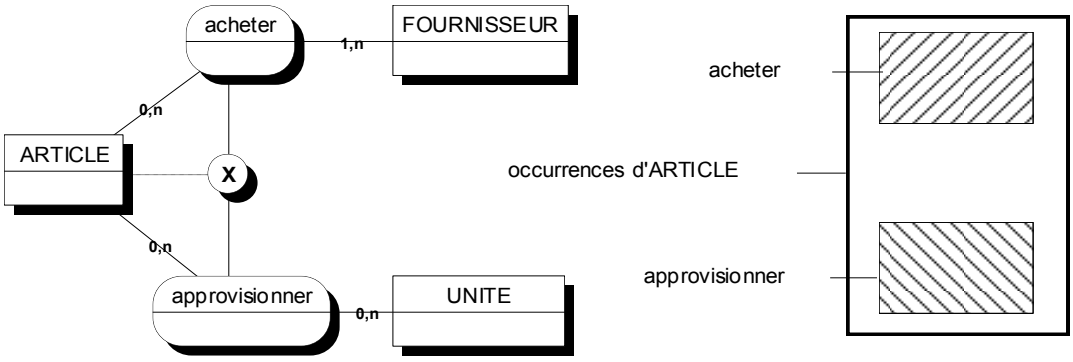
Figure 7.27 : Représentation graphique générale des contraintes interrelations.

Exclusivité de participation d'une entité à plusieurs relations

Deux (ou plusieurs) relations types, au départ d'une entité type commune, peuvent avoir des existences, en termes d'occurrences, mutuellement exclusives. On l'exprime par une contrainte X.

Exemple : un article peut être acheté chez des fournisseurs, approvisionné par des unités de production extérieures, ou élaboré (ou assemblé) directement dans le domaine; il ne peut cependant être à la fois acheté et approvisionné. On a ici une contrainte d'exclusivité de la participation de l'entité article aux relations acheter et approvisionner. La modélisation associée est celle de la

figure 7.28.



Par rapport à l'entité article, les relations acheter et approvisionner sont mutuellement exclusives.

Figure 7.28 : Un exemple d'exclusivité de participation.

Il est possible d'affiner cette exclusivité. En effet, soit une entité A participant à deux relations R1 et R2, trois cas d'exclusivité sont possibles (voir figure 7.29).

Cas 1 : il ne peut y avoir double participation d'une occurrence de A à R1 et R2 (cas précédent).

Cas 2 : étant donné une participation de A à R1, il ne peut y avoir participation de A à R2; par contre, si A participe à R2, il n'y a pas d'exclusivité de participation à R1.

Cas 3 : étant donné une participation de A à R2, il ne peut y avoir participation de A à R1; par contre, si A participe à R1, il n'y a pas d'exclusivité de participation à R2.

On peut préciser graphiquement ces trois cas (voir figure 7.29).

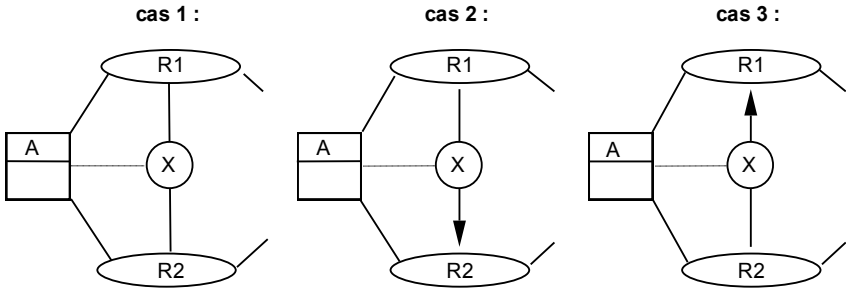
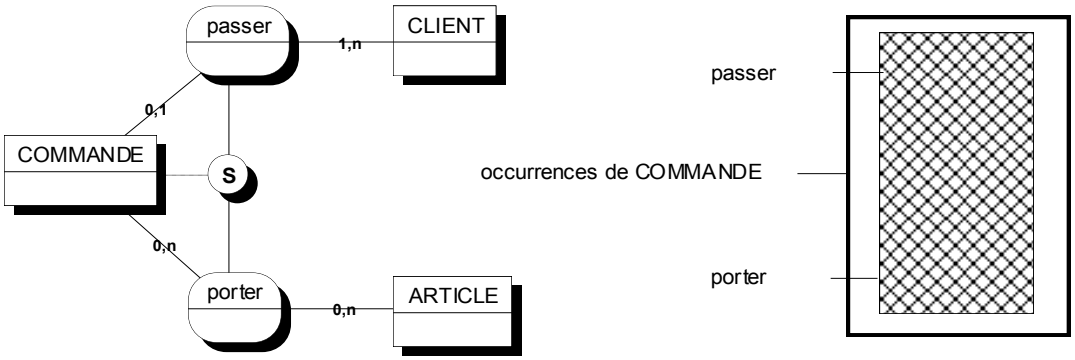


Figure 7.29 : Différentes modalités d'exclusivité de participation.

Simultanéité de participations d'une entité à plusieurs relations

Toute occurrence de l'entité type participe de façon simultanée à deux (ou plusieurs) relations types. On l'exprime par une contrainte S.

Par exemple, une commande portant sur des articles est obligatoirement passée par un client et réciproquement (voir figure 7.30).



Par rapport à l'entité commande, les relations passer et porter sont simultanées.

Figure 7.30 : Un exemple de simultanéité de participation.

La cardinalité mini = 0 de COMMANDE dans PASSER signifie que certaines commandes ne sont pas passées par des clients (commandes internes). La cardinalité mini = 0 de COMMANDE dans PORTER signifie que des commandes peuvent porter sur autre chose que des articles (des prestations par exemple).

Totalité de participations d'une entité à plusieurs relations

Soit une entité type participant à deux (ou plusieurs) relations types, toute occurrence de l'entité participe au moins à une des relations. On l'exprime par une contrainte T.

Par exemple, tout véhicule est au minimum relié soit à un contrat par la relation couvrir, soit à un sinistre par la relation impliquer, soit les deux (voir figure 7.31).

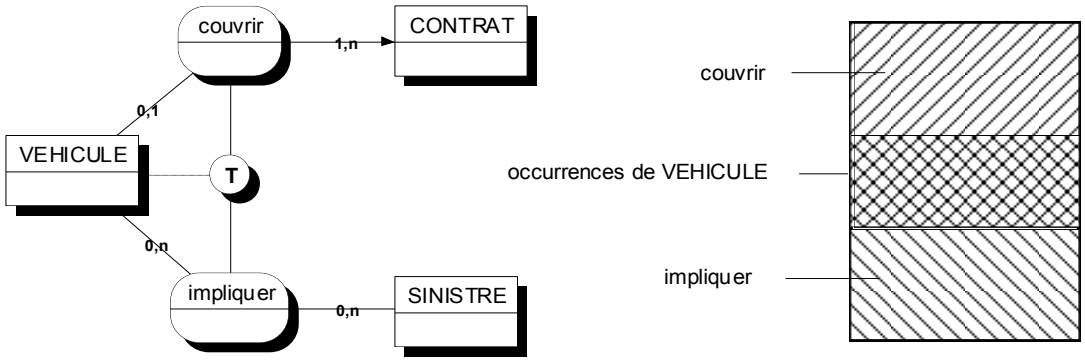
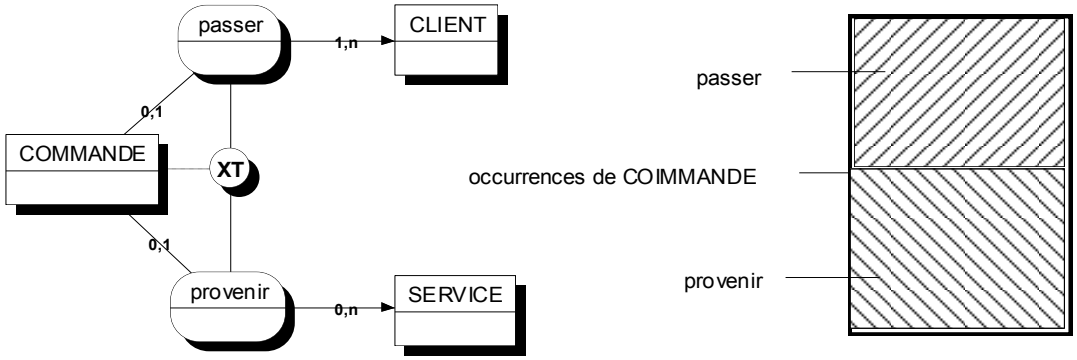


Figure 7.31 : Un exemple de totalité de participation.

Exclusivité et totalité de participations d'une entité à plusieurs relations (Partition)

Le cas de la partition est en fait un cumul des contraintes d'exclusivité X et de totalité T. On l'exprime par une contrainte XT

Soit une entité type commande participant à deux relations types passer et provenir; toute occurrence de l'entité commande participe soit à la relation passer, soit à la relation provenir (voir figure 7.32)..

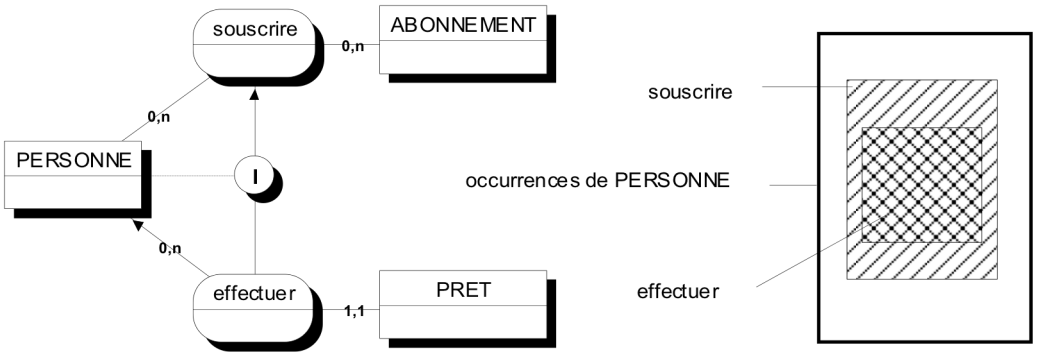


7.32 : Un exemple de participation totale et exclusive

Inclusion de participations d'une entité à plusieurs relations

Soit une entité type participant à deux (ou plusieurs) relations types R1 et R2, toute occurrence de l'entité participant à la relation R1 participe à la relation R2. On l'exprime par une contrainte I, dont l'émetteur est la relation R1 et la cible la relation R2 (flèche vers R2).

Par exemple, une personne qui effectue un prêt doit avoir souscrit un abonnement (voir figure 7.33).



Par rapport à l'entité personne, la relation effectuer est incluse dans la relation souscrire.

Figure 7.33 : Un exemple d'inclusion de participation.

Contraintes sur la participation de plusieurs entités à plusieurs relations

Ces contraintes permettent d'exprimer des conditions d'existence d'occurrences de relations types selon la présence ou l'absence de participations à d'autres relations types ayant des entités communes dans leur collection. Il s'agit d'une généralisation des contraintes présentées précédemment; l'ensemble de référence n'est plus limité à une entité type, mais à des n-uples d'entités. Nous allons étudier différentes situations pouvant se présenter, comme l'illustre la figure 7.34.

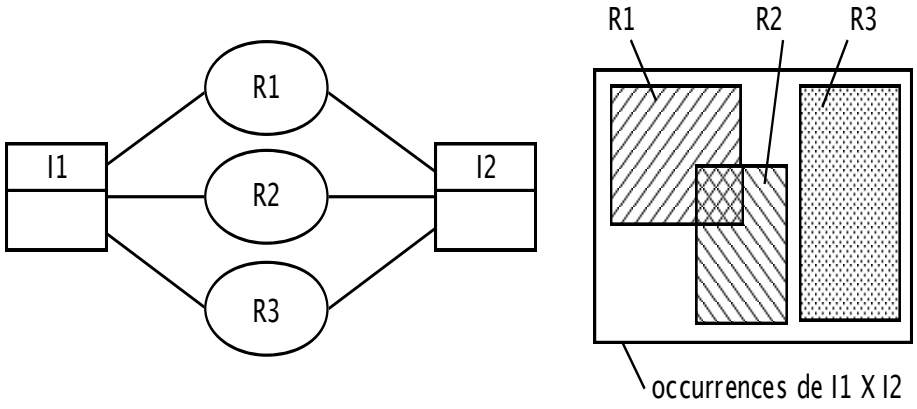


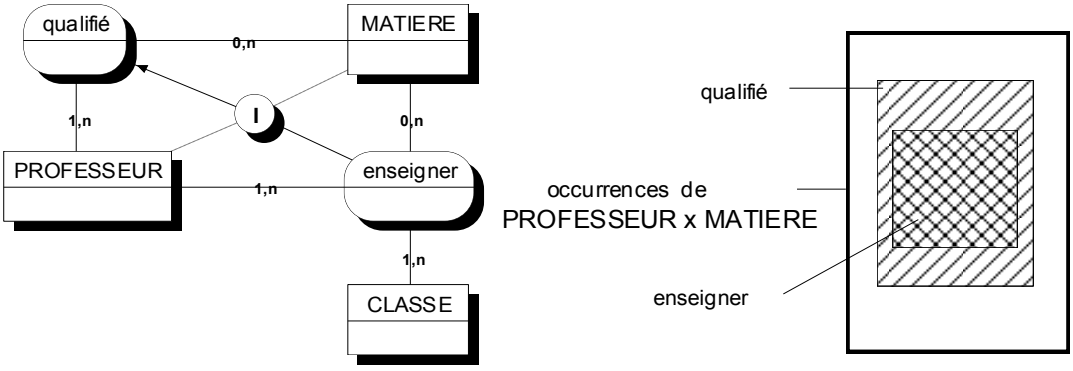
Figure 7.34 : Participation de plusieurs entités à plusieurs relations.

Contraintes d'inclusion de relations sur d'autres relations

Il y a contrainte d'inclusion, relativement à des entités citées, d'une relation R1 dans une relation R2, si la participation des occurrences des entités citées aux

occurrences de la relation R1 implique la participation des occurrences de ces entités aux occurrences de la relation R2. On l'exprime par une contrainte I dont l'émetteur est la relation R1 et la cible la relation R2.

Par exemple, tout professeur qui enseigne une matière à ses classes est qualifié pour cette matière. On a ici une contrainte d'inclusion de la relation enseigner sur la relation qualifier, que l'on modélisera comme sur la figure 7.35.



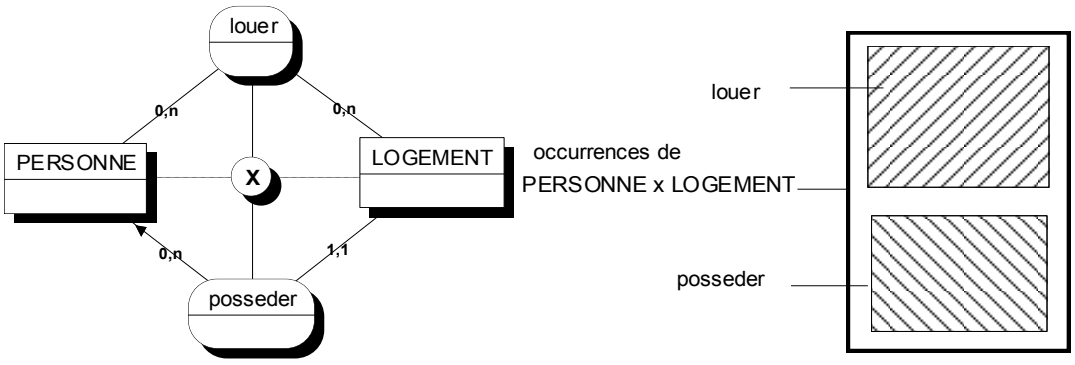
Par rapport aux entités MATIERE et PROFESSEUR, la relation ENSEIGNER est incluse dans la relation QUALIFIER.

Figure 7.35 : Exemple de contrainte d'inclusion interrelations.

Contraintes d'exclusivité de relations sur d'autres relations

Il y a contrainte d'exclusivité, relativement à des entités citées, d'une relation R1 et d'une relation R2, si la participation des occurrences des entités citées aux occurrences de la relation R1 exclut la participation des occurrences de ces entités aux occurrences de la relation R2. On l'exprime par une contrainte X.

Par exemple, une personne ne peut pas être locataire et propriétaire d'un même logement (voir figure 7.36).

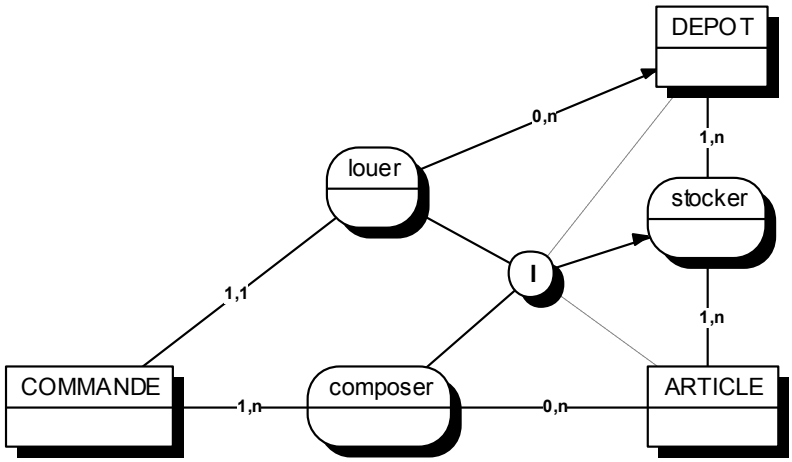


Par rapport aux entités LOGEMENT et PERSONNE, les relations louer et posséder sont mutuellement exclusives.

Figure 7.36 : Un exemple de contrainte d'exclusivté interrelations.

On pourrait également avoir des contraintes sur la participation de plusieurs entités à plusieurs relations de type OU exclusif (contrainte XT), OU inclusif (contrainte T)...

La figure 7.37 présente un exemple plus élaboré de contraintes (cas de la société X).



Par rapport aux entités article et dépôt, quand un article entre dans la composition d'une commande, et que cette commande concerne un dépôt, alors cet article est obligatoirement stocké dans ce dépôt.

Figure 7.37 : Un exemple élaboré de contrainte d'inclusion (cas de la société X).

Contrainte sur les occurrences de relations ayant des entités communes

Identité d'occurrences d'entités impliquées via des relations distinctes (égalité)

Soit une entité I1 reliée indirectement à une entité I2 par des relations binaires fonctionnelles formant des cheminements distincts. Pour toute occurrence de I1, l'occurrence de I2 obtenue par l'un des cheminements est identique à celle obtenue par l'autre cheminement. On l'exprime par une contrainte E.

Par exemple, pour toute occurrence de commande, l'occurrence de client donneur d'ordre de la commande (relation passer) est toujours identique à l'occurrence de client propriétaire (relation posséder) du dépôt auquel doit être livrée (relation livrer à) la commande.

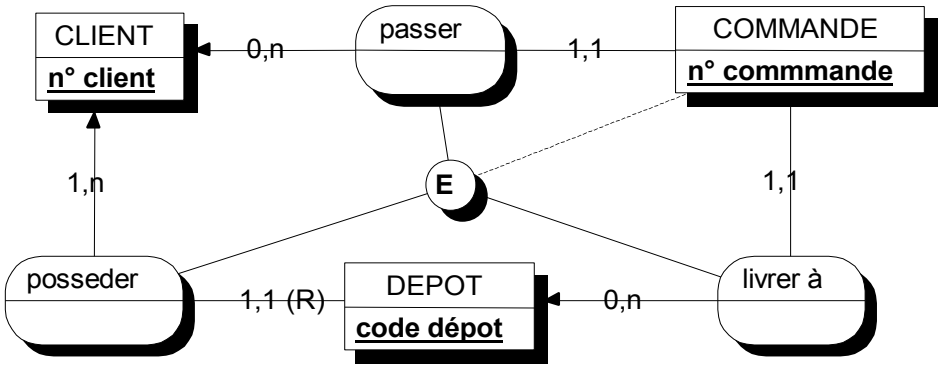


Figure 7.38 : Contrainte d'égalité

Récapitulatif des contraintes inter-relations

Voir tableau de la figure 7.39.

Les exemples peuvent être généralisés à plusieurs entités; la phrase explicative devient : Si (toute) occurrence d'un n-uple des entités I1, I2...participe...

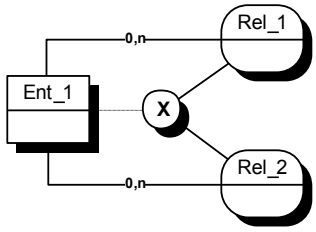
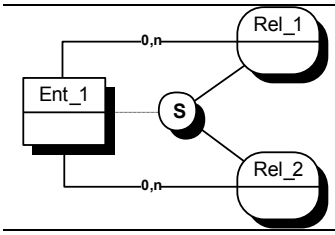
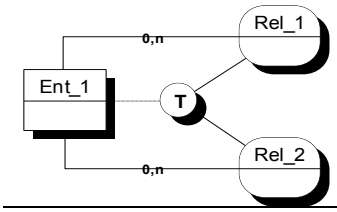
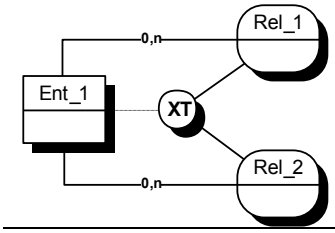
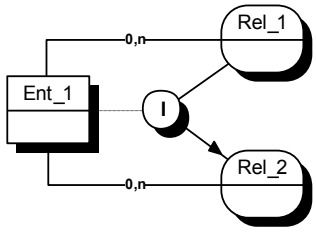
	**EXCLUSIVITÉ** Si une occurrence de l'entité Ent_1 participe à la relation Rel_1, elle ne peut pas participer à la relation Rel_2 et réciproquement. (avec possibilité d'orientation de cette exclusivité)
	**SIMULTANÉITÉ** Toute occurrence de l'entité Ent_1 participant à la relation Rel_1 participe simultanément à la relation Rel_2.
	**TOTALITÉ** Toute occurrence de l'entité Ent_1 participe au moins à l'une des deux relations Rel_1 ou Rel_2.
	**EXCLUSIVITÉ et TOTALITÉ** Toute occurrence de l'entité Ent_1 participe au moins soit à la relation Rel_1, soit à la relation Rel_2, mais pas aux deux à la fois.
	**INCLUSION** Si une occurrence de l'entité Ent_1 participe à la relation Rel_1, elle participe à la relation Rel_2 (mais pas réciproquement).

Figure 7.39 : Tableau récapitulatif des contraintes interrelations.

Contraintes de stabilité

Les contraintes présentées jusqu'ici sont de type statique. Il s'avère nécessaire d'introduire de nouvelles contraintes permettant d'exprimer la stabilité du modèle de données. Ces contraintes concernent principalement :

la « stabilité » des valeurs des propriétés dans le temps,

le rattachement et le détachement d'occurrences d'entités via des occurrences de relations dans le temps.

Ces contraintes s'appliquent en fonctionnement normal, pour l'utilisateur courant du système ou pour un développeur standard d'application client/serveur. On peut toujours envisager qu'un administrateur dûment habilité puisse, sous certaines conditions à préciser, s'affranchir de ces contraintes pour rectifier un dysfonctionnement.

Contraintes de stabilité liées aux propriétés

Propriété stable (S)

Une propriété est dite stable si, étant donné une occurrence de l'entité ou de la relation décrite par cette propriété, la première valeur attribuée à cette propriété ne peut être ultérieurement modifiée.

On conviendra qu'une valeur d'initialisation à NULL, n'est pas une valeur significative. En conséquence, cette valeur pourra être ultérieurement modifiée en valeur significative définitive.

Tout identifiant, même composé, doit, par définition, être stable.

Contraintes de stabilité liées aux relations

Patte de relation définitive (D)

Une patte de relation est définitive si une occurrence de la relation ne peut être supprimée que par et seulement par la suppression simultanée de l'occurrence correspondante de l'entité impliquée dans la patte de relation.

La notation graphique est un (D) porté par la patte concernée (voit figure 7.40).

Rappelons que, par définition, on ne peut pas parler de modification de la collection d'une occurrence d'une relation; il s'agit en fait d'une suppression d'une occurrence de la relation suivie de la création d'une nouvelle occurrence de la relation; la collection d'une relation joue en fait le rôle d'identification de la relation et, comme l'identifiant d'une entité, n'est pas modifiable.

Avec une patte définitive, on ne peut pas supprimer les occurrences de la relation autrement que par la suppression de l'occurrence de l'entité de la patte définitive. En conséquence, on ne peut supprimer une occurrence des autres entités de la collection de la relation (pattes non définitives) tant qu'il reste des

occurrences de la relation où intervient l'occurrence à supprimer.

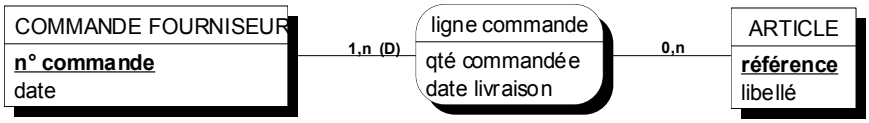


Figure 7.40a : Exemple de patte définitive

Dans une commande fournisseur donnée, on ne peut supprimer une ligne qu'en supprimant simultanément la commande. On ne peut évidemment pas changer d'article (modification de collection, donc supprimer une ligne pour en créer une autre). En conséquence, on ne peut supprimer une occurrence d'article (donc supprimer les occurrences de la relation liées) tant qu'il existe des lignes où intervient cet article; ou alors, supprimer simultanément toutes les commandes qui utilisent cet article...! On peut par contre ajouter quand on veut des lignes de commande à une commande existante.

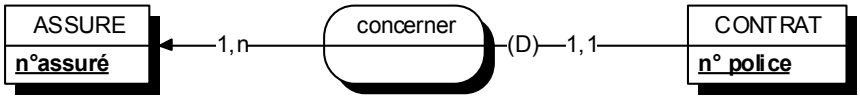


Figure 7.40b : Exemple de patte définitive

Un contrat ne concerne définitivement qu'un et un seul assuré; un contrat ne peut donc, dans le temps, changer d'assuré concerné.

Patte de relation verrouillée (V)

Une patte de relation est verrouillée si, étant donné une occurrence de l'entité reliée par cette patte, toutes les occurrences de la relation dans lesquelles cette occurrence de l'entité intervient, sont créées en même temps que l'occurrence de l'entité. Ultérieurement, on ne peut ni ajouter ni supprimer une occurrence de la relation impliquant cette occurrence d'entité. Cette contrainte lie ainsi la création des occurrences d'une relation à la création des occurrences d'une entité de sa collection.

La notation graphique est un (V) porté par la patte concernée (voit figure 7.41).

Une patte verrouillée est par définition définitive.

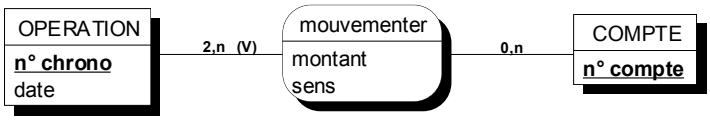


Figure 7.41 : Exemple de patte verrouillée

Cette contrainte est encore plus "stabilisante" que la contrainte de patte définitive. On a rendu totalement dépendant statiquement et dynamiquement l'entité et la relation.

Identifiant relatif

Toute entité type doit être dotée d'un identifiant et nous avons évoqué la difficulté d'inventer de telles propriétés. Certaines entités types ont par ailleurs une existence totalement dépendante d'autres entités types. On peut alors avoir recours à un identifiant relatif.

L'identification relative s'effectue :

- par une propriété stable de cette entité (qui ne remplit pas les conditions d'un identifiant absolu) dite *identifiant relatif*.
- via une relation binaire porteuse d'une dépendance fonctionnelle obligatoire (c'est-à-dire cardinalité 1,1) vers une entité dite de référence; cette patte doit également être définitive.

On note l'identification relative par (R) sur la patte servant de relativité. Dans l'exemple de la figure 7.42, l'identifiant de la ligne de commande est donc n° ligne relatif à commande (ex ligne n° 2 de la commande n° 5432).

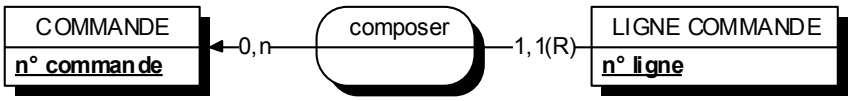


Figure 7.42 : Exemple d'identifiant relatif.

L'identification relative n'est pas un identifiant composé car certaines propriétés constituant l'identification n'appartiennent pas à l'entité ainsi identifiée;il est par ailleurs exclu de dupliquer des propriétés à fin d'identification. Enfin, l'identifiant relatif sera traité de façon particulière au niveau logique.

L'identification relative peut se propager à plusieurs niveaux, à travers des relations types binaires fonctionnelles obligatoires.

L'identification relative peut être multiple et s'exprimer à travers plusieurs relations (voir figure 7.43).

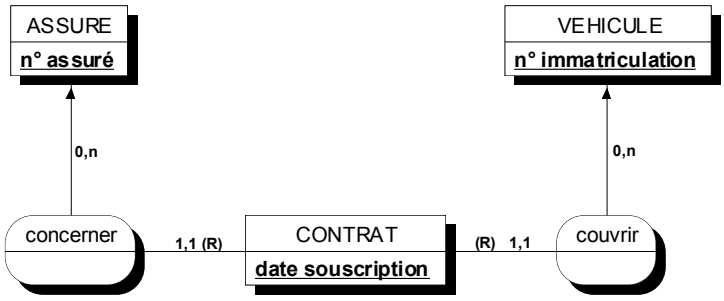


Figure 7.43 : Identification relative multiple.

Dans certains cas d'identification relative multiple, la présence d'identifiant relatif n'est pas obligatoire; l'identification de l'entité s'effectue alors par la composition des identifiants des entités de référence (voir figure 7.44). Dans l'exemple, une telle situation pourrait inciter à modéliser la notion de Dossier comme une relation, surtout si elle n'était pas impliquée elle-même dans une autre relation. Toutefois le caractère d'objet concret d'intérêt pour l'utilisateur du Dossier l'emporte sur la "faiblesse" d'identification; ainsi une telle entité est parfois qualifiée d'*entité faible*.

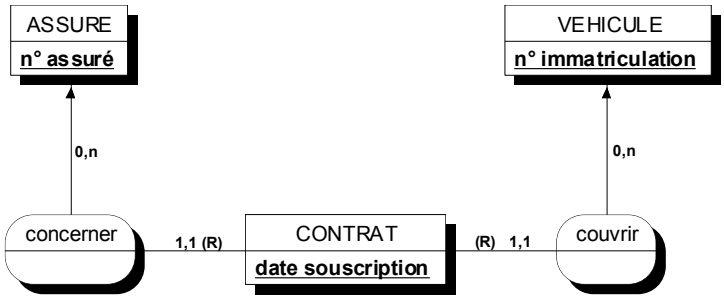


Figure 7.44 : Identification relative multiple et entité faible

Décomposition d'une relation type

Si les relations binaires sont effectivement très nombreuses dans les modèles conceptuels de données (dues à la propension naturelle des concepteurs à simplifier leur perception), l'usage de relations n-aires s'avère parfois nécessaire.

Cependant, lorsque des dépendances fonctionnelles sont définies sur une relation n-aire, une décomposition de cette relation en plusieurs relations de dimension moindre est envisageable. Cette décomposition présente deux intérêts; elle permet d'une part de mieux matérialiser certaines des dépendances fonctionnelles et d'autre part, en réduisant la dimension des relations, de les rendre plus facilement interprétables.

La décomposition de relations n-aires dotées de dépendances fonctionnelles répond à une condition préalable :

Le nombre d'entités concernées par la dépendance fonctionnelle doit être inférieur à la dimension de la relation.

L'opération de décomposition de la relation consiste à :

Modéliser une relation dont la collection comprend les entités types impliquées dans la dépendance fonctionnelle; cette relation est porteuse de la dépendance fonctionnelle.

Sortir de la collection initiale de la relation l'entité cible de la dépendance fonctionnelle (familièrement « couper la patte » de cette entité).

Affecter les cardinalités de la nouvelle relation :

- L'entité cible (sortie de la collection initiale) conserve les cardinalités qu'elle avait dans la relation initiale.
- Si la relation est binaire, la cardinalité mini de l'entité émettrice est celle qu'elle avait dans la relation initiale; la cardinalité maxi est 1.
- Si la relation est de dimension supérieure à 2, les cardinalités des entités émettrices sont celles qu'elles avaient dans la relation initiale.

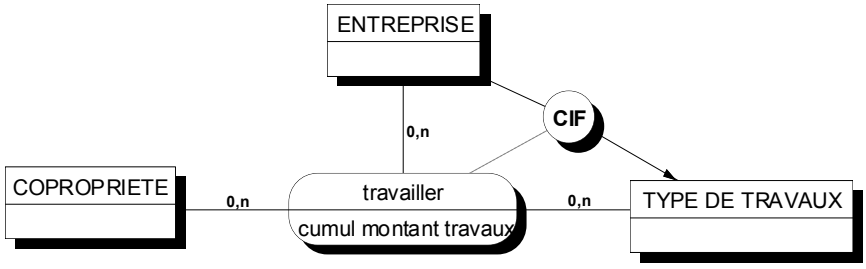
Les éventuelles propriétés restent rattachées à la relation initiale.

Outre le principe général de décomposition, ci-dessus présenté, il faut préciser certaines situations particulières :

- Si les entités émettrices de la dépendance fonctionnelle ont une cardinalité mini égale à 1, alors la décomposition est systématique
- Si les entités émettrices de la dépendance fonctionnelle ont une cardinalité mini égale à 0, alors le concepteur ne procédera à la décomposition que s'il s'est assuré que les deux relations types issues de la décomposition ont une existence liée [Tabourier 86].
- Si une autre relation type, porteuse d'une dépendance fonctionnelle, partage une partie de la collection de la relation type à décomposer, le concepteur doit, auparavant, s'assurer que la relation type issue de la décomposition et porteuse d'une dépendance fonctionnelle est ou non assimilable à la relation type déjà exprimée.

Les figures 7.45 à 7.47 présentent différents exemples de décompositions.

Avant décomposition :



Dépendance fonctionnelle : dans le cadre de la relation travailler, chaque entreprise n'effectue qu'un type de travaux.

Après décomposition :

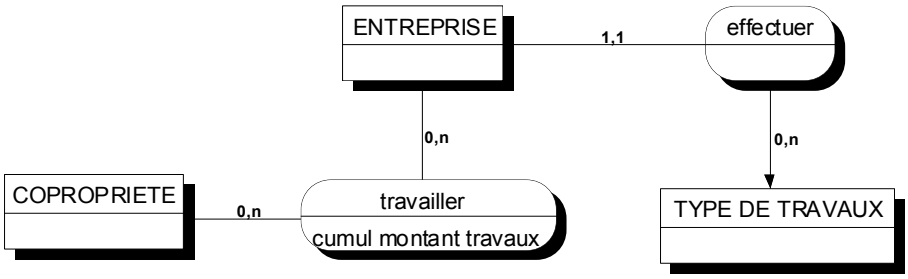
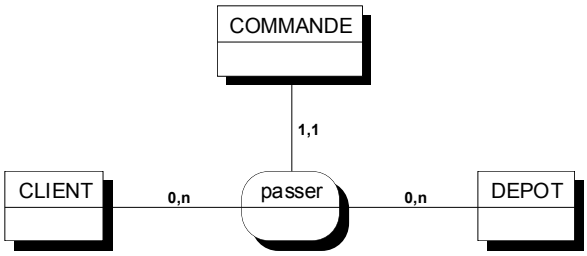


Figure 7.45 : Exemple de décomposition par dépendance fonctionnelle.

Une cardinalité maxi à 1 .dans une relation n-aire implique, par définition, des dépendances fonctionnelles.

Avant décomposition :



Après décomposition :

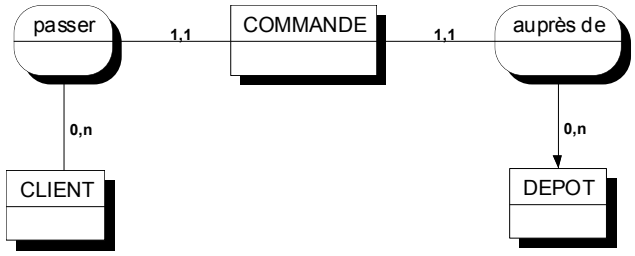
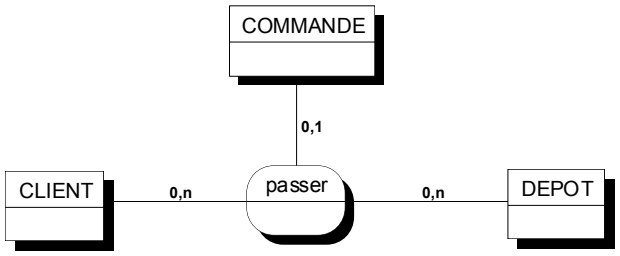


Figure 7.46 : Exemple de décomposition par cardinalité (1,1).

Dans le cas d'une cardinalité (0,1) ., il est nécessaire d'ajouter une contrainte de simultanéité S (figure 7.47 a)

Avant décomposition :



Après décomposition :

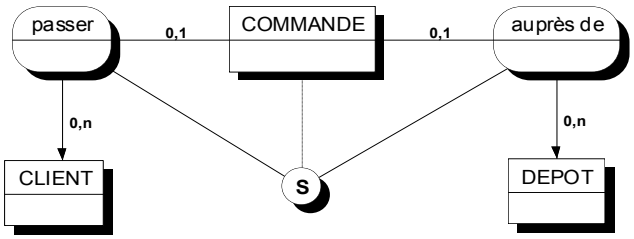


Figure 7. 47a : Exemple de décomposition par cardinalité (0,1).

Prenons enfin l'exemple d'une décomposition complexe où l'on procèdera par étapes.

Au départ, un contrat d'assurance souscrit par un assuré concerne, en cas de risque auto, un assuré et un véhicule, ou seulement un assuré en cas d'autres risques (figure 7.47 b):

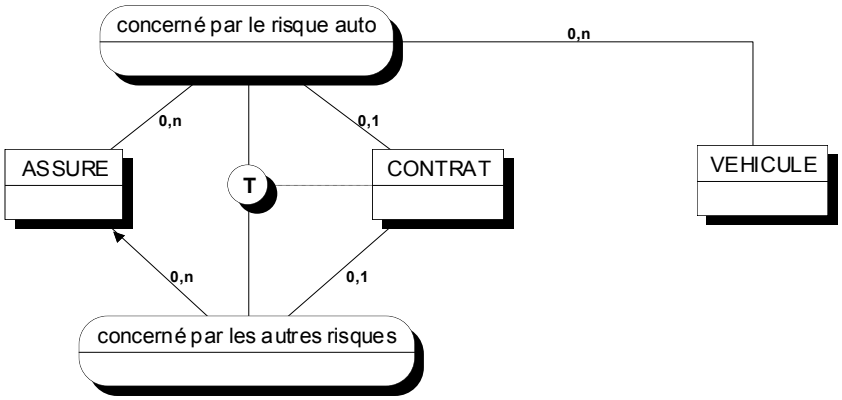


Figure 7.47 b : Exemple complexe à décomposer

Étape 1 : on va décomposer la relation ternaire disposant d'une cardinalité (0,1).

Cette relation ternaire se décompose en deux relations binaires reliées par une contrainte de simultanéité. On choisit de conserver le nom initial pour la relation avec ASSURE, et l'on renomme COUVRIR la relation avec VEHICULE (Figure 7.47 c).

La contrainte de totalité T se reporte sur les relations décomposées.

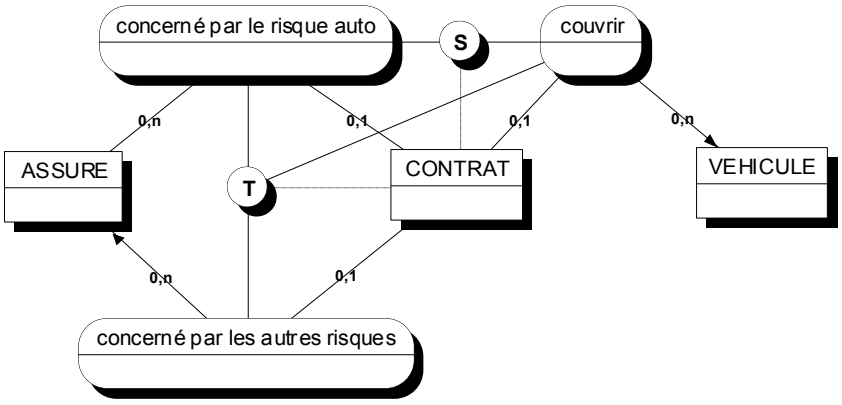


Figure 7.47 c : Exemple de décomposition complexe (étape 1)

Étape 2 : on va fusionner les relations « concerné par le risque auto » et « concerné par les autres risques » munies d'une contrainte de totalité T.

Ces deux relations ont la même collection et des significations pouvant être considérées comme similaires. On fusionne ces deux relations sous le nom de CONCERNER dont la cardinalité par rapport à CONTRAT devient (1,1) — la cardinalité mini 0 se transforme en 1, la contrainte de totalité T disparaît

(Figure 7.47 d).

La sémantique initiale est conservée. Tout contrat est concerné par un assuré et un seul, certains couvrent un véhicule :

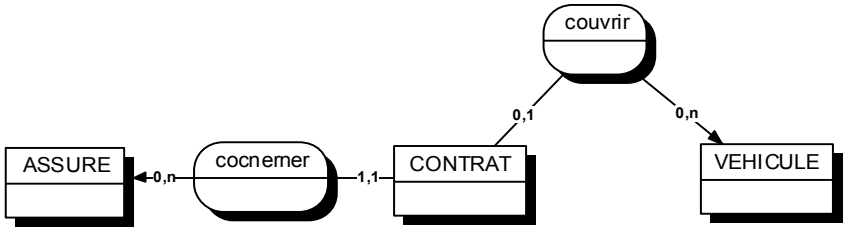


Figure 7.47 c : Exemple de décomposition complexe (étape 2)

Compléments sur le formalisme entité-relation

Les notions ci-après comprennent d'une part des enrichissements introduits progressivement dans le formalisme entité - relation, d'autre part des compléments et réflexions issus de la pratique.

Règles

Jusqu'à présent, dans la méthode Merise, les règles de traitement étaient usuellement rattachées aux traitements où elles s'effectuaient. Cette description des règles dans le cadre des traitements était largement influencée par la situation d'une informatisation où les règles étaient, en général, mises en oeuvre dans les programmes. Depuis ces dernières années, et particulièrement avec le développement du client/serveur, on assiste progressivement à la définition et à la mise en oeuvre de certaines règles au niveau des données.

D'ores et déjà, certaines contraintes sont définies au niveau du modèle conceptuel de données : cardinalités, contraintes inter-relations, contraintes de stabilité, historisation. Ces contraintes ou règles ont d'une part la même permanence que les entités, relations et propriétés du MCD, et d'autre part sont indépendantes du contexte d'utilisation par tel ou tel traitement.

On peut distinguer deux catégories de règles :

- Les règles de traitement qui restent spécifiques et contingentes à une ou certaines activités, c'est à dire conséquentes aux événements qui déclenchent l'activité (opération, tâche, ULT). Ces règles sont formalisées au niveau de la modélisation des traitements (organisationnel ou logique).
- Les règles de traitement qui sont indépendantes de toute activité

(opération, tâche, ULT) spécifique et qui doivent être appliquées au niveau des données pour maintenir une cohérence globale..

Dans tous les cas, une règle s'exprime sur des données qui interviennent comme termes de l'algorithmique. Aussi, nous proposons de définir les règles au niveau du modèle conceptuel de données afin de garantir la cohérence entre l'expression de la règle et les données impliquées. Bien que définies au niveau des données, ces règles pourront toujours être utilisées au niveau des traitements (cf. Chap.8).

L'expression des règles de traitement dans un modèle conceptuel de données comporte :

- le nom de la règle,
- la description de l'algorithmique (sous la forme de français structuré ou pseudo-code),
- les entités, relations et propriétés utilisées par la règle.

La figure 7.48 illustre la représentation graphique d'une règle (un rectangle à coin corné) et des propriétés utilisées par la règle (des pointillés) [Morejon 94]

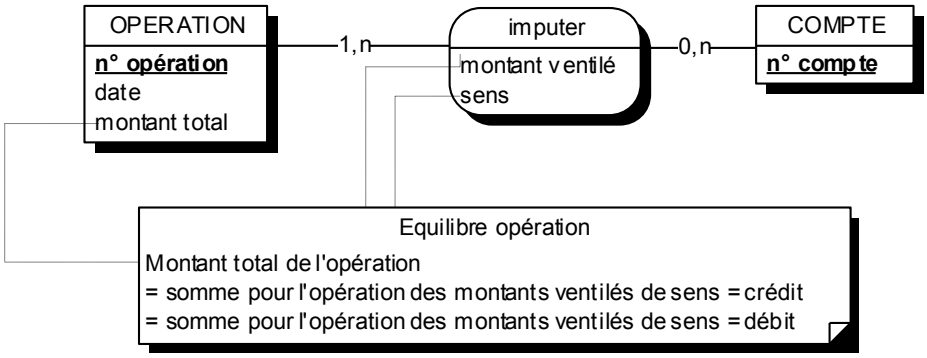


Figure 7.48 Modélisation d'une règle

Modélisation du temps

Le modèle conceptuel de données est, par nature, statique. Or, le concepteur désire parfois intégrer des informations liées au temps. Cette apparente antinomie entre temps et statique pose des problèmes à certains concepteurs.

Propriété à valeurs calendaires

De nombreuses propriétés se réfèrent au temps, par exemple : date de livraison, mois d'échéance, année d'exercice.

Dans ce cas-là, ces propriétés suivent exactement les mêmes règles que les autres propriétés. Il n'y a aucune justification, au niveau conceptuel, à les

modéliser d'une façon particulière (voir figure 7.49).

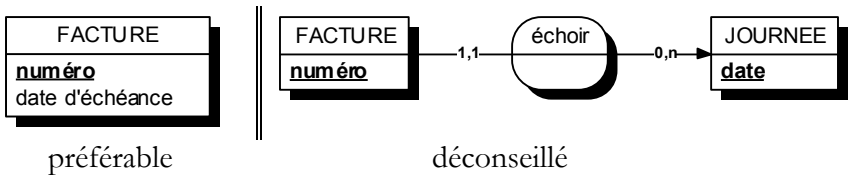


Figure 7.49: Modélisation du temps; valeurs calendaires.

Modélisation de la chronique des valeurs d'une propriété

Parfois, le concepteur est confronté à la modélisation de propriétés présentant une série chronologique de valeurs : chiffre d'affaire mensuel, production quotidienne. Dans un tel cas, le temps devient une véritable dimension qu'il convient donc de modéliser explicitement comme une entité (voir figure 7.50).

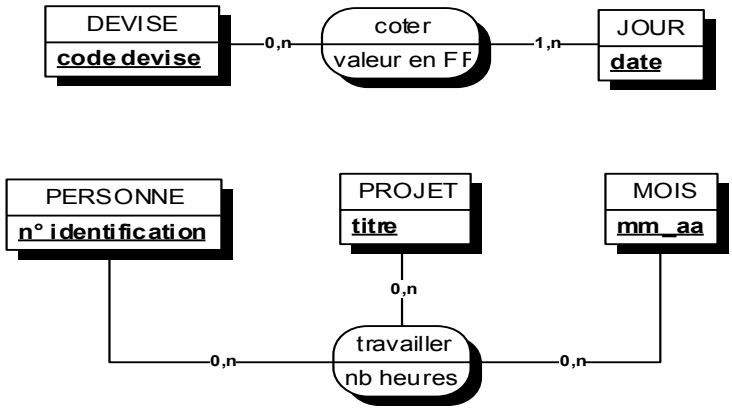


Figure 7.50 : Modélisation du temps; séries chronologiques
Cotation quotidienne d'une devise
Activité mensuelle d'une personne sur un projet

Il faut réserver cette modélisation aux séries chronologiques, c'est dire les cas où les valeurs de la propriété sont régulièrement échelonnées dans le temps. On utilise fréquemment cette modélisation explicite d'une entité temporelle dans les domaines de suivi d'activités

Historisation

Pour certaines propriétés, ou pour l'ensemble des propriétés d'une entité ou d'une relation le concepteur désire parfois conserver, en cas de modification, les valeurs antérieures prises par ces propriétés, pour la même occurrence de l'entité ou de la relation de rattachement; c'est à dire « historiser » les valeurs

des propriétés. Dans ce cas-là, la règle de vérification (non-répétitivité des valeurs) serait théoriquement enfreinte et ces propriétés ne devraient pas être ainsi modélisées (elle est certes vérifiée pour la valeur présente, mais les valeurs antérieures créent une multiplicité). Par ailleurs, on ne peut pas véritablement considérer la liste des valeurs successives comme une série chronologique par manque de régularité dans l'échelle de temps.

Les évolutions du formalisme proposent donc d'indiquer explicitement le caractère *historisable* au niveau d'une propriété, d'une entité ou d'une relation.

Datation et profondeur

L'historisation de valeurs antérieures est précisée par deux caractéristiques

- La *datation* indique l'instant auquel chaque valeur antérieure a été historisée; elle s'assimile à un compostage des valeurs historisées. La précision de la datation peut s'exprimer dans différentes unités de temps : jour-heure, jour, semaine, mois, année. Si deux changements de valeurs interviennent dans la même unité de datation, alors il n'y pas d'historisation mais simple modification de la valeur courante.
- *Exemple* : On choisit d'historiser les valeurs de la propriété adresse d'une personne avec une datation du mois. Une seconde modification de l'adresse dans le même mois ne provoque pas d'historisation de la première mais est considérée comme une correction de la valeur précédente.
- La *profondeur* d'un historique indique le nombre de valeurs antérieures que l'on souhaite conserver; cette profondeur peut être illimitée.

L'indication du caractère historisable peut rester facultative en étude préalable mais devient impérative en étude détaillée. Datation et profondeur ne sont généralement précisées qu'au niveau de la modélisation organisationnelle des données (voir Quantification)

Propriété historisée

La conservation des valeurs antérieures (historisation) ne s'applique qu'à certaines propriétés d'une entité ou d'une relation. Graphiquement, on indique alors le caractère historisable par un (H) au niveau de la propriété (voir figure 7.51).

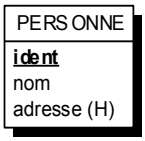


Figure 7.51 : Propriété historisée

Entité historisée

Pour toute modification de valeur de l'une des propriétés d'une entité, on historise l'ensemble des valeurs des propriétés de l'entité. Graphiquement, on indique alors le caractère historisable par un (H) au niveau de l'entité (voir figure 7.52).

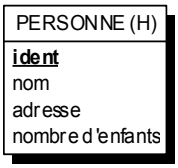


Figure 7.52 : Entité historisée

Relation historisée

Pour toute modification de valeur de l'une des propriétés d'une relation, on historise l'ensemble des valeurs des propriétés de la relation. Graphiquement, on indique alors le caractère historisable par un (H) au niveau de la relation (voir figure 7.53).

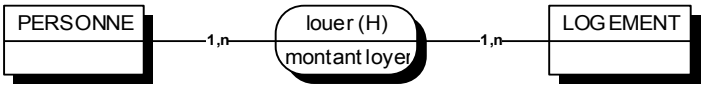


Figure 7.53 : Relation historisée

Patte de relation historisée

Pour tout changement d'occurrence de l'entité constituant la patte d'une relation, on historise la valeur antérieure de l'identifiant de l'entité concernée. Graphiquement, on indique le caractère historisable par un (H) au niveau de la patte de la relation (voir figure 7.54).

D'une part, un assuré peut, être présent dans plusieurs dossiers, d'autre part, on conserve les changements de dossiers successifs de l'assuré.

Figure 7.54 : Patte de relation historisée

Il ne faut pas confondre historisation et archivage. L'historisation concerne l'évolution des valeurs des informations d'une même occurrence. L'archivage aborde la question de la durée de vie des occurrences dans la mémoire immédiate.

Ainsi, à une échéance fixée, on archivera des données qui, en règle générale, ne sont jamais concernées par une historisation.

Liste variable de propriétés

Une règle appliquée aux entités types préconise que toutes les propriétés d'une entité type aient une signification pour toutes les occurrences. Or, dans quelques cas, certaines propriétés ne peuvent suivre cette règle. Pour tenter de résoudre ce problème, nous proposons les solutions suivantes.

Accepter que ces propriétés n'aient pas de pertinence pour certaines occurrences de l'entité.

Cette solution n'est praticable que si le nombre de propriétés impliquées est limité. D'autre part, ce problème peut avoir des répercutions au niveau physique.

La figure 7.55 illustre cette solution. L'entité SALARIE présente la propriété « nom marital », qui n'a de sens que pour la population des salariées féminines.

On l'utilise si le nombre de propriétés concernées est limité, et l'entité stable et d'intérêt pour le domaine.

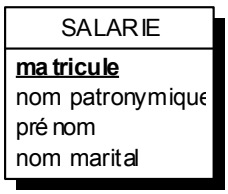


Figure 7.55: Liste variable de propriétés (conservation dans l'entité).

Expliciter les sous-populations évoquées par ces propriétés : modélisation en termes de sous-types.

Cette solution consiste à modéliser autant de sous-types que de sous-populations utilisant de façon pertinente les propriétés concernées (voir figure 7.56).

Remarque : On reconnaît ici la notion de spécialisation..

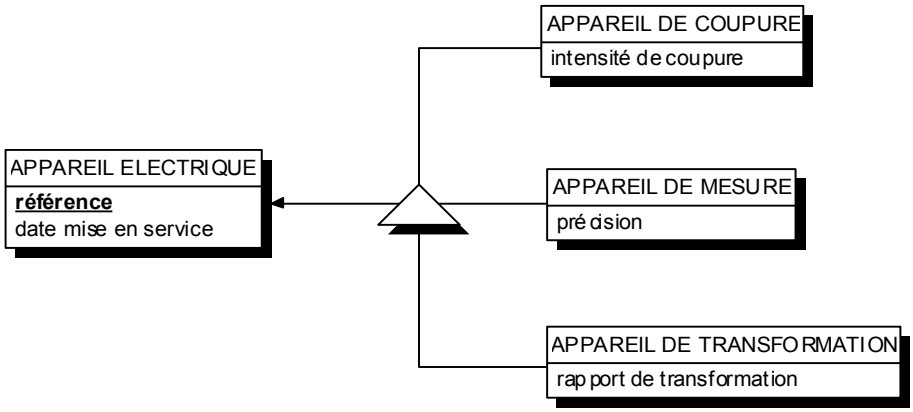


Figure 7.56 : Liste variable de propriétés (modélisation avec sous-types).

On utilise cette modélisation si les sous-populations sont stables et en nombre limité, et si les propriétés descriptives des sous-types sont stables.

Modéliser les propriétés en tant qu'occurrences d'entités types : métamodélisation.

Cette solution, appelée également métamodélisation, consiste à exprimer l'un des types de concept (ici la propriété) dans son propre formalisme (voir figure 7.57).

On modélise ainsi une entité type appelé par exemple Caractère, dont les occurrences sont les propriétés concernées; on établit une relation type entre cette entité type Caractère et l'entité type initiale que l'on a par ailleurs dépouillé des propriétés concernées. Les valeurs prises par les propriétés initiales de l'entité sont désormais des valeurs d'une propriété appelée valeur et rattachée à la relation.

On utilise cette modélisation en cas de grande variabilité de propriétés, de sous-populations évolutives.

Ce type de modélisation apporte une très grande flexibilité mais peut avoir par ailleurs des inconvénients. Aussi convient-il de ne l'utiliser qu'en maîtrisant la modélisation.

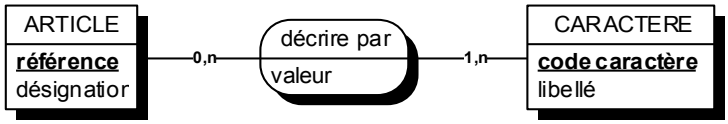


Figure 7.57 : Liste variable de propriétés; métamodélisation.

Propriétés à valeurs codées

Il s'agit de propriétés dont la valeur s'exprime par un code (numérique ou alphanumérique, mnémonique ou non) associé à un libellé explicatif.

Exemples :

Civilité : 1 = Monsieur
2 = Madame
3 = Mademoiselle

Type de véhicule CY = cyclomoteur
MT = moto
VL = véhicule léger
PL = poids lourd
RQ = remorque

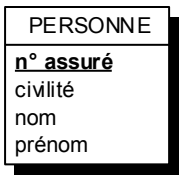
En règle générale, au niveau conceptuel, il est recommandé de modéliser ce type de propriété comme les autres propriétés classiques, c'est à dire de les rattacher à leur entité naturelle (sans tenir compte de l'aspect code + libellé). On peut si nécessaire recourir à un *type utilisateur* spécifique.

Par contre, on déconseille formellement, au niveau conceptuel, de modéliser ce type de propriété par une entité à part, ne comprenant que le code et le libellé, reliée par une relation binaire fonctionnelle à son entité naturelle d'origine. Une telle modélisation conduit

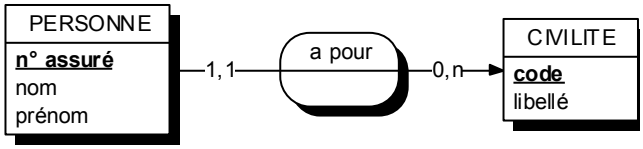
- à créer des entités dont la pertinence et la consistance sont contestables,
- à vider de leur substance les entités d'origine,
- à démultiplier le nombre d'entités et de relations parasitant la lisibilité du modèle au détriment d'entités plus pertinentes.

On doit toutefois recourir à la modélisation sous forme d'entité propre lorsque la notion exprimant la propriété à valeur codée (généralement un type de…) intervient de façon autonome, en particulier dans des relations avec d'autres entités.

Ces propriétés à valeurs codées seront spécifiquement prises en compte lors du passage au modèle logique de données.



conseillé



déconseillé

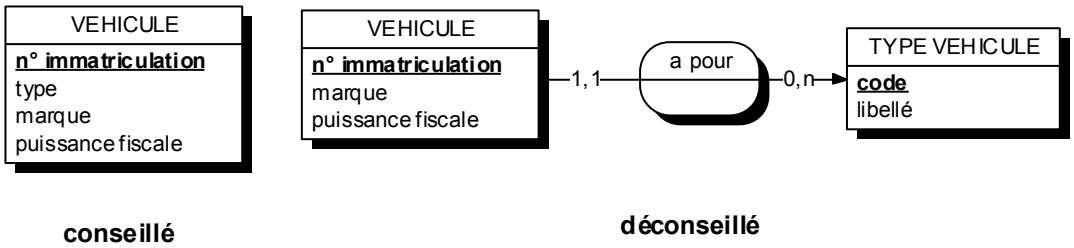


Figure 7.58 : Modélisations des propriétés à valeurs codées

Construction d'un modèle conceptuel de données

Rappelons que le concepteur procédera deux fois à l'élaboration d'un modèle conceptuel de données : lors de l'étude préalable, puis lors de l'étude détaillée. Il est certain que, dans le cadre d'une étude préalable, le concepteur accordera plus d'importance à la structure du modèle (entités et relations) qu'aux propriétés.

La construction d'un modèle conceptuel de données ne s'effectue jamais d'un seul trait. C'est un processus itératif, d'enrichissement progressif, qui trouve souvent son origine dans une alternance avec l'étude des traitements.

Dans tous les cas, on retrouvera ces principes généraux de construction :

Chercher d'abord à modéliser les entités types qui apparaissent le plus naturellement, puis s'intéresser aux relations.

Dès que l'on modélise une entité type, chercher à lui affecter un identifiant, ou du moins l'illustrer par des exemples d'occurrences.

Éviter absolument de réfléchir en termes de fonctionnement (ou traitement); s'astreindre à exprimer des « faits ».

A chaque propriété affectée à une entité type ou à une relation type, s'assurer immédiatement de la règle de vérification (non-répétitivité).

S'assurer que toutes les entités types participent au moins à une relation type.

Préciser les cardinalités mini et maxi de chaque entité dans chaque relation.

Rechercher, au sein de chaque relation type, les éventuelles dépendances fonctionnelles; procéder, si les conditions le permettent, à la décomposition.

Exprimer les éventuelles contraintes inter--relations.

Exprimer les éventuelles règles de traitement.

Régulièrement, relire globalement le modèle afin de vérifier si les futures utilisations envisagées sont prises en compte.

La modélisation conceptuelle des données a pour objectif premier d'exprimer et formaliser les concepts métier perçus dans le discours du domaine étudié, dans un formalisme qui permettra ultérieurement une traduction vers les structures de bases de données. C'est le respect de cet objectif qui préservera son pouvoir d'expression et de communication avec les interlocuteurs du métier.

On ne peut que regretter une pratique constatée sur le terrain qui tend à transformer le MCD en une expression de la future base de données formalisée en entité-relation. Cette pratique conduit à introduire au niveau conceptuel, des modélisations, des solutions et des choix qui n'ont leur place qu'au niveau logique, détournant ainsi le MCD de son objectif premier.

Expression d'un modèle conceptuel de données

Bien que des présentations puissent être adaptées au contexte d'utilisation, évoquons les principaux éléments composant la présentation d'un modèle conceptuel de données :

- Représentation graphique du modèle (schémas). Notons que la lecture des modèles peut être facilitée par la présentation en plusieurs schémas partiels (par thème, avec propriétés, avec contraintes...) et un schéma intégral limité aux éléments structurels.
- Pour chaque entité, description (succincte ou détaillée suivant le niveau d'étude) :
- Pour chaque relation, description :
- de la collection des entités,
- des cardinalités,
- des propriétés affectées,
- des éventuelles dépendances fonctionnelles et contraintes logiques interrelations,
- de l'éventuelle historisation.
- Éventuellement, pour chaque propriété, description de sa signification.
- Pour chaque règle, sa description et les propriétés utilisées.

Comment communiquer à partir d'un modèle conceptuel de données

Dans la modélisation d'un système d'information existant, comme en conception d'un futur système, le modèle conceptuel des données constitue un outil de synthèse précieux pour l'équipe de projet.

On peut aussi souhaiter l'utiliser comme outil de communication avec d'autres intervenants: le responsable utilisateur, l'utilisateur final, ou l'administrateur de bases de données. Comment, concrètement, présenter un MCD en dehors de l'équipe de projet ? Sous quelles conditions ?

Un modèle conceptuel de données est un document technique, souvent difficile à comprendre pour des personnes non formées. Pour un échange bénéfique avec vos interlocuteurs, en particulier les utilisateurs, il faut d'abord expliquer le but du document, puis les notions représentées, en procédant du plus simple au plus complexe, et en le reliant à ce qu'ils connaissent (attention ! Il ne s'agit pas de faire un cours sur le formalisme).

On parviendra à une bonne communication en associant trois descriptions cohérentes complémentaires :

- Une définition et un exemple de chaque notion élémentaire (il s'agit de définir entité ou relation par un texte court et des exemples simples).
- Des schémas très lisibles du MCD réduits à l'essentiel (rapprocher les notions dont les significations sont voisines et limiter le nombre de croisements des pattes du modèle, faire des présentations de schémas partiels où ne figurent que peu de concepts).
- Un texte de liaison en français (il permet de parcourir l'ensemble du MCD selon un scénario raisonnable).

La conjonction du dessin et du texte favorise une bonne compréhension du MCD qui constituera ainsi un outil de communication efficace.

8

Modélisation organisationnelle des traitements

Problématique du modèle organisationnel de traitements (MOT)

Définition de solutions d'organisation

Le modèle conceptuel de traitements a permis de décrire les activités majeures du domaine, sans référence aux ressources nécessaires pour en assurer le fonctionnement; on s'est concentré sur le *quoi* et le *pourquoi*. La construction du modèle organisationnel de traitements se concentre sur le *comment* et va consister à :

- Définir les différentes ressources à mettre en œuvre (ce terme ressource est très général et concerne aussi bien des moyens techniques ou humains, de l'espace, du temps et des données).
- Décomposer les opérations spécifiées au niveau conceptuel en des éléments plus fins et homogènes : les tâches.
- Construire un enchaînement chronologique des activités.
- Organiser l'ensemble des ressources permettant d'assurer l'exécution des tâches envisagées.

En résumé, il s'agit de spécifier avec plus de détails le contenu de chaque opération conceptuelle dont l'expression des tâches était jusqu'ici très sommaire et de construire une ou plusieurs solutions d'organisation.

A la différence des modèles conceptuels (données ou traitements), l'élaboration d'un modèle organisationnel ne présente pas de difficulté théorique, liée à l'effort d'abstraction. Par contre l'extrême diversité des solutions d'organisation envisageables ou le niveau de détail nécessaire rendent cette phase parfois délicate.

La construction d'un MOT nécessite un important effort de la part de l'équipe de projet, pour plusieurs raisons.

Une solution d'organisation doit préciser au minimum :

- l'organisation prévue pour les utilisateurs, avec les différents postes de travail et/ou services;
- la circulation des informations entre ces centres d'activités;
- dans les postes de travail, les différentes tâches à réaliser et selon quelle chronologie.

Le niveau de détail de cette description doit tenir compte de l'étape en cours (étude préalable ou étude détaillée), mais il doit toujours permettre une compréhension immédiate pour un futur utilisateur.

Il y a en général plusieurs solutions possibles; chaque solution doit être décrite comme précédemment (variantes).

Chaque solution d'organisation doit aussi être évaluée selon quatre types de critères :

- critères économiques (La solution à évaluer est-elle plus efficiente que la solution actuelle ? Les coûts et les délais associés restent-ils acceptables pour l'organisation ?).
- critères techniques (La variante à évaluer est-elle réalisable avec les technologies utilisées dans l'entreprise ou disponibles sur le marché ?);
- critères ergonomiques (Les tâches prévues sont-elles bien adaptées aux futurs utilisateurs ?);
- critères d'ordre social (La solution envisagée est-elle conforme aux orientations de l'entreprise en matière de personnel ? Quel serait son impact en matière d'emploi et de qualifications ? Cette solution est-elle acceptable par les futurs utilisateurs ?);

La construction des variantes du MOT pose ainsi tout le problème du changement dans l'entreprise. Ce n'est pas un problème purement technique; il est donc important, en respectant les choix de l'entreprise, de favoriser une approche ouverte de ce changement, approche sociotechnique et participative.

Répartir un système d'information

Cette problématique de l'organisation et du changement suppose un minimum de complexité du système d'information : les données, les traitements et les ressources ne sont pas tous réunis en un lieu géographique unique; ils sont au contraire répartis ou à répartir de nouveau.

La distinction faite entre SIO (système d'information organisationnel) et SII (système d'information informatisé) conduit à une double répartition selon la figure 8.1 :

- Une répartition entre des unités organisationnelles de l'entreprise : dans quels établissements, services ou postes de travail sont exercées les activités de l'entreprise ? Avec quelles ressources humaines et techniques ?
- Une répartition entre des sites informatiques : où sont localisés les différents matériels utilisés ? Par quels réseaux sont-ils reliés ? Quels fichiers ou bases de données supportent-ils ? Où sont les ressources technologiques ? (voir, en troisième partie, les chapitres 12 et 13).

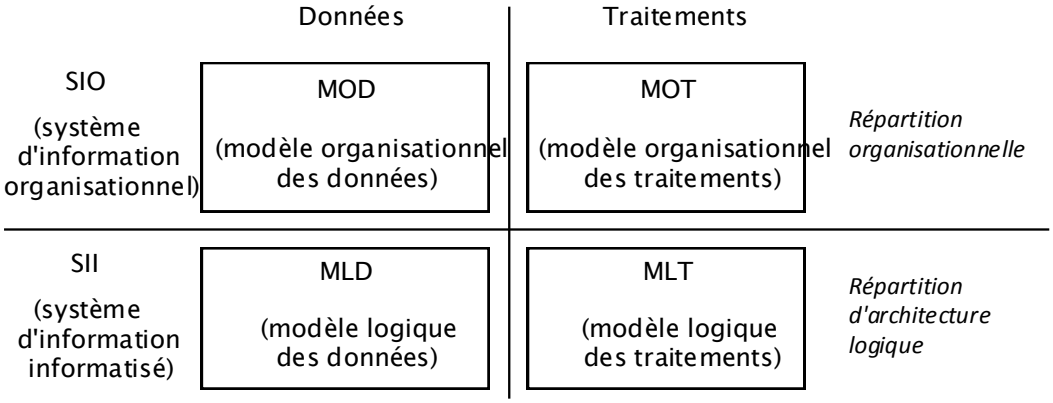


Figure 8.1 : Différents niveaux de répartition.

La répartition organisationnelle est visible pour les utilisateurs de l'entreprise. La répartition technologique peut n'être visible que pour les informaticiens.

Répartition organisationnelle

Nous traiterons dans ce chapitre la **répartition organisationnelle des traitements** à l'aide des modèles organisationnels de traitements ou MOT (pour les données mémorisées, voir au chapitre 10 les modèles organisationnels de données ou MOD).

Cette répartition mettra d'abord en évidence des centres d'activité organisationnels, les postes de travail, dans lesquels seront réalisés les des différentes activités.

Un poste de travail mobilise des ressources humaines et des ressources informatiques; il faudra donc préciser pour chaque poste de travail et chaque tâche la part assurée par l'homme et celle assurée par la machine.

Formalisme de modélisation des traitements au niveau organisationnel

Malgré son importance en matière de conception de système d'information organisationnel, le MOT ne nécessite pas de formalisme spécifique; il reprend très largement les concepts du MCT, parfois réadaptés et enrichis, auxquels sont ajoutés de nouveaux concepts, dont celui de poste de travail.

Le poste de travail

Le poste de travail type, ou poste type, constitue l'une des principales dimensions du modèle organisationnel. Un poste type est un centre d'activité

élémentaire du domaine comprenant tout ce qui est nécessaire à l'exécution de traitements. Pour spécifier un poste type, on décrit :

Les compétences et aptitudes requises par les personnes intervenant dans ce poste type.

Les caractéristiques techniques des matériels associés à ce poste. Bien qu'aucune restriction n'existe, le matériel informatique retiendra davantage l'attention du concepteur. Notons qu'à ce niveau il s'agit de description très générale sans rentrer dans des spécifications techniques de tel ou tel matériel.

L'aménagement général du poste et sa localisation dans l'espace.

Un poste type peut, selon les cas, comprendre :

- Une personne associée à un matériel; par exemple pour le poste secrétariat d'admissions d'une clinique, 1 secrétaire médicale, 1 clavier et écran.
- Plusieurs personnes partageant un matériel; par exemple le poste réception comptoir d'un magasin, 3 vendeurs, 1 clavier, 1 écran, 1 lecteur marques magnétiques (code à barres), 1 imprimante.
- Une ou plusieurs personnes sans matériel; par exemple le poste aire de stockage d'un magasin : 5 manutentionnaires.
- Du matériel sans personnel spécialisé; par exemple le poste « Lecteur de badge horaire flexible ».

Un poste type peut se matérialiser par plusieurs occurrences sur le terrain qui ont, par définition, le même comportement. Par exemple, le poste type secrétariat d'admissions est en trois exemplaires.

La détermination et le choix d'un poste doivent avant tout tenir compte des compétences des personnes affectées : un poste traduit un niveau de responsabilité des personnes qui conditionnera les tâches qui leur seront confiées. Le découpage en postes exprime en partie l'organisation des structures de décision du système de pilotage. La description des compétences et responsabilités d'un poste se traduit parfois par le *profil* d'un poste.

Les postes sont formalisés graphiquement sous la forme de colonnes dans lesquelles sont représentées les tâches réalisées au sein des postes. Selon différents usages, des colonnes complémentaires peuvent accueillir:

les acteurs externes et leurs échanges (événements et résultats) avec les postes.

l'échelle de temps de la chronologie des tâches.

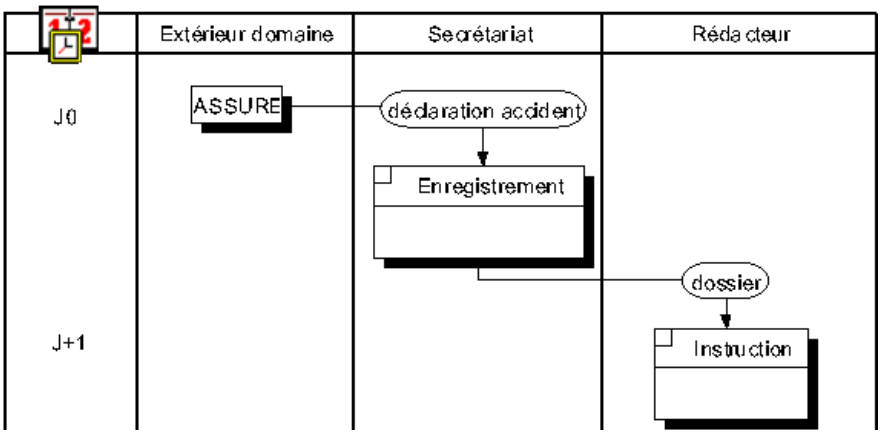


Figure 8.2 : Représentation des postes

L'événement/résultat-message

Les concepts d'événement et de résultat (type et occurrence) sont les mêmes qu'au niveau conceptuel. Par ailleurs, tous les événements et résultats *externes* décrits dans le modèle conceptuel de traitements doivent se retrouver dans le modèle organisationnel, éventuellement sous une forme plus détaillée.

Exemples :

L'événement Demande client du MCT pourra se transformer, dans le MOT, en la succession d'événements suivants :

- Demande initiale.
- Demande modifiée.

L'événement Remise chèque du MCT pourra se transformer, dans le MOT, en la décomposition d'événements (sous-type d'événement) suivants :

- Chèque sur place de notre banque.
- Chèque sur place d'une autre banque.
- Chèque hors place de notre banque.
- Chèque hors place d'une autre banque ;

On trouvera aussi plus fréquemment dans le MOT des événements *temporels* liés au rythme de l'entreprise et qui représentent des échéances. Ces événements échéances n'ont pas d'acteur émetteur explicite (il s'agit en théorie de programmation effectuée par le système de pilotage) :

- Début / fin de mois.

- Début / fin de journée.
- Chaque jour à 11h30.
- Le 25 du mois.

A ces événements périodiques peuvent s'ajouter des événements *décisionnels* qui représentent le déclenchement de tâches à l'initiative explicite des représentants du système de pilotage (fréquemment l'exécutant de la tâche)

- Décision de relancer.
- Sur demande.

A la différence du MCT, dans un MOT, la présence d'un événement formalisant le déclenchement d'une tâche n'est pas obligatoirement représenté sur le schéma; en particulier lorsque deux tâches s'enchaînent dans le même poste, la fin de la première tâche déclenchant immédiatement la seconde. Par contre, tout changement de poste ou de phase (voir plus loin « La phase ») doit faire apparaître un message (ou un état) qui assure la transmission de l'activité.

Dans le modèle organisationnel, on accordera plus d'importance au contenu des messages en décrivant la liste et la structure des informations (surtout lors de l'étude détaillée). Cet aspect de description des données, associé aux traitements, sera étudié plus en détail lors de la validation, sous l'appellation de modèle externe ou vue externe.

L'état

La modélisation des états reste la même qu'au niveau conceptuel; toutefois, les états sont plus fréquemment formalisés au niveau MOT.

Ils expriment des situations du système d'information, plus particulièrement au niveau des données mémorisées, et constituent :

- soit des conditions préalables à une tâche,
- soit des situations résultantes conditionnelles d'une tâche.

Ils permettent entre autre de répartir dans des procédures organisationnelles différentes des activités qui sont en fait dépendantes l'une de l'autre; l'état résultant d'une tâche étant l'état préalable d'une autre.

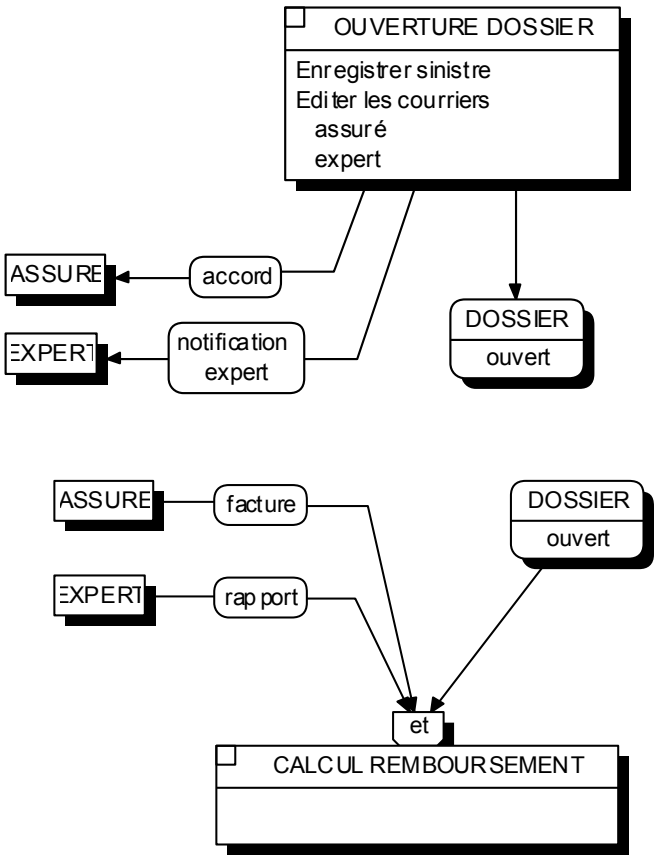


Figure 8.3 : Tâches dépendantes via l'état Dossier ouvert

Rappelons que l'expression de l'état d'un objet peut provenir :

- de la valeur d'une propriété ad hoc de l'objet (entité ou relation),
- d'une règle de traitement impliquant des propriétés de l'objet voire toute une structure de données liées à cet objet; dans ce dernier cas, on sera en présence d'un sous-schéma de données associé à la définition d'un état.

Dans un MOT, les états se comportent comme des événements et résultats qui permettent de préciser les conditions du fonctionnement des procédures organisationnelles, l'enchaînement des tâches restant l'objectif principal de cette modélisation.

On peut par ailleurs envisager un autre type de modélisation privilégiant l'enchaînement des états successifs d'un même objet, conditionné par des événements et des transformations (activités). Cet autre angle de vue de la dynamique d'un système d'information représente le *cycle de vie de l'objet* (voir Chapitre 9 : « Cycle de vie des objets »)

La tâche

La tâche type modélise un ensemble nommé d'activités élémentaires, perçues comme homogènes, concourant à un même but.

La symbolisation graphique de la tâche reprend celle de l'opération et des concepts secondaires associés définis au niveau conceptuel : synchronisation, description et conditions d'émission.

La tâche peut également être perçue comme la décomposition d'une opération conceptuelle. Par exemple, la figure 8.4 montre la décomposition en tâches de l'opération conceptuelle Ouverture dossier

---

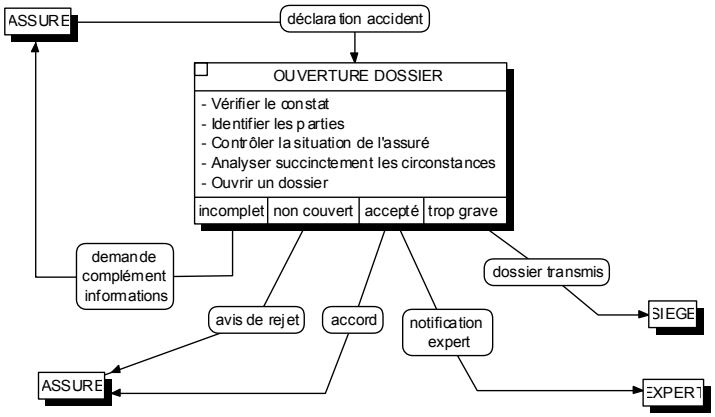
La décomposition organisationnelle ne reconduit pas nécessairement la liste et l'ordre des fonctions élémentaires composant l'opération. Au contraire, le travail du concepteur consiste à imaginer de nouveaux découpages et assemblages d'activités élémentaires, ainsi que des enchaînements et répartitions adaptées aux ressources des postes. La figure 8.9 présente une procédure organisationnelle basée sur une autre décomposition de l'opération conceptuelle d'Ouverture dossier.

Caractères organisationnels

La tâche type est caractérisée par les paramètres suivants :

Le **poste type** auquel la tâche est affectée. Une tâche est intégralement assurée dans un poste type. Cette affectation est unique; on peut toutefois parler de tâches similaires si des activités quasi identiques sont exécutées dans des postes différents. L'affectation d'une tâche à un poste se modélise en représentant la tâche dans la colonne du poste correspondant.

Le **degré d'automatisation** de la tâche. Par abus de langage, et par rapport à l'objectif de la méthode, seule l'informatisation sera prise en compte.



décomposée en

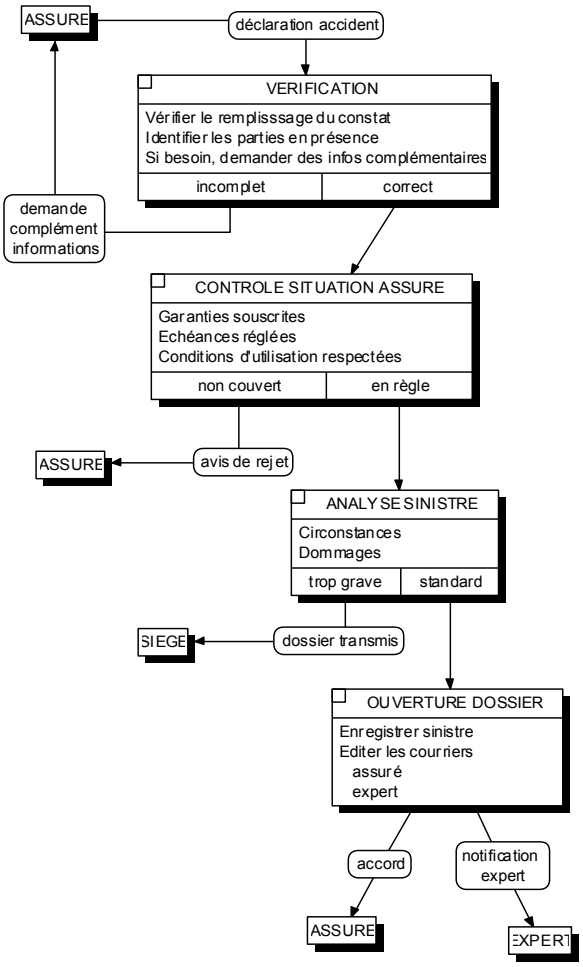


Figure 8.4 : Décomposition de l'opération Ouverture dossier en tâches

Le poste disposant de deux types de ressources, humaine et informatique, le degré d'automatisation traduit l'utilisation de ces ressources pour l'exécution de la tâche. On distingue ainsi trois degrés d'automatisation :

- Une tâche peut être *manuelle* (M); durant son déroulement, seule la ressource humaine est mobilisée. Par exemple, le contrôle de la déclaration d'accident, la recherche des articles dans les rayons, etc.
- Une tâche peut être *interactive* (I) ou conversationnelle (C) ; durant son déroulement, les ressources humaine et informatique sont mobilisées. Par exemple, la saisie d'un sinistre, l'enregistrement d'une commande.
- Une tâche peut être *automatique* (A); durant son déroulement, seule la ressource informatique est mobilisée. L'intervention humaine pour le lancement ou la récupération des résultats de la tâche ne remet pas en

cause le caractère automatique, car la ressource humaine n'est pas impliquée dans l'activité même de la tâche. Par exemple, le calcul de la quittance, l'édition des relevés.

Le **délai de réponse** de la tâche. Ce délai exprime la rapidité de prise en compte d'une nouvelle occurrence d'événement, à condition que l'ensemble des ressources nécessaires à l'exécution de la tâche soient disponibles. On distingue deux valeurs :

- *Réponse immédiate* (I); dès la survenance de l'événement et si les ressources sont disponibles, la tâche traite l'événement.
- *Réponse différée* (D); le déclenchement de la tâche n'est pas uniquement lié à la survenance de l'événement, mais attend une condition complémentaire — délai, intervalle de temps, un ordre du pilote, généralement programmé (la fin de journée, le début du mois...). Les occurrences de l'événement attendent ce top extérieur qui permettra à la tâche de les traiter.

Le **mode de fonctionnement** de la tâche. Comment prend-elle en compte les différentes occurrences d'événement présentes ? Ce mode de fonctionnement peut prendre deux valeurs :

- *Unitaire* (U); la tâche et les ressources associées traitent les occurrences d'événement une par une. A la fin de la tâche, les ressources libérées redeviennent disponibles soit pour prendre une nouvelle occurrence en attente sur la même tâche, soit pour permettre à une autre tâche de démarrer.
- *Par lot* (L); la tâche et les ressources associées prennent en charge un lot (dont la taille est à préciser) et restent mobilisées jusqu'à la fin du traitement du lot.

L'évaluation du caractère unitaire ou par lot doit s'effectuer par référence à l'occurrence définie pour l'événement en entrée de la tâche.

Des caractères tels que la périodicité, la fréquence et la durée peuvent venir compléter la description organisationnelle de la tâche.

A l'exception du degré d'automatisation, ces caractéristiques organisationnelles peuvent être optionnelles en étude préalable.

Description d'une tâche

La tâche est décrite par l'ensemble des activités homogènes à réaliser et comporte entre autre :

- Des *règles de traitements* exécutées par la tâche. Une règle de traitement consiste à décrire, sous une forme structurée, un algorithme appliqué à

un ensemble de données.

- Des actions effectuées par la tâche sur des données mémorisées, constituant un *sous-schéma de données*. Ces actions, de mise à jour ou de consultation, mettent en jeu les informations présentes sur les messages, les informations impliquées dans les règles de traitement et les informations déjà mémorisées représentées par un sous-schéma du MCD ou du MOD.
- Des choix et décisions effectués par l'utilisateur lors de l'exécution de la tâche

Suivant le degré de détail souhaité, le concepteur peut représenter une tâche soit au sein d'une procédure organisationnelle (figure 8.10) sans détailler la description mais en privilégiant enchaînement au sein des postes, soit en "gros plan" avec l'expression de la description, des règles et du sous-schéma (figure 8.5).

La règle de traitement

Une règle de traitement est un ensemble structuré sous une forme algorithmique :

- d'expressions logiques,
- d'expressions arithmétiques,
- d'actions.

L'introduction explicite des règles de traitement dans l'expression du contenu des tâches s'est effectué très tôt dans la méthode Merise sans disposer pour cela de formalisation. L'isolement et la réutilisation des règles dans le processus de développement informatique a rendu progressivement nécessaire l'expression autonome de règles de traitement nommées, conduisant à la constitution de répertoires de règles. Ce phénomène a été particulièrement sensible dans les domaines à forte composante réglementaire (administration, assurance, banque,...).

L'introduction explicite des règles de traitement dans l'expression du contenu des tâches s'est effectué très tôt dans la méthode Merise sans disposer pour cela de formalisation. L'isolement et la réutilisation des règles dans le processus de développement informatique a rendu progressivement nécessaire l'expression autonome de règles de traitement nommées, conduisant à la constitution de répertoires de règles. Ce phénomène a été particulièrement sensible dans les domaines à forte composante réglementaire (administration, assurance, banque,...).

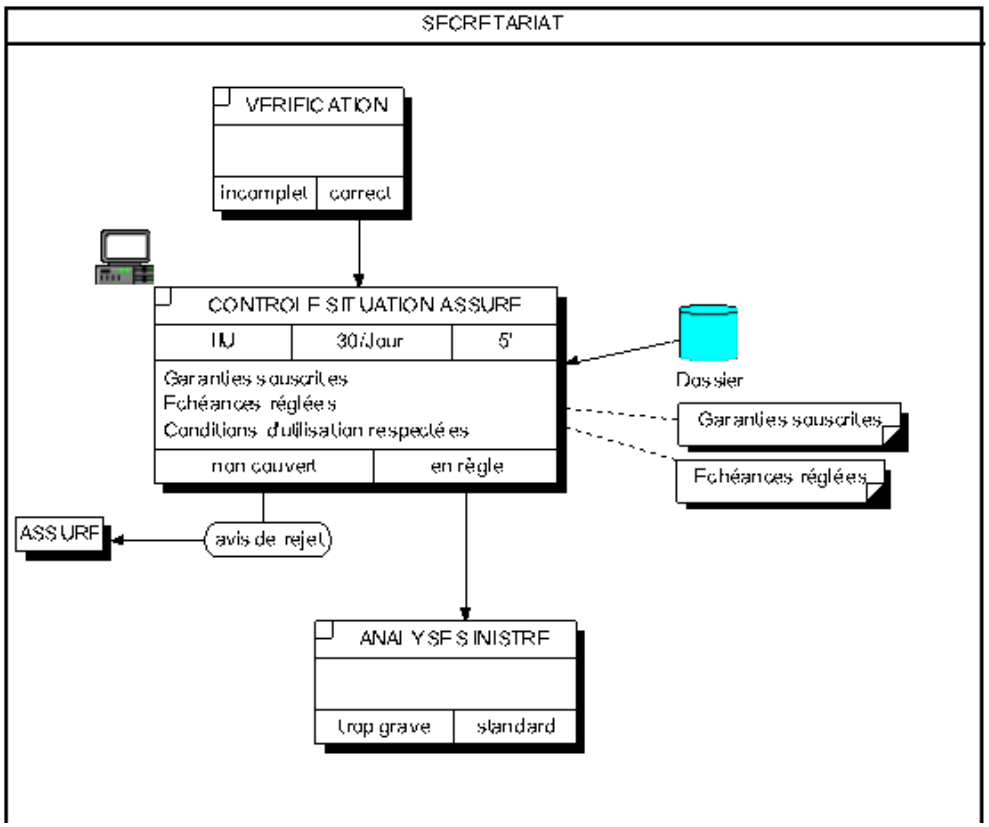


Figure 8.5 : Représentation détaillée d'une tâche.

Comme nous l'avons déjà expliqué dans la modélisation conceptuelle des données (voir « Règles » dans MCD) on peut distinguer deux catégories de règles :

- Des règles de traitement spécifiques et contingentes à une ou certaines activités, c'est à dire conséquentes aux événements qui déclenchent l'activité. Ces règles sont à formaliser au niveau de la modélisation des traitements (organisationnel ou logique).
- Des règles de traitement indépendantes de toute activité spécifique et qui doivent être appliquées au niveau des données pour maintenir une cohérence globale. Ces règles de traitement sont décrites au niveau du modèle conceptuel de données.

Dans la description des traitement appliqués par une tâche, on peut utiliser les deux catégories de règles. La figure 8.5 illustre la représentation graphique des règles Garanties souscrites et Echéances réglées utilisées par la tâche Contrôle situation assuré.

Le contenu détaillé de la règle peut être exprimé sous la forme de langage

structuré (naturel ou pseudo-code) où peuvent apparaître :

- des propriétés du modèle de données présentes dans le sous-schéma de données,
- des règles de traitements, permettant ainsi une construction arborescente des règles de traitement avec possibilité de réutilisation de règles élémentaires.

On peut en outre établir une typologie des règles de traitements selon leur nature (surtout pour les règles élémentaires), par exemple : calcul, contrôle, sélection, sommation.

La mise en évidence des règles de traitements et leur modélisation autonome dès le modèle organisationnel favorisera leur implémentation informatique ultérieure.

Le sous-schéma conceptuel/organisationnel de données

Un sous-schéma conceptuel/organisationnel de données (voir figure 8.6) est un sous-ensemble, généralement connexe, d'entités types et de relations types, définies sur le modèle conceptuel de données (MCD) ou le modèle organisationnel de données (MOD), et associé à une tâche. Cette tâche effectue des actions (mise à jour ou lecture) sur les occurrences de ces entités et relations.

Sur le schéma d'un MOT, le sous-schéma (ou vue de données) apparaît seulement par son nom, associé à une icone. Les actions sur l'entité pivot (voir Confrontation données traitements) peuvent être éventuellement symbolisées avec les conventions suivantes

Certains auteurs détaillent explicitement le sous-schéma et les actions, ce qui nous semble surcharger la représentation graphique d'un MOT.

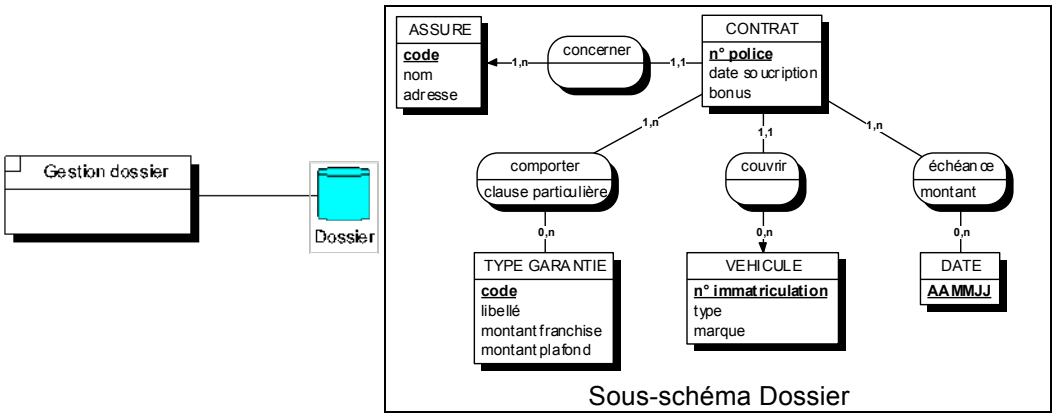
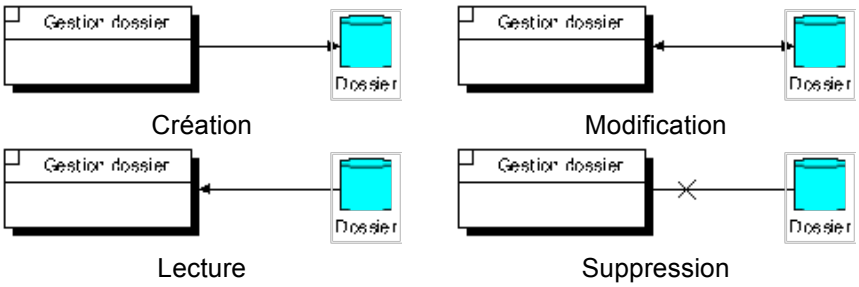


Figure 8.6 : Sous-schéma de données



Plusieurs tâches différentes peuvent partager un même sous-schéma conceptuel/organisationnel de données, avec éventuellement des actions différentes.

On peut également associer un sous-schéma à une procédure organisationnelle, à une phase, voire à une règle de traitement ou un état.

Les ressources

Le formalisme préconisé par Merise, au niveau organisationnel des traitements, ne propose pas explicitement de modélisation et de représentation graphique des ressources utilisées, de leur disponibilité et de leur consommation.

Cependant, le formalisme des réseaux de Petri (base théorique de la modélisation des traitements dans Merise) permettrait de représenter cette disponibilité et consommation d'une ressource, en la modélisant sous un concept au comportement équivalent à celui d'un événement/résultat, avec de préférence un symbole différent. Certains praticiens de Merise utilisent depuis plusieurs années cette formalisation (voir figure 8.7).

Il est certain que les consommations de ressources par les tâches du MOT permettent une approche du dimensionnement technique et organisationnel (effectif et charges de travail des ressources humaines, caractéristiques des ressources techniques).

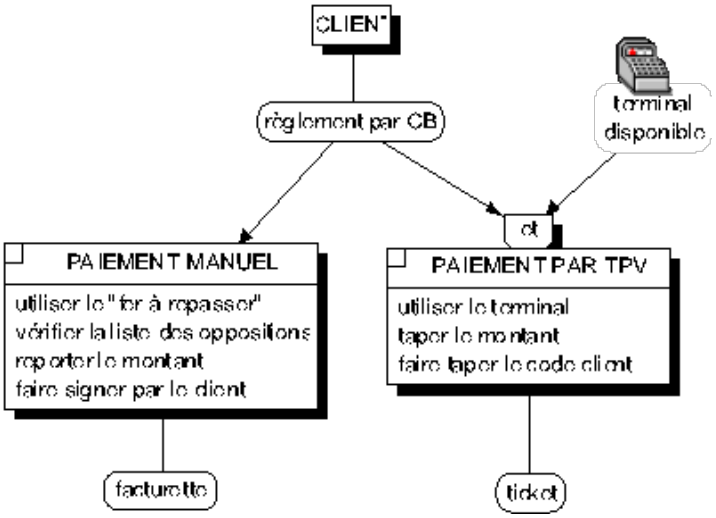


Figure 8.7 : Modélisation de disponibilité de ressources.

La phase

La phase est une succession de tâches exécutées consécutivement au sein d'un même poste. Dès qu'une occurrence d'événement rentre dans la phase, c'est-à-dire déclenche la première tâche de cette phase, les ressources humaine et informatique nécessaires à l'exécution de ces tâches restent mobilisées et indisponibles (même si elles ne sont pas utilisées par certaines tâches) jusqu'à la fin du traitement de l'occurrence d'événement initial (cette sortie pouvant survenir à l'issue de toute tâche de la phase); la phase pourra alors traiter une nouvelle occurrence d'événement. La figure 8.8 illustre un exemple de regroupement en phase.

La phase permet de reconstituer une séquence d'activités qui ne peuvent être interrompues par d'autres événements. Le découpage en tâches au sein de la phase est alors fréquemment dû à une alternance de degrés d'automatisation différents.

- La notion de phase est intéressante à plusieurs titres :
- Elle permet une meilleure analyse de l'utilisation optimale des ressources dans le cas où certaines ressources ne sont pas mobilisées tout au long de la phase.
- Elle offre un découpage des rythmes de travail au sein du poste.
- Elle met en évidence des points d'attente entre les phases, en particulier entre les postes de travail, qui risquent de générer des files d'attente.

Si des tâches consécutives au sein d'une phase sont de mêmes caractéristiques

(par exemple Automatisée, Immédiat), il est possible de regrouper ces tâches en une seule et même tâche.

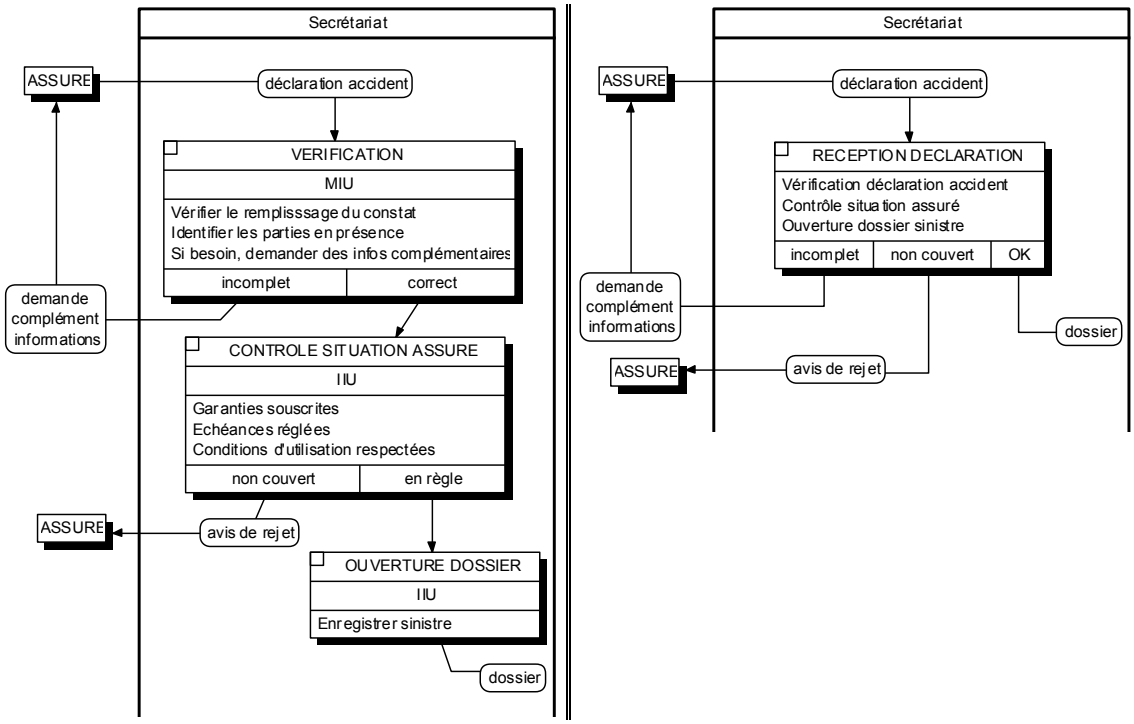


Figure 8.8 : Regroupement de tâches en phase.

La procédure organisationnelle

C'est un enchaînement de tâches et/ou de phases, d'intérêt pour l'organisation. La procédure organisationnelle prend en compte un événement type (ou plusieurs synchronisés), appelé événement initial de la procédure, et produit tous les résultats types qui en découlent.

Procédures, phases et tâches permettent d'introduire de façon naturelle une modularité dans la présentation du MOT.

Il arrive qu'un événement type présente en fait plusieurs variantes.

Par exemple, la demande de passeport recouvre les deux cas suivants : demande d'un premier passeport, demande de renouvellement.

Selon le cas, le déroulement de la procédure sera différent; certaines tâches ne seront pas exécutées, ou seront exécutées selon des règles différentes.

On parle alors de variantes différentes de la procédure; on précise souvent la fréquence de ces variantes (40 % en première délivrance, 60 % en renouvellement).

Représentation graphique d'un modèle organisationnel de traitements

Cette représentation reprend largement les diagrammes de circulation bien connus des organisateurs. En pratique la présentation de la modélisation organisationnelle des traitements s'effectue a travers les différents schémas de procédures; exprimés en phases ou tâches selon le degré de détail souhaité. La figure 8.9 présente un schéma MOT associé à la procédure d'Ouverture d'un dossier de sinistre du cas assurance.

Organisation synchrone ou asynchrone

Dans la modélisation organisationnelle des traitements, l'enchaînement entre les tâches est un aspect important de l'expression de la procédure organisationnelle. La modélisation de ces enchaînements peut s'effectuer de diverses manières qui, chacune, traduisent des fonctionnements différents.

A sein d'une phase, l'enchaînement entre deux tâches se modélise par un simple lien fléché (pas d'événement ou d'état intermédiaire), exprimant ainsi la continuité des activités dans la phase (non interruptible).

Par contre, lorsque l'enchaînement s'effectue entre des tâches appartenant à des phases différentes (au sein d'un même poste ou entre des postes distincts), l'enchaînement doit être formalisé par l'intermédiaire soit d'un événement soit d'un état. Le choix de l'une ou l'autre modélisation exprimera un fonctionnement *synchrone* ou *asynchrone* de la procédure.

Dans le cas d'un fonctionnement synchrone, modélisé par un événement (quel que soit son support), l'événement émis par l'une des tâches vient "stimuler" l'autre tâche qui la prendra en compte généralement immédiatement (si ses ressources sont disponibles). Le rythme de fonctionnement de la tâche réceptrice est totalement synchronisé par celui de la tâche émettrice.

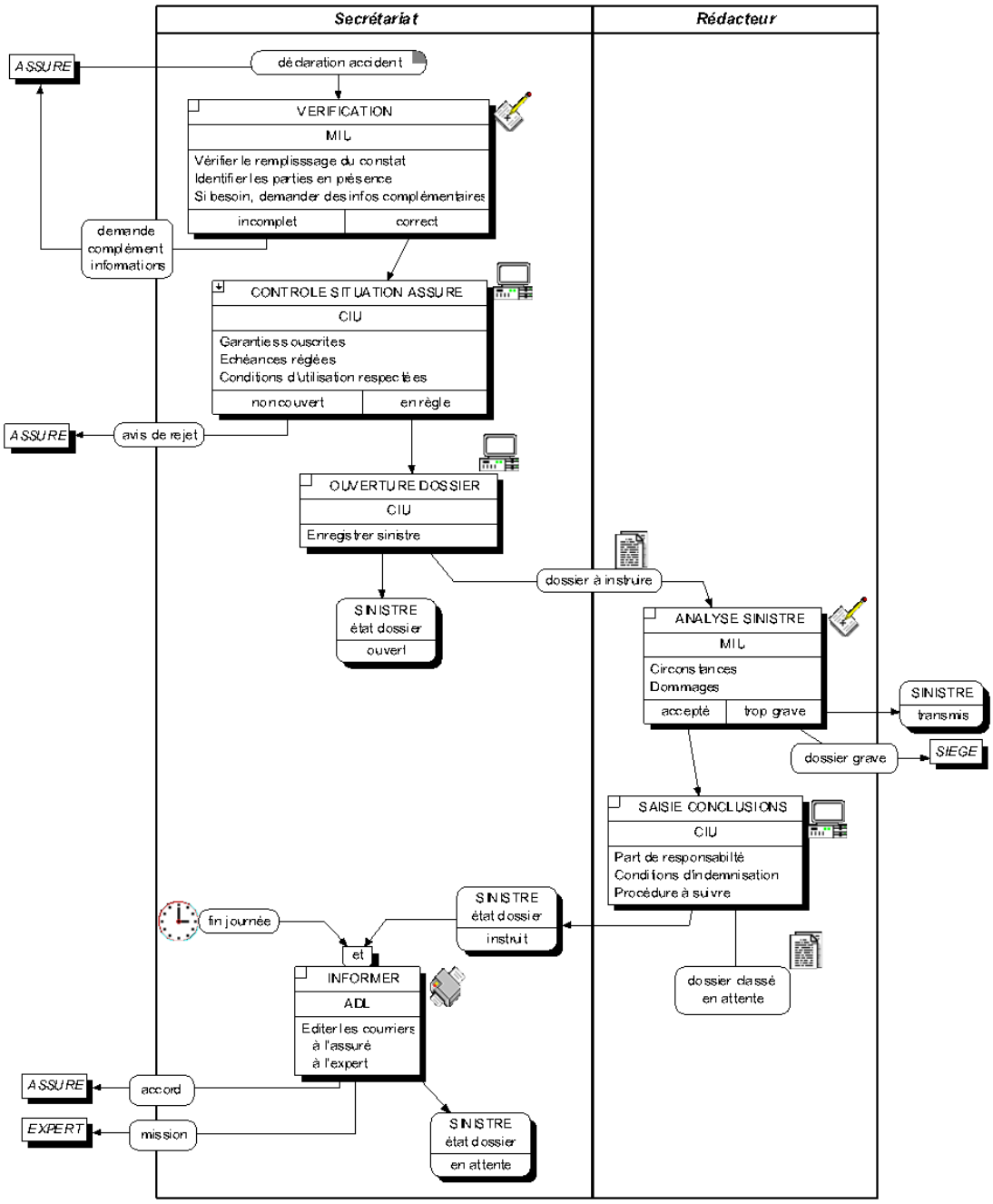


Figure 8.9 : Exemple de schéma organisationnel de traitements
Procédure d'ouverture d'un dossier dans le cas assurance.

Par exemple, dans la figure 8.9, le concepteur a choisi un fonctionnement synchrone entre la tâche Ouverture dossier au secrétariat et la tâche Analyse du sinistre du rédacteur. L'enchaînement est donc modélisé par un événement, matérialisé ici par la transmission du dossier à instruire. Le rédacteur sera

immédiatement averti qu'un dossier est à instruire; normalement il devrait traiter dans la séquence le dossier communiqué. (un tel choix d'organisation est bien évidemment discutable).

Dans le cas d'un fonctionnement asynchrone, la tâche émettrice produit en résultat un état qui est repris en entrée de la tâche réceptrice, synchronisé par un événement temporel ou décisionnel; à la différence de l'événement, l'état n'a aucun rôle de stimulus. Le rythme de fonctionnement de la tâche réceptrice est totalement désynchronisé par rapport à celui de la tâche émettrice; la tâche réceptrice doit prendre l'initiative du déclenchement des traitements.

Toujours dans la figure 8.9, le concepteur a choisi un fonctionnement asynchrone entre la tâche Saisie des conclusions et la tâche Informer l'assuré et l'expert. L'enchaînement est modélisé par un état; matérialisé par Sinistre état dossier instruit. La tâche Informer qui prendra en compte ces dossiers sera déclenchée à l'initiative du secrétariat, en l'occurrence en fin de journée.

Dans le cas d'une modélisation de fonctionnement asynchrone, il n'est pas nécessaire de faire obligatoirement figurer l'enchaînement entre les tâches. On retrouve alors la même problématique que celle évoquée au niveau des MCT entre l'approche orientée processus et l'approche orientée états.

Le choix d'une organisation synchrone ou asynchrone relève également des préoccupations du B.P.R. et trouve aujourd'hui un écho particulier dans les nouvelles approches de "workflow".

Notions complémentaires pour la formalisation organisationnelle des traitements

Les concepts de base présentés suffisent généralement pour construire un modèle organisationnel de traitements. Toutefois, certaines situations à modéliser rendent nécessaires des éléments complémentaires tels que :

- La *fréquence* d'un événement, fréquence de survenance des occurrences de cet événement. Elle peut être constante (100 commandes/jour) ou variable (loi de distribution dans le temps).
- La *capacité* d'un événement, nombre maximum d'occurrences qu'un événement type peut accepter. Par défaut, cette valeur est infinie.
- La *participation* d'un événement à une synchronisation, nombre d'occurrences différentes de l'événement nécessaires au déclenchement de la synchronisation. Par défaut cette valeur est 1, une valeur typique est tous.
- La *durée de contribution* d'un événement à une synchronisation, temps d'attente maximum entre la survenance de l'événement et sa consommation par la tâche; au-delà, cette occurrence d'événement disparaît. Par défaut, cette valeur est infinie. Une valeur typique est 0,

indiquant que l'événement doit survenir en dernier si l'on veut démarrer la tâche.

- Des *conditions locales* d'une synchronisation, expression logique portant sur les valeurs des messages permettant de sélectionner des occurrences d'événements devant participer à la synchronisation.
- La *durée limite* de synchronisation, temps maximum d'attente entre la survenance du premier événement contributif et l'activation de la synchronisation. Si ce délai est dépassé, toutes les occurrences d'événements en attente sur cette synchronisation sont purgées.
- Le *délai de synchronisation*, temps passé entre le moment où la condition est vraie et le moment où la tâche est déclenchée. Par défaut, cette valeur est 0.
- La *durée* de la tâche, temps passé entre le déclenchement de la tâche et la production de résultat. Cette durée peut être variable suivant les conditions d'émission des résultats.
- La *périodicité* de la tâche, période ou cycle de répétition d'une tâche. Par exemple : quotidienne, hebdomadaire, mensuelle, annuelle.
- La *duplication d'un résultat*, nombre d'occurrences identiques d'un résultat émis par la tâche. Par défaut, cette valeur est 1.

Les notions de fréquence, de périodicité et de durée sont essentielles, et dès l'étude préalable, il est nécessaire de les connaître avec une précision suffisante. On indiquera pour chaque estimation la valeur minimale, la valeur maximale et la valeur la plus probable. Bien entendu ces estimations devront être cohérentes avec la quantification établie pour le modèle organisationnel des données (voir MOD).

Construction d'un modèle organisationnel de traitements

Par deux fois, le concepteur procédera à l'élaboration d'un modèle organisationnel : lors de l'étude préalable, puis lors de l'étude détaillée. Il faut noter cependant la grande différence d'attitude dans la modélisation, selon les deux étapes d'étude :

En étude préalable, le concepteur doit s'attacher à proposer une variété de solutions par le choix des postes, du degré d'automatisation et de l'organisation des tâches. Le concepteur ne doit pas, par contre, s'attarder à détailler le contenu des tâches ou des messages.

En étude détaillée, les principes organisationnels ont été fixés, il reste au concepteur à détailler le contenu des tâches et des messages, à expliciter toutes les procédures même secondaires.

Dans tous les cas, on retrouvera des principes généraux de construction que nous résumons :

- Faire le choix des postes, en spécifiant les ressources humaines et informatiques.
- Décomposer chaque opération en tâches, les ordonner (le nouvel enchaînement des tâches n'est pas nécessairement celui des fonctions de l'opération), les affecter aux postes, préciser les différentes caractéristiques (degré d'automatisation, délai de réponse, mode de travail).
- S'assurer de la faisabilité des tâches par rapport aux ressources composant le poste.
- Préciser les différentes phases.
- Evaluer l'ergonomie générale de chaque poste par rapport à l'ensemble des phases à assurer; évaluer si la nature des tâches assignées est compatible avec les qualifications des personnes affectées au poste, avec les conditions générales de l'environnement.
- Envisager des solutions alternatives : variantes de procédures.

Expression d'un modèle organisationnel de traitements

Du point de vue documentation, pour exprimer les solutions d'organisation (variantes du MOT), on présentera les éléments suivants :

description des postes;

schéma d'enchaînement des tâches présenté par procédure organisationnelle;

suivant la nécessité, présentation de quelques variantes de la procédure;

liste des phases par poste;

pour chaque tâche, description (succincte ou détaillée suivant le niveau de l'étude)

- des événements et états en entrée;
- des règles de traitement exprimées en termes d'algorithme;
- du sous-schéma conceptuel ou organisationnel de données;
- des actions effectuées sur les données mémorisées;
- des résultats et états produits ;
- des conditions de production de ces résultats.

Plus largement, ces solutions d'organisation doivent être présentées aux gestionnaires et aux utilisateurs du domaine, selon des modalités adaptées à l'entreprise.

Cette présentation permettra une évaluation et si nécessaire une amélioration des solutions envisagées; mais aussi une sensibilisation des personnels aux changements liés à la future organisation du travail.

Cela suppose bien entendu une préparation sérieuse de ces présentations, et un temps suffisant de parole pour les gestionnaires et les utilisateurs, leur permettant de formuler leurs observations, critiques et suggestions :

- sur le contenu et la désignation des postes de travail, des tâches et des messages;
- sur le déroulement des tâches dans le temps, sans oublier les principaux cas particuliers;
- sur les différentes variantes présentées; comment satisfont-elles les exigences ergonomiques et techniques, sociales et économiques ?

Modularité dans un modèle organisationnel de traitements

Comme pour le MCT, il est nécessaire de pouvoir représenter un MOT sous une forme globale ou sous une forme détaillée.

Au niveau organisationnel, nous optons pour la technique de *stratification* (voir chapitre 3 « Niveaux d'abstraction et degrés de détails »), qui consiste à définir des modèles avec des concepts types différenciés. Notre expérience nous conduit à utiliser trois niveaux de détail : modèle organisationnel de *procédures*, modèle organisationnel de *phases*, modèle organisationnel par *tâches*.

Dans chaque type de modèle, l'activité (la procédure, la phase, la tâche) est représentée par le symbole classique de l'activité.

Le modèle organisationnel de procédures permet une vision très macroscopique : seules figurent sur le schéma les procédures et leurs enchaînements avec, pour chacune, le ou les événements initiaux et les résultats qui en découlent. On retient comme critère organisationnel la répartition en postes et un degré très global d'automatisation (essentiellement manuel, essentiellement transactionnel, essentiellement automatisé...). Pratiquement, une telle modélisation est assez proche du degré de détail d'un MCT courant auquel on adjoint certains aspects organisationnels. Ce degré de modélisation peut être utilisé avec profit pour l'analyse de l'existant (voir chapitre 17 « Analyse du modèle organisationnel actuel »). On peut appeler ce type de modèle Macro-MOT.

Le modèle organisationnel de phases permet une vision intermédiaire du fonctionnement organisé : seules figurent sur le schéma les phases et leurs enchaînements avec, pour chacune, le ou les événements initiaux et les résultats qui en découlent. Comme pour le modèle de procédures, on exprime la répartition par poste et le degré global d'automatisation. L'intérêt d'un tel modèle réside dans l'unité de temps et de mobilisation des ressources exprimée par la phase, qui permet de formaliser les rythmes de fonctionnement. Cette modélisation offre un degré de modélisation intermédiaire très apprécié dans les présentations des solutions organisationnelles.

Le modèle organisationnel de tâches est une vision relativement détaillée : elle seule permet cependant de préciser la nature exacte et homogène des activités organisées ainsi que leurs enchaînements. D'après notre expérience, ce niveau de modélisation est le plus approprié, même au niveau de l'étude préalable, pour élaborer des solutions organisationnelles, quitte à le présenter ensuite sous la forme d'un modèle de phases. Ce modèle est, par contre, indispensable au niveau de l'étude détaillée, pour préparer le passage au niveau logique.

Ergonomie et modèles organisationnels de traitements

Pour analyser ou concevoir un système d'information, il ne suffit pas d'avoir une bonne approche technique pour obtenir un système performant, c'est-à-dire sûr, maniable et minimisant les risques d'erreur humaine.

Comme pour tous les systèmes complexes (nucléaire, navigation aérienne, industrie chimique), il faut de plus en plus prendre en compte la qualité de l'interface homme-machine, c'est-à-dire la dimension ergonomique.

Dans le cadre de cet ouvrage, nous ne traiterons pas en détail cet aspect fondamental de l'ergonomie. Nous nous limiterons à évoquer les questions principales que le concepteur doit se poser :

Comment analyser l'activité réelle d'un opérateur au travail ?

Comment définir les interfaces de dialogue d'un logiciel ?

Comment évaluer l'ergonomie d'un logiciel ou un progiciel préexistant ?

Nous renvoyons le lecteur à des études et livres traitant de l'ergonomie en général [Guelaud et al. 75] [Laville 81] [Sperandio 84] [De Moulin 86] ou spécialisés en ergonomie du logiciel comme [Valentin & Lucongsang 87] dont nous nous inspirerons largement.

Introduction à l'ergonomie du logiciel

L'ergonomie peut être définie comme l'ensemble des connaissances scientifiques relatives à l'homme et nécessaires pour concevoir des outils, des techniques et des dispositifs qui puissent être utilisés avec le maximum de confort, de sécurité et d'efficacité.

Les principales caractéristiques de l'ergonomie sont :

- la nécessité de connaissances interdisciplinaires (physiologie, psychologie, sociologie, ingénierie, médecine...);
- l'effort de prise en compte de la globalité d'une situation de travail (contenu des tâches, environnement, organisation, formation...);
- l'analyse du travail réel effectué par les opérateurs.

Plus précisément, l'ergonomie du logiciel a pour objectif de permettre une meilleure adéquation des applications informatiques aux besoins des utilisateurs.

En ingénierie de systèmes d'information, nous intégrerons les considérations ergonomiques :

- au niveau organisationnel à travers l'analyse des postes,

- au niveau logique à travers la conception des interfaces homme-machine.

L'analyse du travail

En ergonomie du logiciel, un certain nombre de techniques, d'éléments méthodologiques ont été proposés afin de guider l'ergonome. Son travail sera tout d'abord centré sur une analyse du travail regroupant une *analyse de la tâche* et une *analyse de l'activité* [Valentin & Lucongsang 87].

Lors de l'analyse de la tâche, l'ergonome recueillera des éléments sur l'organisation du travail, les liens entre les services, les caractéristiques des postes de travail et il précisera surtout le contenu du travail prescrit dans le cadre de cette tâche. Il précisera ainsi les objectifs à atteindre, les procédures et les règles de fonctionnement que l'utilisateur est censé utiliser.

Ensuite, au travers de l'analyse de l'activité, l'ergonome s'efforcera de comprendre comment l'utilisateur travaille ou travaillera réellement à son poste de travail pour réaliser cette tâche prescrite. Une observation fine permettra de relever les aspects de l'activité qui conditionnent le plus la réalisation de la tâche. Elle consistera entre autres à identifier les informations et les ressources nécessaires au déroulement de la tâche, leur ordre d'utilisation, les informations manquantes, inutilisées, les repères principaux permettant à l'utilisateur de prendre ses décisions, ainsi que les erreurs, oublis constatés.

L'ergonome peut ensuite effectuer une synthèse, en comparant les données recueillies lors des analyses de tâche et d'activité. En cherchant les raisons des écarts observés entre le prescrit et le réalisé, il fera ressortir des points critiques dans le fonctionnement de l'application sur le poste de travail.

Pour conclure, nous dirons que l'ergonomie n'est pas un vernis supplémentaire ajouté à un logiciel classique par ailleurs; c'est une autre façon d'étudier et de comprendre le travail humain, en particulier pour concevoir de nouveaux systèmes plus souples et mieux adaptés aux utilisateurs.

Le logiciel n'est pas le tout du travail de l'entreprise; pour le travail de production administrative quotidienne, même sans informatique, l'ergonomie peut apporter beaucoup en matière de conditions de travail au sens large, comme en matière d'efficacité. Il est important de situer les efforts en ergonomie du logiciel dans la perspective d'ensemble de l'amélioration de la performance de l'entreprise. S'il est parfois nécessaire de consulter des spécialistes de ces problèmes : il faut aussi se rappeler que les utilisateurs ont certainement de bonnes idées sur la façon d'améliorer ou de mieux organiser leur travail; une approche sociotechnique et participative est le meilleur complément à l'approche ergonomique.

9

Cycle de vie des objets et objets métiers

Introduction

Ces dernières années, l'approche objet a progressivement doté le génie logiciel d'un nouveau référentiel méthodologique dont la méthode Merise a su tirer profit en adaptant certains concepts au contexte de l'ingénierie de systèmes d'information.

Ainsi (cf. chapitre 7) le formalisme entité-relation a été étendu pour prendre en compte dans les modèles de données le sous-typage d'entité, la spécialisation et la généralisation. Il en est de même de la notion d'état qui a naturellement trouvé sa place dans la modélisation des traitements. Cette notion d'état est associée à une modélisation dynamique des traitements qui se développera dans Merise au travers de l'élaboration de cycles de vie d'objets (CVO).

La notion d'objet métier est aussi apparu dans le sillage des méthodes objet en différenciation de celle d'objet logiciel. Souvent associée à une offre commerciale de composants logiciels répondant aux besoins d'un métier donné, la notion d'objet métier reste encore floue et en conséquence difficilement opératoire dans le cadre d'une méthode de conception de systèmes d'information comme Merise. Elle est cependant une notion en pleine émergence permettant d'appréhender l'ingénierie des systèmes d'information selon de nouveaux éclairages, notamment celui de la réutilisation.

Dans ce chapitre, nous présentons en premier la modélisation du cycle de vie d'objet (CVO) en exposant l'intérêt de sa spécification, le formalisme que nous préconisons pour les modéliser ainsi que quelques conseils d'élaboration. En second, nous abordons le concept d'objet métier en le positionnant tout d'abord dans le cadre de l'ingénierie des systèmes d'information avec Merise, puis en introduisant un ensemble d'éléments méthodologiques de modélisation relatifs d'une part à leur formalisation et d'autre part aux démarches associées à leur élaboration.

Notons que les notions de CVO et d'objet métier, dont l'intérêt a été confirmé

par la pratique, ont été évoqués dans Merise dès le milieu des années 90, notamment dans Merise/2 [Panet, Letouche 94]. Enfin, ces deux notions complètent le cadre de modélisation proposé par Merise et concernent principalement les niveaux d'abstraction conceptuel et organisationnel, et dans une moindre mesure le niveau logique.

Cycle de vie des objets

Approche fonctionnelle ou dynamique des traitements

L'analyse séparée données - traitements est, nous l'avons vu en première partie, l'un des principes fondamentaux de la méthode Merise où les données représentent l'aspect statique et les traitements, l'aspect activités. Depuis le début des travaux sur la méthode Merise, deux approches se sont présentées - voire opposées - pour la description des activités (les traitements) et de leur dynamique:

- une approche « fonctionnelle et événementielle » dans laquelle les activités (processus assurant des fonctions) sont déclenchées par l'apparition d'événements (approche fonctionnelle orientée événement - activité)
- une approche centrée sur la spécification de l'évolution des données dans le temps, leur changement de valeur ou d'état prescrit ou proscrit suite à leur transformation par des activités (approche orientée état - transition) et que nous qualifieront aussi, par abus de langage, d'approche « dynamique .

L'approche retenue dans Merise est une approche fonctionnelle événementielle en partie inspirée des réseaux de Pétri (domaine de l'automatique) que l'on retrouve dans les modèles conceptuels, organisationnels et logiques de traitements. Cette approche novatrice par l'aspect événementiel reste assez classique de par son caractère fonctionnel.

Dans les domaines de l'informatique technique et scientifique, sous l'influence des modélisations des automatismes (automates à états finis), l'approche état - transition s'est progressivement développée. L'approche objet trouvant en grande partie ses origines dans ces domaines, il était naturel de voir la modélisation dynamique réapparaître et cohabiter avec une modélisation fonctionnelle.

Aussi de nombreux auteurs dont ceux de Merise /2 [Panet, Letouche 94] préconisent une modélisation selon trois aspects complémentaires : statique, fonctionnel et dynamique, au lieu de s'en tenir aux deux aspects statique et fonctionnel préconisés à l'origine dans Merise (figure 9.1).

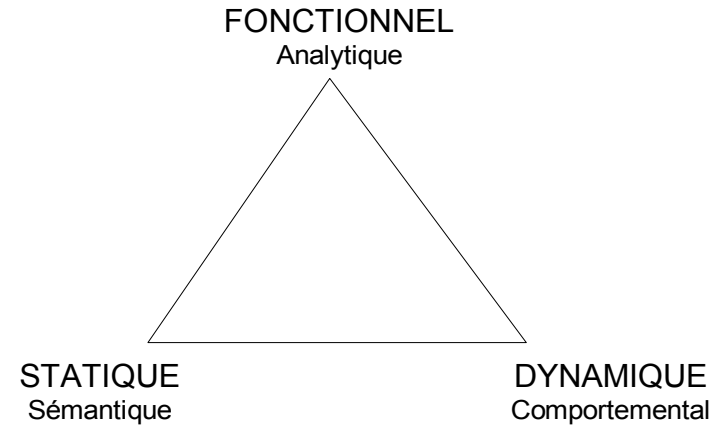


Figure 9.1 : Les trois aspects de modélisation

L'aspect fonctionnel décrit ce que fait le système (les fonctions réalisées) ; l'aspect dynamique décrit comment le système se comporte (les changements d'états de ce dernier). On retrouve ainsi, outre l'aspect statique toujours représenté par les données, les deux approches fonctionnelle et dynamique de modélisation des traitements précédemment évoquées et qui cohabitent.

Modélisation fonctionnelle

C'est une modélisation de l'activité du S.I.O. guidée par la chronologie des activités consécutives à un événement déclencheur (que fait-on suite à l'arrivée d'une déclaration d'accident ?). Elle représente d'abord une vision utilisateur du fonctionnement de son domaine.

On retrouve aujourd'hui cette approche dans le Business Process Reengineering ainsi que dans l'analyse des procédures en vue de certification ISO 9000. Pratiquée depuis longtemps en gestion par les organisateurs, elle s'est appuyée sur des modélisations différentes suivant les méthodes (Structured Analysis, SADT,...) considérant toujours l'activité comme concept central.

Dans la méthode Merise cette modélisation se traduit par l'élaboration des modèles conceptuels, organisationnels et logiques de traitements qui sont amplement développés dans cet ouvrage.

Modélisation dynamique

Cette modélisation des activités se situe dans le contexte d'une approche centrée sur les données. La vision statique des données est représentée par des objets et des associations, modélisés par des entités et des relations d'un MCD ou un MOD dans Merise.

L'évolution de ces objets et associations dans le temps (création, modification des valeurs, suppression) est appelée dynamique des données. L'expression de cette dynamique repose essentiellement sur la modélisation des états successifs pris par ces objets, le passage d'un état à un autre et les traitements associés consécutifs à un événement (le règlement du sinistre à l'assuré fait passer le dossier de "en cours" à "réglé"). La modélisation dynamique correspondant ainsi à une vision des traitements à partir des objets de données.

Cette modélisation dynamique des activités est largement utilisée dans les méthodes objets. La deuxième génération de la méthode Merise propose de modéliser cette dynamique sous la forme de cycles de vie d'objets (CVO).

Formalisme de modélisation du cycle de vie des objets

Concepts généraux de la modélisation de la dynamique

Comme nous venons de le rappeler, la modélisation de la dynamique des données n'est pas nouvelle. Cette modélisation consiste à spécifier les différents changements d'états caractérisant un objet spécifique (ici une entité).

La plupart des formalismes utilisés à cet usage, et en particulier ceux retenus dans les méthodes ou notations objets (OMT ou UML par exemple), reposent sur les concepts suivants:

- *Etat* (abstraction des valeurs des attributs et des associations d'un objet)
- *Evénement* (stimulus accompagné éventuellement d'information)
- *Transition* (modification d'état provoquée par un événement)

Ces formalismes s'expriment par des diagrammes d'états – transitions, communément appelés diagramme d'états, graphes dont les nœuds sont des états et les arcs orientés des transitions, transitions généralement désignées par des noms d'événements, événements à l'origine du changement d'états.

La figure 9.2 illustre le diagramme d'état du dossier de sinistre reprenant l'exemple utilisé dans les chapitres de l'ouvrage relatifs au MCT et au MOT.

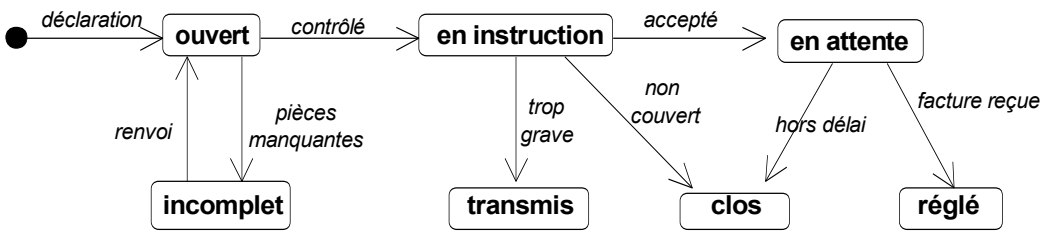


Figure 9.2 : Diagramme d'états du dossier de sinistre

Dans les méthodes de génie logiciel orientées objet le comportement des objets d'une classe peut être décrit formellement en terme d'états et d'événements par un automate à états finis. Cette description ne concerne que les objets ayant un comportement réactif significatif. Les modèles dynamiques préconisés dans OMT spécifient de tels comportements. Dans la notation UML le formalisme retenu pour représenter ces automates s'inspire des « statecharts » proposé par Harel. Ce formalisme, de type diagramme d'états – transitions, permet entre autres de spécifier des conditions booléennes associées au déclenchement d'une transition lors de l'occurrence d'un événement, de spécifier des actions attachées au déclenchement d'une transition et enfin une généralisation et une agrégation d'états.

Concepts retenus pour le cycle de vie des objets dans Merise

Pour modéliser la dynamique associée aux données, nous proposons d'une part de s'inspirer des concepts communément retenus avec les diagrammes d'états, d'autre part de réaliser une économie de formalisme en utilisant celui déjà introduit pour la modélisation des traitements dans Merise.

En effet, les concepts des diagrammes d'états évoqués ci-dessus permettant la modélisation de la dynamique existent déjà dans le formalisme des traitements proposé dans Merise :

- *l'Etat* (appliqué généralement à une entité - objet)
- l'Evénement
- *l'Activité* (opération, tâche) qui sera pour la circonstance, dans ce type de modèle, appelée *Transition* avec si nécessaire la possibilité d'exprimer des synchronisations et des conditions.

La modélisation du CVO Merise présente cependant quelques particularités propres à son utilisation dans le cadre d'un S.I.O.:

Le passage d'un état à un autre nécessite obligatoirement une transition indiquant à minima les activités permettant ce changement d'état.

Une transition n'est pas obligatoirement déclenchée par un événement explicite; on peut alors considérer que le déclenchement de la transition

est implicitement liée à un événement décisionnel

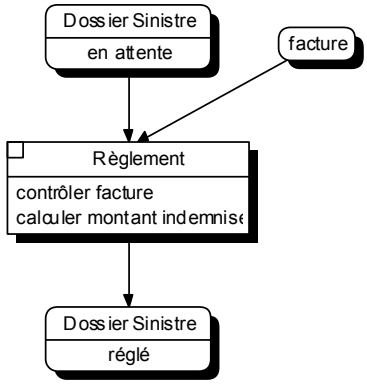


Figure 9.3 : Illustration des concepts du CVO Merise

La figure 9.3 illustre le passage de l'état « en attente » de l'objet « Dossier Sinistre » à l'état « réglé », passage déclenché par la survenance de l'événement « facture » et nécessitant la réalisation de l'activité « Règlement ».

Ce choix de réutilisation des concepts et des symboles est délibéré. Nous estimons qu'il n'est pas souhaitable dans une méthode (nous ne sommes pas ici dans le contexte d'une notation) de multiplier inutilement les formalismes. Nous avons déjà eu recours à un même formalisme pour spécifier les différents modèles fonctionnels de traitements; nous l'étendons désormais au modèle dynamique. Ainsi les MCT, les MOT, les MLT et les CVO constituent différentes modélisations de l'activité d'un système d'information organisationnel (SIO) ou informatisé (SII).

En fait, dans la modélisation des cycles de vie d objets, les concepts de modélisation restent identiques, seule l'approche de modélisation et donc l'agencement des différents concepts changent. Tout d'abord, on se préoccupe d'un objet (entité ou relation) pour lequel on recense l'ensemble des états possibles Ensuite, on analyse quand et comment (événement, condition) peut s'effectuer le passage (transition) d'un état à un autre, et quelles sont les activités associées à cette transition.

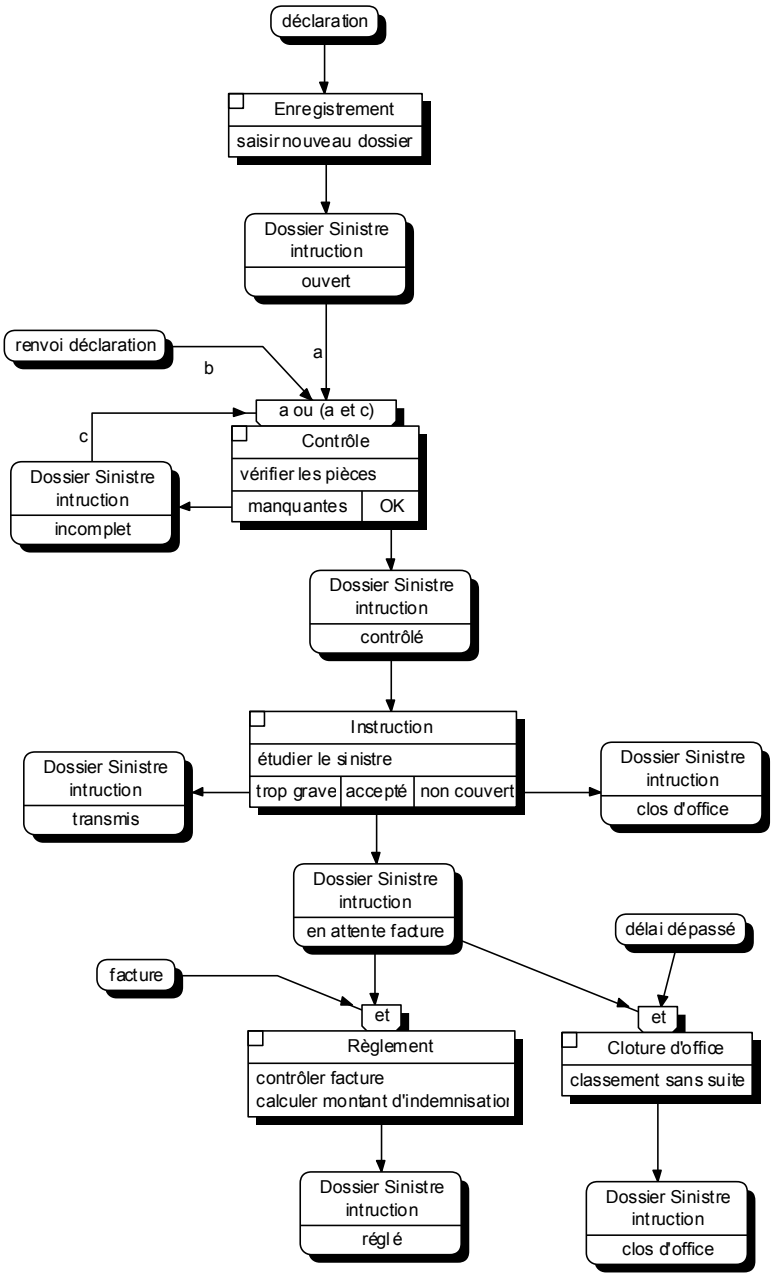


Figure 9.4 : Cycle de vie du Dossier de Sinistre

Un CVO est représenté graphiquement par l'ensemble des états de l'objet avec les transitions et les événements déclencheurs, comme l'illustre la figure 9.4 présentant le CVO de l'objet « Dossier Sinistre ».

L'analyse des états d'un objet peut parfois s'effectuer selon plusieurs *types d'états*, plusieurs cycles de vie. Par exemple, le Dossier de sinistre peut être analysé selon :

- Les différents états d'avancement de l'instruction (type d'état : instruction) : Ouvert, contrôlé, transmis, en attente facture, réglé, clos d'office,…
- Les différents états du suivi entre compagnie (type d'état : suivi compensation) : Non communiqué, déclaration transmise, évaluation transmise, régularisé,…

C.V.O., niveau d'abstraction et degré de détail

Les principes généraux de modélisation dans la méthode Merise reposent sur un découpage en niveaux d'abstraction (conceptuel, organisationnel, logique, physique) associé, dans le cas des activités, à un degré de détail (de global à détaillé) (cf Chapitre 3). Comment se situe la modélisation des CVO dans ce découpage ?

Dans la modélisation fonctionnelle des traitements, les états et les événements apparaissent dès le niveau conceptuel, associés aux opérations. Ils se reconduisent au niveau organisationnel où d'autre états et événements peuvent alors apparaître suite à la décomposition organisationnelle des opérations en tâches. Ces états d'origine organisationnelle traduisent des situations généralement transitoires dues à des choix de segmentation en tâches manuelles / informatisées ou de répartition dans des postes différents. Il faut également rappeler que le passage du niveau conceptuel au niveau organisationnel s'accompagne généralement d'un accroissement du degré de détail.

On peut donc distinguer deux niveaux de CVO dépendant essentiellement du niveau auquel sont pris en compte les états :

- Un CVO conceptuel est construit avec des états qui interviennent au niveau conceptuel de traitements. Ce sont des états stables, perçus par les acteurs de l'environnement.
- Un CVO organisationnel est construit avec les états exprimés au niveau organisationnel (intégrant donc ceux du niveau conceptuel). Ce niveau de CVO est donc nécessairement plus détaillé et riche en états. Certains de ces états ne sont perceptibles que par les acteurs internes au système (on les nomme souvent états intermédiaires ou transitoires).

Dans la pratique, afin de prendre en compte tous les états possibles, le CVO présenté est implicitement de niveau organisationnel. Nous retrouverons cette hypothèse dans la démarche d'élaboration d'un CVO.

Dualité entre la modélisation fonctionnelle des traitements et la modélisation du cycle de vie des objets

La modélisation conceptuelle puis organisationnelle des traitements ont mis en situation la quasi totalité des concepts du CVO (états, événements, activités), mais avec des points de vue différents. Aussi nous souhaitons mettre en évidence une certaine dualité et complémentarité existant entre la modélisation fonctionnelle (MCT, MOT) et le CVO tant dans l'expression que dans le processus d'élaboration :

- Dans le MCT / M.O.T., on modélise (approche fonctionnelle) sous la forme d'un processus ou d'une procédure, la succession des activités consécutives à un événement initial. Les états de divers objets apparaissent comme des conditions préalables ou des résultats conditionnels de ces activités. On s'intéresse essentiellement à l'analyse, à la répartition et aux ressources mises en oeuvre pour effectuer ces activités. Chaque schéma représente un processus ou une procédure. Seuls le degré de détail et la prise en compte des choix et contraintes organisationnelles introduisent une différence entre ces deux niveaux de modélisation.
- Dans le C.V.O., on modélise (approche dynamique) l'ensemble des états possibles d'un même objet. Les événements et les transitions expriment les conditions de passage d'un état à un autre. On s'intéresse au comportement d'un objet à travers tous les traitements qui peuvent lui être associés. Chaque schéma représente le cycle de vie d'un objet donné.
- Dans le MCT /M.O.T., l'événement et l'activité (opération ou tâche) sont les concepts principaux, l'état est un concept secondaire.
- Dans le C.V.O., l'état est un concept principal, l'événement et l'activité (transition) sont des concepts secondaires

Notons qu'un processus ou une procédure organisationnelle peut être considérée comme une partie du "cycle de vie" du S.I.O., c'est à dire un ensemble de traitements successifs à un flux et concernant plusieurs objets, à travers les états et les sous-schémas de données. Une procédure organisationnelle reste cantonnée dans un domaine ou sous -domaine. Le cycle de vie d'un objet réunit l'ensemble des traitements possibles sur un objet. Il peut ainsi s'étendre sur plusieurs domaines.

Elaboration d'un C.V.O.

On peut envisager deux démarches pour la construction d'un C.V.O. :

- Le C.V.O. est déduit de l'ensemble des traitements fonctionnels. Pour chaque objet, on extrait ses différents états modélisés dans les différents

processus ou procédures avec les activités et les événements associés. En considérant les activités (opération ou tâches) comme des transitions, on reconstitue dans un schéma unique les traitements (transitions) conduisant à un changement d'état. Une telle démarche suppose que la modélisation fonctionnelle soit complète.

- Le C.V.O. est construit directement. Pour chaque objet issu du M.C.D. / M.O.D. et pour lequel on perçoit des états possibles, on recense les états puis on exprime les événements et les transitions associés.

Pratiquement, les modèles MCT/MOT et CVO peuvent être élaborés alternativement, chacun venant enrichir et vérifier l'autre.

- La construction du MCT/M.O.T., par la prise en compte des états, met en évidence des traitements associés (en condition préalable ou en résultat) qui permettent d'alimenter par déduction la construction du C.V.O.
- La construction du C.V.O., par la recherche exhaustive des états d'un objet, oblige à exprimer des événements et des activités qui auraient probablement échappé à une analyse fonctionnelle, et conduit ainsi à un enrichissement du MCT/M.O.T.

Cette complémentarité entre MCT/ M.O.T. et C.V.O. contribue à la qualité de la modélisation des traitements.

En pratique, on n'élaborera des C.V.O. que pour les objets qui présentent un ensemble d'états significatifs, on considère que les états triviaux (créé, modifié, supprimé) ne présentent qu'un intérêt très limité au niveau S.I.O.

En conclusion, la modélisation du cycle de vie des objets, apparaît tout à fait adaptée au cadre de la méthode Merise. Elle vient judicieusement combler une lacune face aux méthodes de conception orientées objets la dotant ainsi d'un nouvel atout pour affronter le challenge des nouvelles évolutions méthodologiques.

Objets métiers

La notion d'objet métier est progressivement apparue dans la méthode Merise au début de la décennie 90 comme une extension de la modélisation conceptuelle des données. Elle a été par ailleurs mise en avant à partir des méthodes objets, par différentiation avec les objets logiciels[1]. Enfin, cette notion est souvent reprise en tant qu'offre commerciale de composants

[1] Il est plaisant de lire dans [KET 98] que « Pour mette en avant l'importance des objets métiers, UML a défini un stéréotype particulier permettant de les représenter : il s'agit de la notion d'entité ».

logiciels adaptés à un métier et prêts à l'emploi.

Cette notion d'objet métier présente pour le moins de multiples facettes. Dans le cadre de la méthode Merise, elle ne bénéficie pas encore de suffisamment de retour d'expérience pour constituer une base méthodologique éprouvée.

Toutefois dans le cadre de la quatrième édition de cet ouvrage, il nous est apparu judicieux de présenter tout d'abord deux différentes perspectives méthodologiques associées à l'objet métier, d'une part la macro-modélisation et d'autre part la modélisation intégrative. Ensuite dans la perspective de la macro-modélisation, qui nous apparaît actuellement la plus praticable, nous introduirons un certain nombre d'éléments méthodologiques permettant la conception d'objets métiers dans le cadre de Merise. Ces éléments nous semblent ouvrir une voie opérationnelle que nous espérons intéressante.

L'objet métier : une macro-modélisation de données

La modélisation en termes d'entités et de relations proposées dans le MCD est certes centrée autour « d'objets d'intérêt pour l'utilisateur », mais elle reste encore fortement contrainte par des règles de modélisation (en particulier celle de vérification). Ces règles sont nécessaires pour obtenir une certaine rigueur de modélisation indispensable pour permettre une informatisation (passage du SIO au SII), mais elles déroutent parfois l'utilisateur d'une perception plus « naturelle ».

Prenons l'exemple classique de la notion de Commande où, quelle que soit la modélisation adoptée, les lignes composant cette commande, sont naturellement intégrées par un utilisateur dans cette notion de Commande.

La perception de l'utilisateur tend ainsi a regrouper autour d'un concept majeur, généralement représenté par une entité conceptuelle de même nom, des relations et des entités fortement liées sémantiquement (du point de vue du métier, les lignes d'une commande font partie intégrante d'un commande).

L'objet métier apparaît ainsi comme une agrégation nommée d'entités et de relations, perceptible comme un tout par le métier. L'objet métier permet une modélisation macroscopique, sa composition détaillée en termes d'entités et de relations résultant du respect des règles de formalisation conceptuelle (figure 9.5).

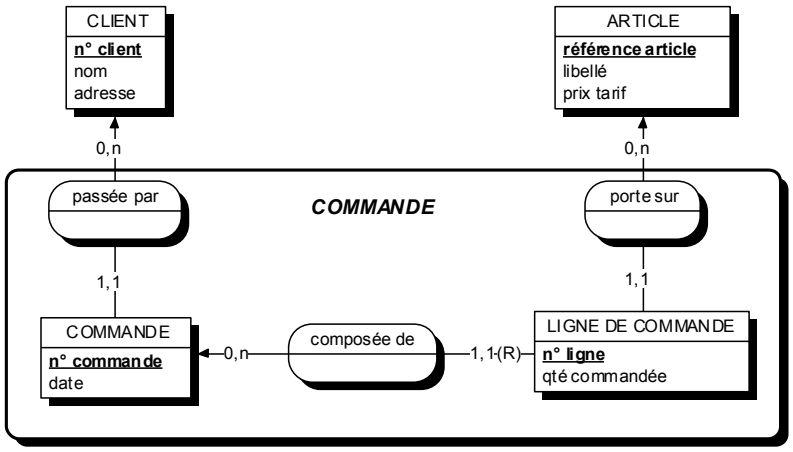


Figure 9.5 : L'objet métier Commande vu comme une macro-modélisation de données.

L'objet métier en tant qu'agrégation ou macro-modélisation a été initialement proposé dans le cadre des évolutions de la méthode Merise par P.A. Bres sous le concept d'objet « naturel » [Bres 91]. Dans cette approche, la modélisation des objets métiers est obtenue par un découpage disjoint (partitionnement) du modèle de données (en général de niveau conceptuel) et représente une macro-modélisation des données qui offre ainsi un « effet de zoom » sur les données, contextuel à un métier, analogue à celui de la décomposition hiérarchique (raffinement) présente dans les traitements.

Notons que cette approche est également introduite dans la méthode P+ (méthode voisine de Merise et proposée au Québec par la Sté DMR) où les « sujets » (objets métiers) se décomposent en « facettes » modélisées en entité – relation.

Cette expression des objets métiers comme une macro-modélisation construite sur le modèle conceptuel de données semble être actuellement la plus communément admise. Aussi, l'avons nous retenue comme approche de conception des objets métiers.

L'objet métier : une modélisation intégrative

Dans la méthode Merise, un concept nommé du monde réel peut se retrouver modélisé sous le même nom, de différentes façons, selon le type de modélisation. Par exemple dans le domaine de l'assurance :

- *L'assuré* est modélisé comme un *acteur* dans un diagramme de flux et un MCT, et comme une *entité* dans un MCD.
- *Le sinistre* se retrouve modélisé (avec des nuances d'appellation) sous la forme d'une *entité* , d'un *état*, de différentes évolutions de messages, probablement comme une *vue externe* et enfin présenté au travers d'une

maquette.

Le concept d'*objet métier* peut alors recouvrir, pour un même concept du monde réel, l'ensemble des aspects de la modélisation : sa définition statique, son comportement dynamique, son utilisation fonctionnelle (cf. figure 9.1). L'objet métier apparaît alors comme l'intégration de diverses facettes d'un concept du métier qu'explicitent les différentes modélisations formalisées :

- sa modélisation de données (MCD),
- son fonctionnement (MCT, MOT),
- son comportement (CVO),
- ses présentations (maquettes d'IHM).

L'objet métier, intégrant les diverses facettes ou composantes modélisées, devient alors une « superstructure » des modélisations classiques de la méthode Merise. On remarquera que la vision du paragraphe précédent (macro-modélisation de données) n'est alors qu'une des facettes de la décomposition de l'objet métier intégrateur.

Cette vision intégrative de l'objet métier présente de nombreux intérêts, notamment de :

- favoriser une approche modulaire de l'étude des SIO,
- reprendre des techniques de modélisation connues tout en réduisant l'étendue,
- préparer la constitution de composants métiers réutilisables.

Cette approche de l'objet métier en tant que modélisation intégrative nous apparaît plus ambitieuse et plus prometteuse que celle d'une macro-modélisation. Cependant, elle nécessite encore bien des réflexions, des expérimentations et la mise au point d'outils logiciels l'instrumentant pour pouvoir constituer un véritable apport méthodologique opérationnel.

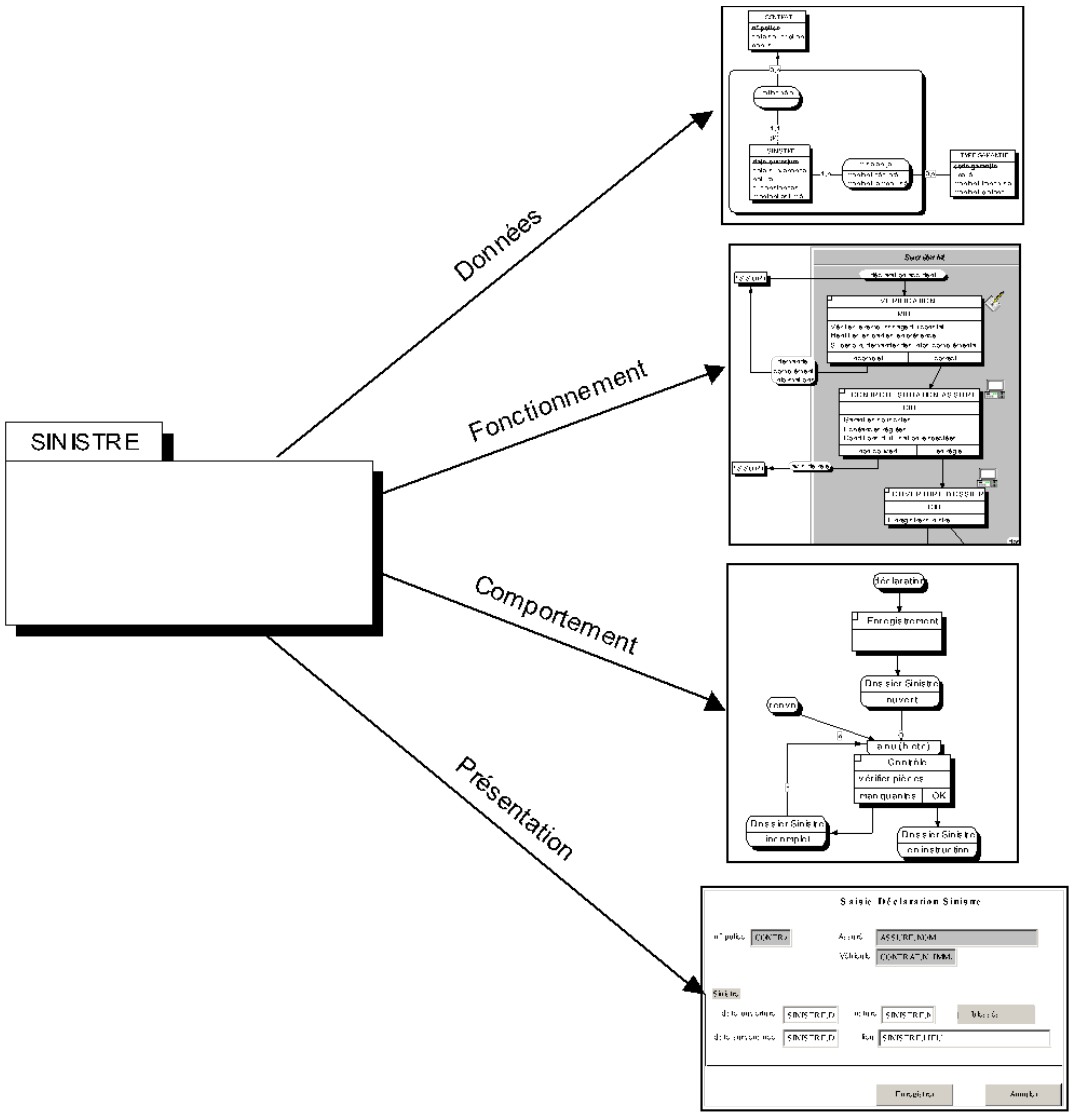


Figure 9.6 : L'objet métier Sinistre vu comme une intégration des modélisations

Conception des objets métiers

Nous venons de distinguer deux perspectives méthodologiques différentes de l'objet métier dans la conception de systèmes d'information organisationnel (SIO) : la macro-modélisation et la modélisation intégrative. Cette dernière nous apparaît encore en gestation. Par contre, il est dès à présent possible de proposer, dans le cadre de Merise, des éléments méthodologiques permettant tout d'abord la modélisation d'objets métiers et ensuite d'accompagner le

concepteur dans l'élaboration de ces modèles en lui proposant une démarche spécifique.

Formalisation des objets métiers dans Merise

Issue d'une macro-modélisation, un objet métier formalise un regroupement nommé d'entités et de relations d'un modèle conceptuel de données, fortement liées sémantiquement et permettant une définition globale de son contenu et de son comportement.

Le symbole retenu pour représenter un objet métier s'inspire (dans le graphisme et la construction) de celui de paquetage dans UML comme l'illustre la figure 9.7 suivante :

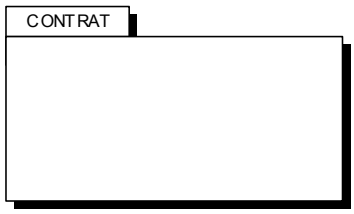


Figure 9.7 : Représentation pliée de l'objet métier

L'objet métier peut être représenté sous deux formes :

objet plié : seul le symbole de l'objet métier est représenté (Figure 9.7),

objet déplié : le contenu de l'objet métier en termes d'entités et de relations est représenté dans le symbole (Figure 9.8)

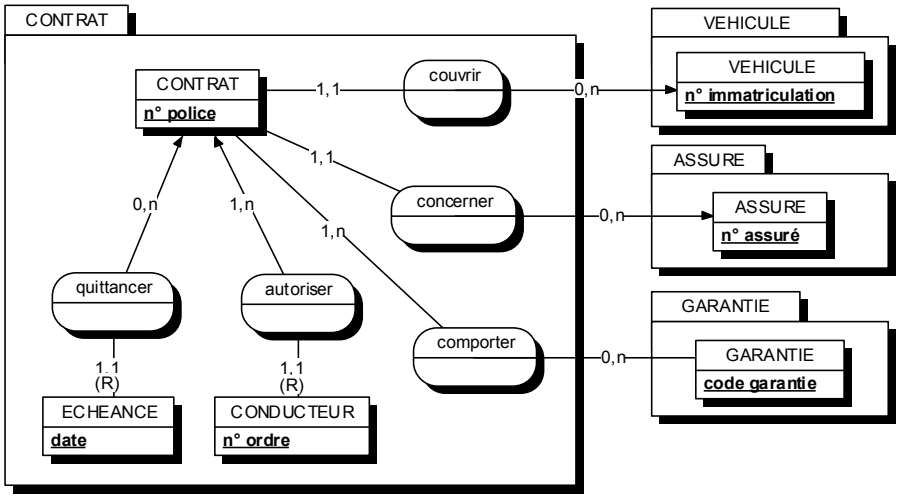


Figure 9.8 : Représentation dépliée de l'objet métier

Entité racine d'un objet métier

On appelle *racine d'un objet métier*, l'entité composante ayant la plus grande

affinité sémantique avec l'objet métier, qui en exprime l'essentiel de sa signification. Cela se concrétise généralement par l'identité de nom entre l'objet métier et son entité racine. Par exemple, dans la figure 9.8, l'entité Contrat est considérée comme la racine de l'objet métier Contrat.

Cette assimilation sémantique entre l'objet métier et son entité racine implique une même distingabilité d'occurrences. Ainsi, une occurrence de l'objet métier Contrat équivaut à une occurrence de l'entité Contrat. Bien qu'un objet métier ne soit pas explicitement doté d'un identifiant, on voit bien que la problématique de l'identification de l'entité racine (ou le choix de cette entité racine) conditionne la perception et la compréhension de l'objet métier.

Les autres entités, ainsi que les relations composant l'objet métier, sont qualifiées de secondaires et apparaissent « encapsulées » dans l'objet métier.

Lien entre objets métier

La construction d'un objet métier par regroupement d'entités et de relations conduit à un découpage du MCD selon des règles qui seront précisées dans le paragraphe suivant. Ce découpage passe nécessairement sur des pattes de relations qui lient une relation appartenant à un objet métier à une entité appartenant à un autre objet métier.

Le *lien entre objets métier* est équivalent à la patte « coupée ». C'est par ce lien entre objets métier que s'effectue la continuité sémantique exprimée dans le MCD, concrétisant le fait que les objets métier ne sont pas indépendants. Le lien d'un objet métier (origine, référant) vers un autre objet métier (référé) exprime que, dans l'objet métier origine, on utilise des données présentes dans -ou fournies par- l'objet métier référé.

Dans le cas d'une représentation pliée des objets métiers, on représente ces liens sous la forme d'une flèche en pointillé rappelant le lien de référence à un autre objet métier comme l'illustre la figure 9.9.

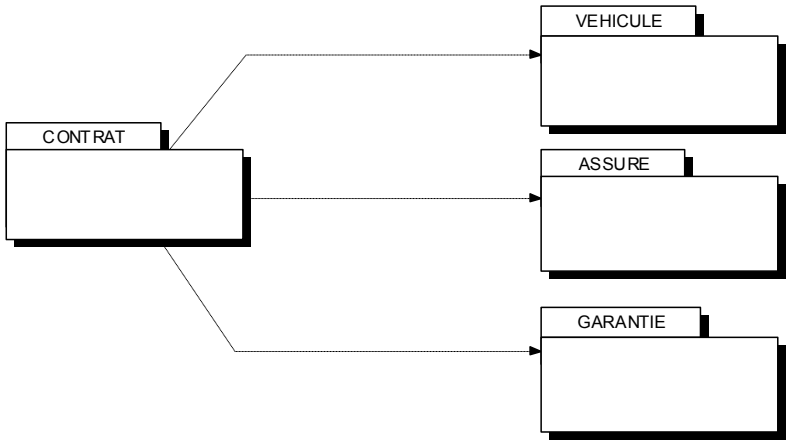


Figure 9.9 : Liens entre objets métiers

Entité externe dans un objet métier

La modélisation en termes d'objets métiers, sous la forme pliée ou dépliée, montre la nécessité de liens entre les objets métiers. Or, l'approche objet métier a pour objectif la recherche d'une modularité visant à réduire l'étendue de la modélisation conceptuelle des données en la bornant aux entités et relations sémantiquement regroupées autour du concept métier.

Il apparaît alors nécessaire, dans une modélisation conceptuelle limitée au contenu d'un objet métier (représentation dépliée autonome), de distinguer :

- les entités et les relations *propres* ou internes à l'objet métier,
- des entités *externes* qui évoquent des données d'autres objets métiers mais dont le contenu est nécessaire et utilisé dans le cadre de l'objet métier modélisé.

Dans la description conceptuelle des données d'un objet métier, l'entité externe représente ainsi la perception du contenu partiel d'un autre objet métier, vu à partir de l'objet métier étudié. Sa formalisation et ses principes de construction sont identiques à ceux définis pour l'entité externe dans le cadre des vues externes (cf. chap. 11).

Dans la représentation dépliée autonome d'un objet métier (limitée à son contenu), une entité externe :

- se symbolise comme une entité avec un style graphique distinctif (pointillé, non ombré),
- comporte des propriétés externes respectant la règle de vérification,
- ne possède par obligatoirement d'identifiant,
- participe à des relations avec des entités internes à l'objet métier,

- fait référence à un seul objet métier, plusieurs entités externes pouvant faire référence à un même objet métier,
- peut s'appeler comme l'objet métier référencé, mais aussi utiliser une autre appellation exprimant la perception externe de l'objet métier.

Comme dans le contexte de la modélisation des vues externes, la spécification d'une entité externe au sein d'un objet métier ne cherche pas à définir des entités conceptuelles. Elle se borne à exprimer, dans un formalisme unique, un ensemble d'informations nécessaires à l'objet métier étudié, en provenance d'un autre objet métier, sans que son contenu soit structuré et conceptualisé. Il ne s'agit pas de reconstituer la structure conceptuelle qu'auraient ces informations dans l'objet métier référé.

La figure 9.10 présente la vision dépliée autonome de l'objet métier Sinistre avec:

- l'entité externe Contrat qui réfère à l'objet métier Contrat,
- l'entité externe Garantie qui réfère à l'objet externe Garantie.

Dans l'entité externe Contrat, on remarquera que certaines des propriétés externes évoquant l'assuré et le véhicule sont perçues comme provenant de l'objet métier Contrat, sans préjuger de leur réelle appartenance directe à cet objet. L'entité externe Contrat est la vision qu'a l'objet métier Sinistre de l'objet métier Contrat.

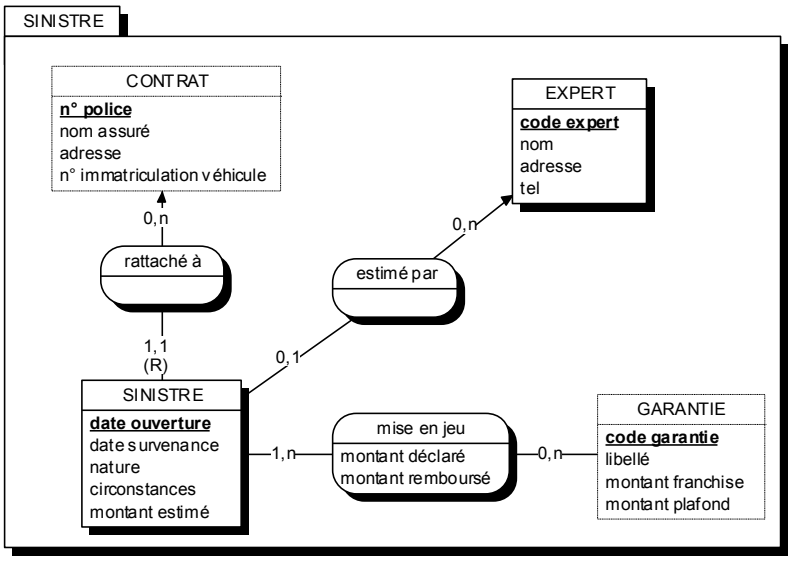


Figure 9.10 : Description dépliée autonome de l'objet métier Sinistre avec ses entités externes

Dans le processus d'élaboration des objets métiers, il faut s'assurer que toute

entité externe a une correspondance sémantique (structure et contenu) dans la description conceptuelle des données de l'objet métier auquel elle se réfère. Pour valider cette correspondance, on appliquera les mêmes règles que celles définies pour la confrontation détaillée des vues externes (cf. chap. 11 : Algorithme de confrontation détaillée) dont nous résumons les principes :

Equivalence entre propriété externe et propriété conceptuelle :

- directement,
- via une relation,
- via une règle.

Correspondance entre entité externe et sous-ensemble conceptuel :

- reconstitution du sous ensemble conceptuel portant les propriétés conceptuelles obtenues par équivalence,
- détermination du « pivot » du sous-ensemble, entité conceptuelle dont l'occurrence (distingabilité) correspond à celle de l'entité externe.

Représentations de la modélisation en objets métier

Comme nous venons de le constater, on dispose de plusieurs présentations graphiques possibles de la modélisation en objets métiers. Ces diverses représentations, que nous déclinons ci-après, permettent de mettre en évidence tantôt une présentation générale de l'ensemble des objets métiers inter-reliés (vision macroscopique), tantôt une description plus détaillée d'un objet métier (vision dépliée autonome), tantôt une superposition du découpage en objets métiers sur un modèle conceptuel de données (vision explicative).

Suivant le besoin et le contexte, le concepteur utilise la représentation qui lui semble la plus adaptée.

Schéma général d'objets métiers

Dans un tel schéma, l'ensemble des objets métiers identifiés dans le domaine étudié sont représentés sous forme pliée avec leurs liens entre objets métiers. La figure 9.9 est une illustration d'une telle présentation.

Cette présentation offre une vision générale « panoramique » et macroscopique des objets métiers et de leurs inter-relations. C'est cette représentation qui est usuellement nommée Modèle d'Objets Métiers (MOM) [2]

[2] Rappelons, qu'à l'instar de toutes les autres modélisations de Merise, il y a souvent confusion entre le modèle (définitions, dans un formalisme, de l'ensemble des concepts perçus) et une représentation graphique symbolisée (intégrale ou partielle) de ce modèle.

Schéma détaillé d'un objet métier

Il s'agit de la représentation dépliée autonome d'un objet métier dont le contenu sémantique détaillé est modélisé conceptuellement en termes d'entités (propres et externes) et de relations. Les figures 9.10 et 9.13 sont des illustrations d'une telle présentation.

Dans la pratique, on établira un schéma distinct pour chaque objet métier. On peut alors éventuellement faire l'économie de la représentation du symbole de l'objet métier.

Schéma de découpage en objets métiers

Ce schéma est en fait un modèle conceptuel de données sur lequel on fait figurer le contour des objets métiers sous forme évidemment dépliée. Les figures 9.8 et 9.12 sont des illustrations d'une telle représentation.

Un tel schéma est plutôt destiné à expliquer le découpage en objets métiers opéré sur un modèle conceptuel de données. Dans la pratique, il devient assez difficile de respecter la symbolisation de l'objet métier qui prend rapidement une forme polygonale. De plus, cette représentation atteint très vite des limites de taille et de lisibilité qui conduisent le concepteur à fractionner ces schémas en se focalisant, pour chacun, sur un objet métier.

Elaboration des objets métiers

Elaborer un objet métier c'est, d'une part le définir, d'autre part décrire son contenu (en termes de données dans le contexte de l'approche macro-modélisation retenue).

Deux démarches peuvent être adoptées pour l'élaboration d'objets métier :

- une *démarche par composition*, qui fait apparaître les objets métiers à partir d'une modélisation conceptuelle de données, en appliquant certaines règles de découpage .
- une *démarche par décomposition* qui consiste à définir d'abord les objets métier du domaine étudié, à formaliser le contenu conceptuel des données de chacun, puis à reconstituer les liens entre objets métier permettant de vérifier une continuité et une compatibilité sémantique au niveau de la modélisation conceptuelle des données de l'ensemble du domaine étudié.

Ces deux approches sont complémentaires et l'ont peut, selon la complexité et l'étendue du domaine étudié, préférer l'une ou l'autre approche, voire mixer les deux. Notons que cette dualité d'approche est similaire à celle déjà rencontrée pour l'élaboration d'un MCD : déductive ou inductive (cf. chap. 7).

Elaboration d'objets métier par composition

Dans cette approche, la détermination des objets métier est réalisée par un découpage du MCD, en regroupant des entités et relations autour d'entités racines « candidates ». Elle nécessite l'existence préalable d'une modélisation conceptuelle des données sur l'ensemble du domaine étudié, à partir de laquelle sont déterminés les objets métiers. On peut qualifier cette approche de « bottom – up ».

Pour réaliser ce découpage, nous proposons une démarche[3] heuristique, aidée par l'application de règles qui, bien que non formelles, guident le concepteur dans l'identification des objets métier. Cette démarche s'articule autour des étapes chronologiques suivantes :

1/ Choisir des entités racines candidates

On retient, parmi les entités du MCD, celles dont le nom évoque un concept important dans le discours du métier.

2/ Etendre autour des entités racines

A partir d'une entité racine, agréger les entités que l'on peut considérer comme structurellement et sémantiquement sous la dépendance de l'entité racine. Ces entités sont considérées comme internes à l'objet métier, ainsi que les relations qui les relient.

3/Délimiter la frontière de chaque objet métier

Poursuivre l'extension de l'objet métier jusqu'à rencontrer une entité que l'on considère comme n'appartenant pas à cet objet métier. Ce choix ne repose pas sur des règles formelles mais plus sur le choix du concepteur interprétant la perception du métier. Indiquons cependant quelques principes pour appréhender la frontière de l'objet métier :

- On ne peut intégrer dans un objet métier une entité qui serait déjà intégrée dans un autre objet métier, en particulier une entité racine candidate.
- Les entités sous-types devraient appartenir au même objet métier que leur sur-type.

4/ Déterminer le lien entre les objets

Il s'agit de déterminer si la relation impliquée à la frontière de l'objet métier considéré appartient ou non à cet objet. Cette frontière coupe nécessairement une des pattes de la relation qui correspondra au lien entre objets métier. Pour ce choix de découpage, on s'appuie sur les règles illustrées par le tableau suivant :

[3] Cette démarche s'inspire en partie des travaux de P.A Bres sur les objets naturels et de la SEMA dans le cadre de Merise/2

Conventions :

- Le trait vertical positionne la coupure de la frontière .
- L'une des entités est interne (appartient à l'objet métier à délimiter).
- L'autre entité est étrangère (appartient à un autre objet métier).
- La règle sur les cardinalités est symétrique selon le positionnement de l'entité interne.
- Pour les relations de dimension > 2, on applique la règle pour chaque couple de pattes.

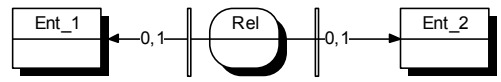
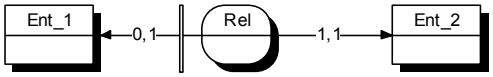
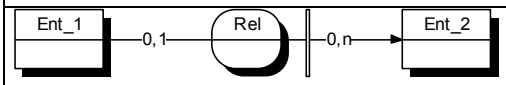
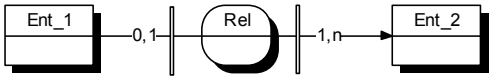
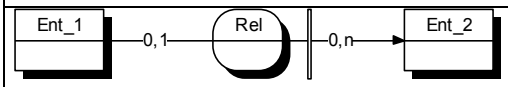
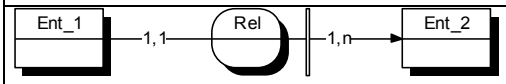
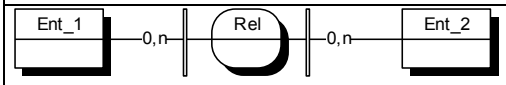
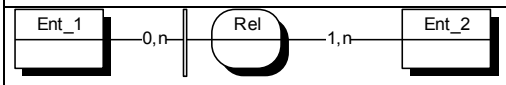
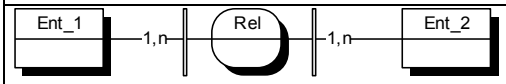
	La coupure est indifféremment possible d'un coté ou de l'autre
	On doit cependant s'interroger sur la pertinence de la délimitation de la frontière au niveau de cette relation
	
	Bien que possible des deux cotés, la coupure se fait de préférence coté Ent_2
	
	
	La coupure est indifféremment possible d'un coté ou de l'autre
	
	La coupure est indifféremment possible d'un coté ou de l'autre

Figure 9.12 : Règles de découpage d'une relation « frontière »

5/ Etablir les entités externes

Afin de rendre relativement autonome chaque objet métier déterminé et de permettre une représentation dépliée autonome, on matérialise, dans l'objet métier, la conséquence de la coupure de la patte de la relation frontière, par une entité externe correspondant a minima à l'entité étrangère impliquée dans la

relation frontière. Ultérieurement, le concepteur pourra enrichir cette entité externe avec de nouvelles propriétés externes, voire définir de nouvelles entités externes.

A l'issue de ce processus de composition des objets métiers, il convient de vérifier globalement la qualité du résultat en appliquant les principes suivants qui relèvent plus de la « bonne pratique » que de règles formelles :

- *Pertinence* : Chaque objet métier doit avoir un intérêt du point de vue du métier. Son nom doit être intelligible dans le discours utilisateur.
- *Consistance* : L'envergure de l'objet métier, en terme de contenu (entités, relations, propriétés) doit être consistant. On doit s'interroger sur un objet métier ne contenant qu'une entité décrite seulement par quelques propriétés.
- *Disjonction* : Une entité ou une relation conceptuelle ne peut appartenir qu'à un seul objet métier. Le découpage en objets métier réalise une partition du MCD.
- *Continuité* : Dans un objet métier, toutes ses entités sont atteignables à partir de sa racine (directement ou indirectement) via des relations. L'ensemble des entités et des relations forment un graphe connexe.
- *Granularité* : L'étendue d'un objet métier, en terme de nombre d'entités et de relations, doit rester raisonnable (inférieur à la vingtaine…).
- *Autonomie* : L'existence d'une occurrence d'un objet métier est autant que possible indépendante des autre objets métier. Dans l'application des règles de découpage, on privilégiera les relations « frontières » à cardinalités mini 0 sur l'ensemble des pattes. Des cardinalités mini à 1 induisent en effet une dépendance d'existence entre les objets métiers.

En illustration de cette élaboration d'objets métier par composition, la figure 9.12 présente un MCD découpé en objets métier (schéma de découpage en objets métiers). La figure 9.13 détaille le contenu de l'objet métier Contrat avec les entités externes a minima issues du découpage (schéma détaillé d'un objet métier).

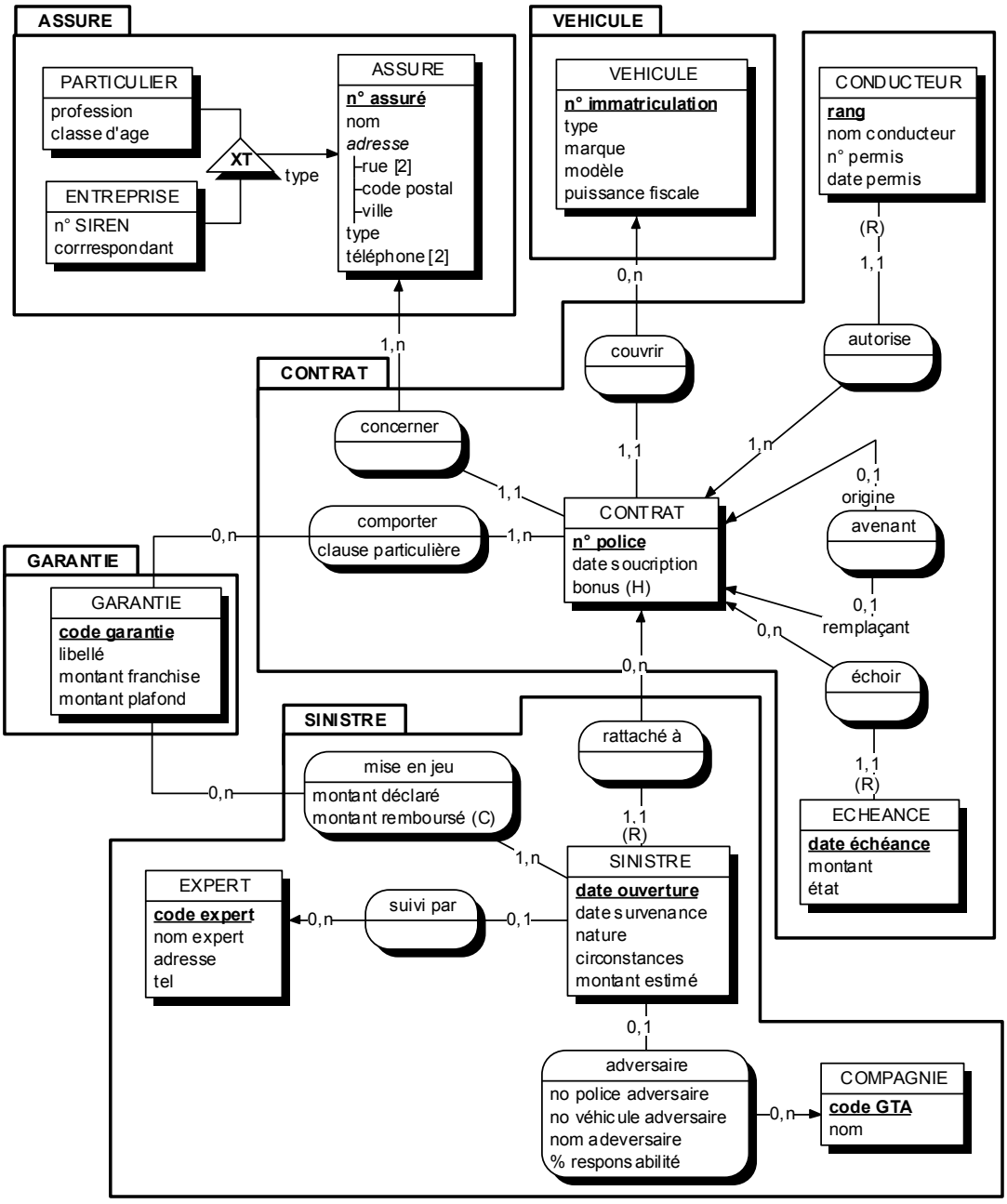


Figure 9.12 : Composition des objets métier à partir d'un modèle conceptuel de données
Schéma de découpage

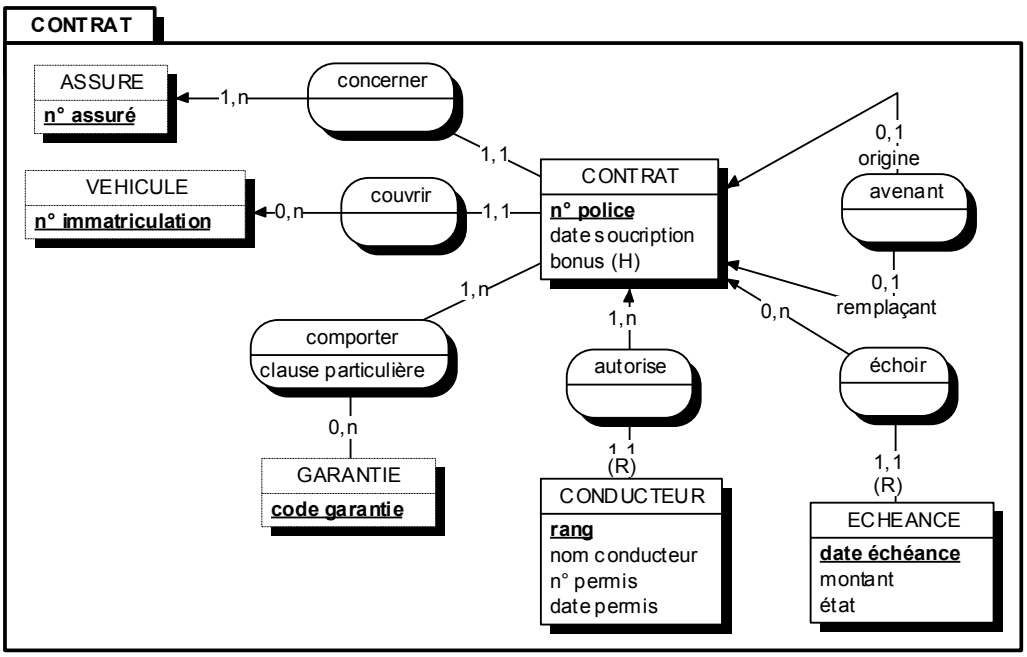


Figure 9.13 : Contenu de l'objet métier Contrat après découpage
Schéma détaillé

Elaboration d'objets métier par décomposition

Dans cette démarche, la détermination des objets métier est un choix a priori du concepteur en fonction de la pertinence perçue de la notion associée à cet objet dans le discours du métier. Le concepteur détaille ensuite le contenu des données de l'objet par une modélisation conceptuelle de données. On peut qualifier cette approche de « top - down ».

Cette modélisation du contenu des données conceptuelles de l'objet métier s'effectue selon les règles habituelles d'utilisation du formalisme Entité – Relation ; seule l'étendue de la modélisation est limitée à la frontière estimée de l'objet. Lorsque certaines données (entités et propriétés) sont considérées comme provenant d'un autre objet métier, elles sont modélisées sous la forme d'entités (et de propriétés) externes.

Cette démarche par décomposition produit une modélisation conceptuelle des données de chaque objet métier, en faisant toutefois référence à d'autre objets métier par l'intermédiaire des entités externes.

A l'issue de cette première étape de la démarche, le concepteur doit s'assurer de la recomposition possible et cohérente d'un MCD portant sur l'ensemble du domaine étudié, à partir des modélisations conceptuelles des données de chaque objet métier.

Validation par recomposition

Cette recomposition consiste à réunir l'ensemble des données modélisées des objets métiers et à reconstituer les liens entres les objets métiers qui théoriquement correspondent à des pattes de relations (cf. § Lien entre objets métiers). Cette validation s'appuie sur les entités externes définies dans chaque objet métier.

Pour chaque entité externe (cf. § Entité externe dans un objet métier), on applique l'algorithme de validation externe/conceptuel, par rapport à la modélisation conceptuelle des données de son objet métier de référence. Cet algorithme, qui est identique à celui appliqué pour la validation des entités externes dans les vues externes (cf. chapitre 11 : Validation algorithmique) permet de déterminer :

- Le sous-ensemble conceptuel correspondant à l'entité externe,
- L'entité pivot de ce sous-ensemble.

Le lien entre les objets doit correspondre à une patte de relation entre :

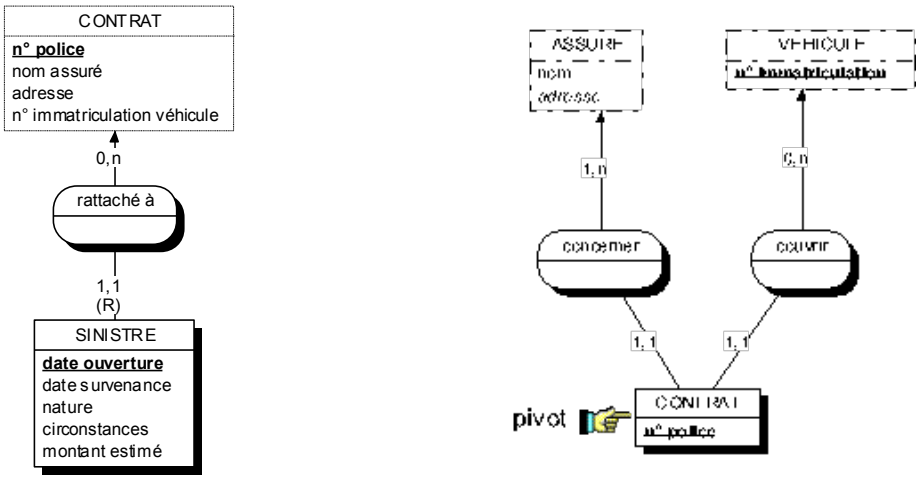
- La relation interne à laquelle participe l'entité externe,
- L'entité pivot déterminée dans l'autre objet métier.

On va ainsi reconstituer les pattes « coupées » qui apparaissaient dans une démarche par composition.

Illustrons cette démarche par un exemple :

- La figure 9.10, par une représentation déplié autonome ou schéma détaillé, propose une modélisation conceptuelle des données de l'objet métier Sinistre, avec notamment l'entité externe Contrat qui réfère à l'objet métier Contrat.
- La figure 9.14 présente la correspondance entre l'entité externe Contrat et son sous-ensemble conceptuel dans l'objet métier Contrat, ainsi que son pivot l'entité Contrat. On remarquera que ce sous-ensemble comporte des entités externes qui à leur tour devront être validées.
- On a donc bien reconstitué la patte de relation entre Rattaché à et Contrat qui apparaît bien comme coupée dans la figure 9.12 (schéma de découpage).

Cette démarche de recomposition ne vise pas nécessairement à reconstituer un MCD général sur l'ensemble des objets métiers, mais plutôt à valider les objets métiers deux à deux pour vérifier la cohérence de leurs liens d'utilisation à travers les entités externes.



Vue partielle de l'objet métier SINISTRE

Sous-ensemble conceptuel dans l'objet métier CONTRAT

Figure 9.14 : Correspondance entre entité externe et sous-ensemble conceptuel dans l'objet métier de référence

Conclusion sur la modélisation des objets métier

La modélisation en terme d'objets métier n'a été que succinctement abordée dans ce chapitre et uniquement à travers sa composante données de niveau conceptuel, selon une approche de macro-modélisation.

Les prolongements méthodologiques conduisant à une approche intégratrice, évoquée en introduction de la notion d'objet métier et consistant à prendre en compte les autres composantes de l'objet métier (comportement, fonctionnement, présentation) y compris pour les autres niveaux (organisationnel et logique), nécessitent des développements qui dépassent le cadre de cet ouvrage. L'approche objets métier peut désormais s'inscrire comme un complément et une évolution prometteuse de l'approche« merisienne » classique en ingénierie de systèmes d'information.

10

Modélisation organisationnelle des données

Problématique du modèle organisationnel de données (MOD)

Comme nous l'avons vu précédemment, la modélisation conceptuelle des données visait à représenter la signification des informations utilisées dans un domaine d'activité de l'entreprise sans tenir compte de contraintes organisationnelles, économiques ou techniques. Elle exprimait des objets concrets ou abstraits, des associations entre ces objets et des informations descriptives, formalisées en termes d'entités, de relations et de propriétés.

La modélisation organisationnelle des données va permettre de prendre en compte des éléments relevant de l'utilisation des ressources de mémorisation :

- Le choix des informations à mémoriser informatiquement.
- La quantification (ou volume) et la durée de vie des informations à mémoriser.
- La répartition des données informatisées entre unités organisationnelles.
- L'accès aux données informatisées pour chaque unité organisationnelle

Ces différentes préoccupations nous conduiront à définir deux niveaux de modélisation organisationnelle des données : le MOD *global*, directement dérivé du MCD, et les MOD *locaux*, spécifiques chacun à une unité organisationnelle. Les MOD locaux seront dérivés du MOD global en prenant en compte des choix d'organisation, en particulier de répartition.

Les modèles organisationnels de données s'expriment avec le même formalisme que le modèle conceptuel de données (entité - relation) auquel on ajoutera quelques notions complémentaires. Aussi, dans la première génération

d'utilisation de la méthode Merise, les concepteurs ont-ils pu sans difficulté considérer ce modèle organisationnel des données comme un affinement du modèle conceptuel des données, intégrant les conséquences des choix organisationnels cohérents avec le modèle organisationnel des traitements.

Le modèle organisationnel de données apparaît donc comme une représentation, exprimée avec le formalisme entité - relation, des informations qui seront mémorisées informatiquement compte tenu des volumes, de la répartition et de l'accessibilité, sans encore tenir compte des conditions de structuration, de stockage et de performance liées à la technologie de mémorisation informatique qui sera utilisée.

Il est important de distinguer :

- ✓ La répartition organisationnelle des données qui impacte directement les tâches de l'utilisateur à travers le besoin et la disponibilité des données (quelle que soit la localisation physique de mémorisation); cette répartition est exprimée par les MOD locaux,
- ✓ La répartition informatique des données, liée à l'implantation du stockage des données, éventuellement transparente à l'utilisateur et exprimée dans les modèles logiques et physiques de données.

Choix des données à mémoriser

Il s'agit de choisir, à partir des informations formalisées sur le MCD, celles qui devront être effectivement mémorisées informatiquement dans le système d'information informatisé (SII) (ou données informatisées). Notons que les autres informations seront mémorisées « manuellement » (support papier ou autre non informatique) mais feront toujours partie des informations constituant la mémoire du système d'information organisationnel (SIO).

Le modèle organisationnel des données ainsi obtenu est de niveau global; il ne prend pas en compte les choix d'utilisations réparties. Ce MOD dérive directement du MCD auquel on peut être conduit à :

- Supprimer des éléments (entités, relations, propriétés) qui ne seront pas mémorisés informatiquement.
- Modifier certains éléments (entités, relations, propriétés, cardinalités...) compte tenu du choix de mémorisation informatisé.
- Ajouter de nouvelles informations
  - pour permettre de faire le lien entre les données mémorisées et les données restées manuelles; par exemple la référence de fiches, de dossiers, d'un ensemble de mesures réalisées, de plans...
  - pour mémoriser des états du système d'information consécutifs

au déroulement des traitements dans le MOT.

La détermination des informations informatisées ou manuelles peut s'effectuer :

- Par rapport à l'intérêt de conserver telle information (dialogue avec l'utilisateur).
- Pour des motifs de volume (voir paragraphe suivant).
- Suite à l'utilisation d'informations exclusivement par des tâches manuelles (MOT et confrontation données traitements : grille de cohérence ou confrontation détaillée).
- Pour rester en cohérence avec des choix de types de ressource informatisée, effectués au niveau du modèle organisationnel de traitements.

Les praticiens de la méthode Merise retrouveront ici la plupart des modèles conceptuels de données qu'ils ont réalisés, tout en s'interrogeant parfois sur la pureté conceptuelle de leur modélisation. Sur le terrain, la confusion éventuelle entre le modèle conceptuel de données et le modèle organisationnel de données global, du fait de l'identité du formalisme utilisé, ne remet pas en cause le rôle primordial d'expression de la signification des informations utilisées et le dialogue avec les utilisateurs.

Quantification du modèle organisationnel de données

La quantification du MOD s'effectue essentiellement au niveau du MOD global. Elle consiste à préciser le type et la longueur des propriétés ainsi que les contraintes sur les valeurs et le nombre d'occurrences des entités et des relations

Cette quantification peut s'effectuer déjà au niveau conceptuel; c'est en fait une préoccupation constante du concepteur. Par exemple, l'évaluation du nombre d'occurrences d'une entité peut être un révélateur de la compréhension de cette entité. Notons que cette quantification est statique et détermine en première approximation le volume des données à mémoriser. Elle permettra entre autres de mieux évaluer les possibilités de répartition organisationnelle des données et en conséquence de dériver les MOD locaux.

Type et taille des propriétés

Le concepteur doit préciser le type de données ainsi que la taille des propriétés, en termes de nombre de caractères. Il peut recourir aux types standards utilisés informatique ou définir des types utilisateurs. Il indiquera enfin les domaines de valeurs à respecter par certaines propriétés.

Types standards

Au niveau du MOD, on doit a minima distinguer les types de données caractères et numériques, ainsi que le nombre de caractères significatifs (la longueur). Nous proposons, à défaut, la notation suivante :

- An (champ alphanumérique de n caractères).
- Nn (champ numérique de n chiffres — valeur entière).
- Nn.p (champ numérique de n chiffres significatifs dont p chiffres après la virgule).

On peut également s'inspirer de la typologie de données définie en informatique, et plus particulièrement dans les bases de données. On peut ainsi distinguer, avec plus ou moins de détail selon le contexte et les outils de modélisation utilisés, les types suivants :

- An (alphanumérique de n caractères)
- caractère (variable, long).
- Nn.p (champ numérique de n chiffres significatifs et p chiffres après la virgule - précision).
- entier (court, long), décimal, réel (double précision), monétaire.
- Dn (champ calendaire de n caractères significatifs)
- date, heure, date + heure.
- Sn (type spécial de n caractères significatifs)
- booléen, binaire, mémo, image, son, etc.

Ces longueurs indiquées ne permettent pas réellement une estimation du volume qu'occuperont ces données, car, suivant la technique de mémorisation retenue ultérieurement aux niveaux logique et physique, la taille en octets peut être très variable. De plus, selon certains systèmes, le choix du mode de représentation informatique peut être un paramètre d'optimisation. On peut cependant, pour chaque cible de base de données envisagée, se doter d'une règle d'équivalence entre les longueurs indiquées selon les types de données et leur occupation probable en octets.

Domaines de valeur

Il s'agit de contraintes que doivent respecter les valeurs d'une propriété. Celles ci peuvent porter sur :

des valeurs minimales, maximales,

des valeurs par défaut,

une liste limitée de valeurs à respecter .

Types utilisateurs

Un type utilisateur est une particularisation nommée de type de données pour lequel on fixe un type, une longueur et éventuellement un domaine de valeur.

Exemple :

Nom standard : A40

Ligne adresse : A60

Taux en % : N5.2

Oui Non : Booléen 0=non 1 = oui

Nombre d'occurrences des entités et relations

Il s'agit d'évaluer le nombre maximum d'occurrences des entités et des relations que l'on voudra avoir dans la future base pour constituer la mémoire immédiate du système d'information.

Mémoire immédiate, mémoire à long terme

Jusqu'à présent, dans la modélisation conceptuelle des données, aucune notion de leur durée de vie n'a été prise en compte; la mémoire du système d'information était considérée comme illimitée. Toute information, quelle que soit son ancienneté, était théoriquement disponible dès qu'elle était exprimée dans le modèle. Les préoccupations économiques du modèle organisationnel de données ne peuvent se satisfaire de cette hypothèse. On distingue alors deux mémoires (voir figure 9.1) :

- La *mémoire immédiate*, contenant l'ensemble des données immédiatement accessibles aux traitements; généralement « en ligne » sur les mémoires du système informatique.
- La *mémoire à long terme*, composée des archives intégrales ou condensées des informations anciennement présentes dans la mémoire immédiate; la disponibilité de ces informations sera rarement immédiate et présentera fréquemment une structure adaptée à l'économie de stockage.

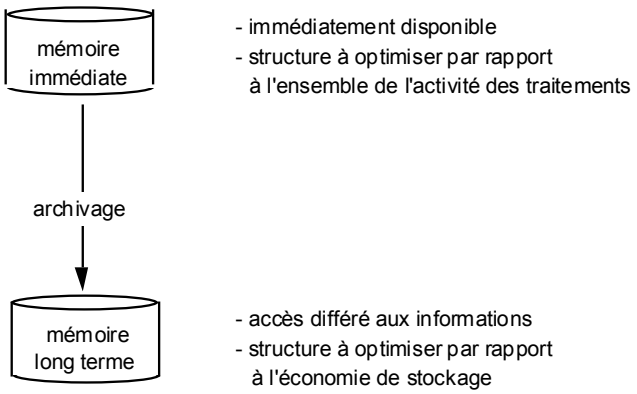


Figure 9.1 : Mémoire immédiate, mémoire à long terme.

Ces deux mémoires sont cohérentes et compatibles car elles proviennent toutes les deux des spécifications initiales du modèle conceptuel de données. Au moment de passer au modèle organisationnel de données, le concepteur doit se prononcer sur la « profondeur » de la mémoire immédiate du système d'information; donc définir la durée de vie des informations.

Durée de vie des entités et des relations

L'ensemble de la mémoire immédiate est constitué d'occurrences d'entités et de relations décrites par des propriétés. La durée de vie concerne celle des occurrences. On distingue, suivant les règles de détermination de la durée de vie, deux catégories qui s'appliquent aux entités et relations :

- Les entités ou relations à *durée de vie indéterminée* (ou permanentes) : la durée de vie d'une occurrence n'est pas déterminée lors de sa création; la suppression d'une occurrence dépend soit d'un événement, soit d'une règle indépendante de l'ancienneté de cette occurrence. Exemple : un compte clos depuis une année est à archiver.
- Les entités ou relations à *durée de vie limitée* : la durée de vie d'une occurrence est fixée lors de sa création, et généralement constante pour le concept. Exemple : les écritures sont archivées au bout de 6 mois.

Pour les concepts permanents, le nombre maximum d'occurrences présentes dans la mémoire immédiate tient compte de l'effectif au chargement initial et de la fréquence de création.

Pour les concepts à durée de vie limitée, le nombre maximum d'occurrences dans la mémoire immédiate est évalué par la fréquence de création de ce concept, multipliée par la durée de vie.

En fait, la détermination de la durée de vie des occurrences de concepts résulte d'un compromis entre :

la profondeur de la mémoire immédiate ou la disponibilité des informations;

le volume occupé par ces mêmes informations.

Conséquences de la durée de vie sur l'élaboration du MOD

Le modèle conceptuel de données a été élaboré sans tenir compte de la durée de vie associée aux concepts. Certaines informations calculées apparaissant dans des résultats n'ont pas été considérées comme devant être directement prises en compte dans le modèle conceptuel (voir figure 9.2).

La règle de calcul, construite sur des propriétés conceptuelles, permettant d'obtenir cette information calculée sera spécifiée dans les traitements (MCT ou plus probablement MOT puis MLT).

Par exemple, pour l'entité CLIENT, le chiffre d'affaires depuis le début de l'année est égal à la somme, sur toutes ses commandes passées depuis le début de l'année, du montant total de la commande (propriété éventuellement calculée elle aussi).

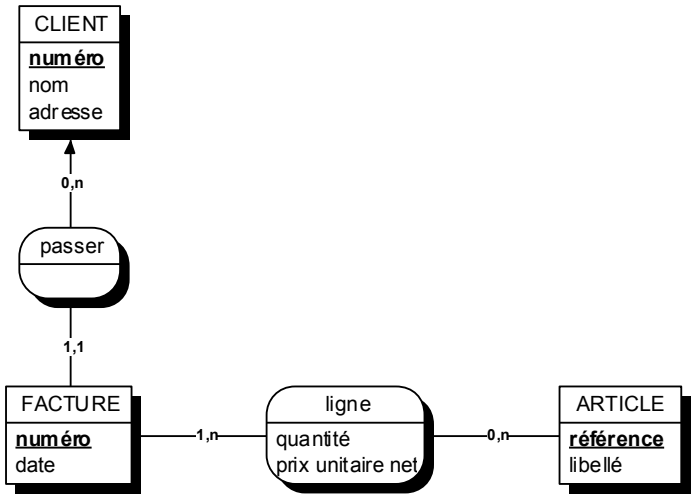


Figure 9.2 : MOD de référence sans durée de vie.

Durées de vie :

Client, article : permanents.

Facture, passer, ligne : 3 mois.

Règles de calcul :

Montant total commande = ∑ quantité livrée x prix vente.

Chiffre d'affaires annuel client = ∑ montant total commande.

Ainsi exprimée, cette règle permet toujours, sémantiquement, de calculer ce

chiffre d'affaires cumulé, sans en exiger la mémorisation.

Si la durée de vie des commandes est inférieure à un an (trois mois par exemple), il sera impossible de reconstituer ce cumul au-delà de la durée de vie des commandes.

Cette situation se reproduit chaque fois que des informations, termes d'une règle de calcul, ont une durée de vie inférieure à l'information résultante, non mémorisée.

On doit alors considérer cette information résultante comme une information primaire, et la modéliser comme une propriété d'une entité ou relation.

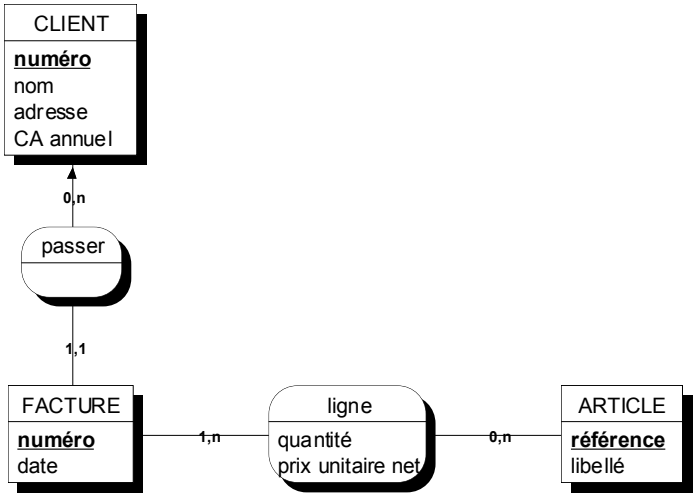


Figure 9.3 : MOD intégrant les redondances dues aux durées de vie.

Pour accueillir cette nouvelle propriété, le concepteur peut être conduit à

- affecter cette propriété à une entité ou à une relation existant déjà;
- créer une nouvelle entité type ou une nouvelle relation type;
- modifier la collection d'une relation déjà existante.

Ces modifications s'expriment sur le modèle organisationnel de données. Elles ne remettent cependant pas en cause la signification du modèle conceptuel initial.

En pratique, le concepteur élaborera son modèle organisationnel des données à partir du modèle conceptuel de données. Celui-ci est essentiellement sémantique; le modèle organisationnel de données intègre les redondances d'informations dues à la prise en compte de la durée de vie différente des informations (voir figure 9.3).

A cette occasion, le concepteur pourra également élaborer un autre modèle organisationnel des données pour la mémoire à long terme, ne comprenant que les informations que l'on désire archiver.

Etant donné la nature du problème, sa prise en compte dans la construction d'un modèle organisationnel de données interviendra plus vraisemblablement en étude détaillée.

Il ne faut pas confondre la notion d'archivage, liée à la durée de vie des informations dans la mémoire immédiate, avec la notion d'historisation lié à la conservation des valeurs antérieures des propriétés d'une occurrence d'entité ou de relation (voir « Historisation » au chapitre 7)

Les procédures organisationnelles et informatiques d'archivage doivent être spécifiées lors de l'étude détaillée.

Quantification des cardinalités

Le nombre d'occurrences des entités est généralement assez aisé à estimer; celui des relations est beaucoup plus difficile, compte tenu de la nature même des relations: des associations entre objets. Dans la plupart des cas, ce dénombrement s'effectuera à partir du nombre d'occurrences d'une entité de sa collection et de la quantification de la cardinalité de la « patte ».

Dans le modèle conceptuel de données, les cardinalités maximales multiples sont spécifiées jusqu'à présent par n. Il faut, au niveau du MOD, évaluer cette multiplicité. Prenons par exemple la modélisation de la figure 9.4.

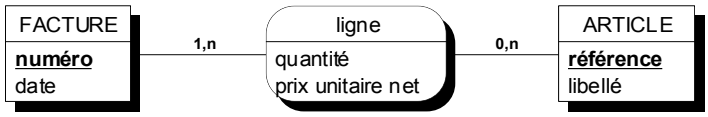


Figure 9.4 : Cardinalités conceptuelles.

La valeur de la cardinalité maximale n de Facture par rapport à ligne doit être précisée. La valorisation des cardinalités peut s'exprimer par :

- la cardinalité maximale,
- la cardinalité maximale à 95 %,
- la cardinalité modale,
- la cardinalité moyenne.

Le choix de telle ou telle valeur se fera selon l'intérêt dans le processus d'optimisation; dans tous les cas, au niveau du MOD, on définira la *cardinalité moyenne*, en particulier pour calculer le nombre d'occurrences des relations.

La quantification de la cardinalité mini s'exprime par le *taux de participation*. En

effet, au niveau conceptuel, la cardinalité mini peut prendre les valeurs 0 (optionnelle) ou 1 (obligatoire). Le taux de participation ne se détermine que pour les cardinalités mini à 0 et mesure le pourcentage d'occurrences de l'entité qui participent aux occurrences de la relation. Il prend donc une valeur décimale comprise entre {] 0, 1 [.}

La quantification de la *cardinalité moyenne* n'est pas toujours facile. Soit on fixe arbitrairement cette cardinalité, soit à partir de statistiques sur les populations d'entités concernées par la relation, il est possible de la déterminer de façon plus précise. Par exemple, pour l'exemple précédent, on pourrait avoir la distribution des lignes de facture par facture illustré par la figure 9.5.

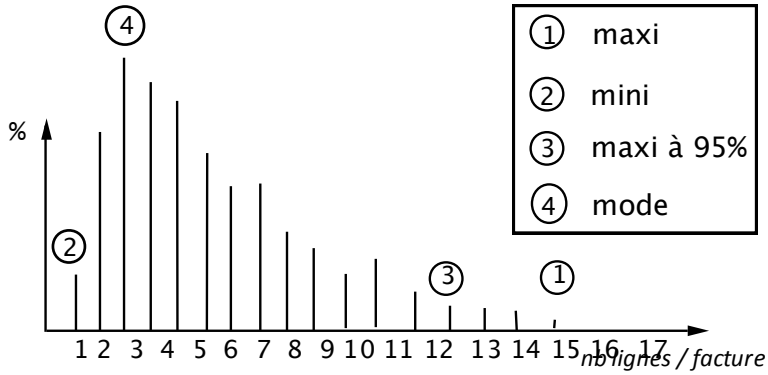


Figure 9.5 : Distribution nombre de lignes / facture.

Pour notre exemple, on peut supposer que la loi de répartition des cardinalités est approximativement triangulaire, comme le montre la figure 9.6.

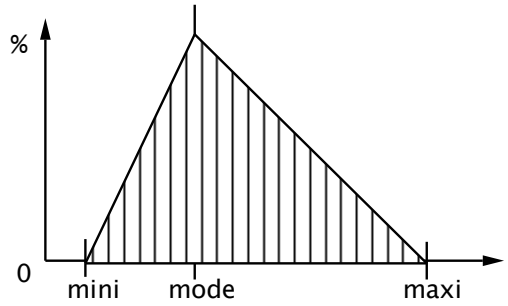


Figure 9.6 : Loi de répartition triangulaire.

Etant donné une telle répartition triangulaire, on peut calculer la cardinalité moyenne par la formule suivante :

cardinalité moyenne = [(m + 2M + N) / 4] x P

avec :

m = valeur mini (0 exclu)

M = valeur modale

N = valeur maxi

P = taux de participation

Par exemple, si M = 3; N = 17; m = 1; P = 1; on a :

cardinalité moyenne = ((1 + 2 x 3 + 17) / 4) x 1 = **6**

Cette loi de distribution triangulaire permet d'estimer les moyennes sur des répartitions dissymétriques (assez fréquentes dans la réalité) et donne d'assez bons résultats pratiques.

Notons que la cardinalité moyenne d'une cardinalité (1,1) est toujours 1 et que la cardinalité moyenne d'une cardinalité (0,1) est égale au taux de participation.

Dans le cas de plusieurs populations (voir figure 9.7), on recherche la moyenne de chaque population, puis on calcule la moyenne pondérée.

Par exemple :

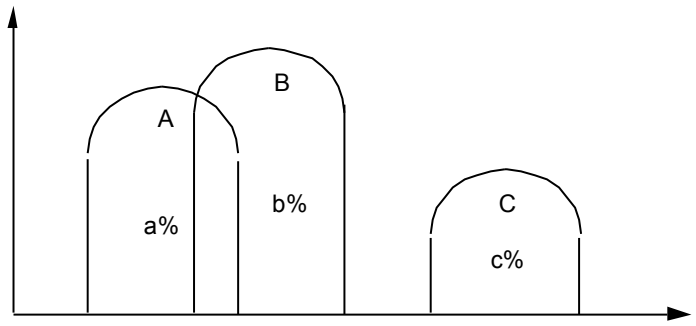


Figure 9.7 : Cas de plusieurs populations.

moyenne de la population A = VA, % de la population totale = a

moyenne de la population B = VB, % de la population totale = b

moyenne de la population C = VC, % de la population totale = c

moyenne pondérée = a.VA + b.VB + c.VC

Évaluation du volume global du modèle organisationnel de données

Le chiffrage se fait de proche en proche dans le modèle organisationnel de données. En effet si le concepteur n'a pas de problèmes majeurs pour définir la taille des propriétés, il peut avoir plus de difficultés pour estimer la population

des entités, des relations et la cardinalité moyenne des relations, n'ayant pas toute l'information nécessaire. Il est alors conduit à tirer parti au maximum des estimations dont il dispose pour chiffrer son modèle.

La quantification des propriétés, entités relations et cardinalités est généralement présentée sous forme de tableaux où figurent les différentes valeurs estimées (voir figure 9.8).

ENTITE - Relation	Total taille propriétés	Nombre d'occurrences	Volume
FACTURE	12	1 500	*18 000*
ARTICLE	40	2 000	*80 000*
ligne	10	*9 000*	*90 000*

RELATION	ENTITE	cardinalité moyenne	taux de participation
ligne	FACTURE ARTICLE	6 *4,5*	1 0,9

Figure 9.8 : Tableaux de chiffrage d'un MOD

A partir des éléments quantifiés précédemment déterminés, le concepteur peut procéder à une évaluation brute du volume total des données à mémoriser, pour la mémoire immédiate.

Ce chiffrage en volume brut n'estime que l'occupation des données utiles. Il ne tient pas compte du volume résultant du mode de mémorisation des pattes entre entités et relations. Cet aspect dépendra du mode de mémorisation informatique et sera chiffré au niveau du modèle logique de données en tenant compte de l'optimisation et des caractéristiques techniques du système de gestion de base de données utilisé.

Il faut considérer cette estimation en volume brut comme une valeur minimum pour l'occupation ultérieure du volume informatique. Elle peut cependant, en introduisant un facteur multiplicatif (1,5 à 2,5 suivant les technologies), fournir une indication pour le volume de la base de données résultante.

Répartition organisationnelle des données

Jusqu'à présent, nous avons raisonné sur un MOD global, très proche dans son contenu du MCD. Il représente l'ensemble des données à mémoriser utilisables dans le domaine d'activité étudié.

La modélisation organisationnelle des données va également se préoccuper de

la répartition d'utilisation de ces données suivant les différentes unités organisationnelles.

La connaissance de cette répartition organisationnelle des données présente un intérêt certain pour orienter ultérieurement la répartition informatique des données, en particulier dans des environnements clients / serveurs.

MOD local à une unité organisationnelle

L'unité organisationnelle recouvre généralement un ensemble de postes représentant par exemple un service ou un site géographique. Les utilisateurs d'une unité organisationnelle ont une vue commune et partagée d'un ensemble de données : le MOD local. Le M.O.D. local et l'unité organisationnelle sont donc un moyen d'exprimer, du point de vue de l'utilisateur, les données accessibles par un ensemble de postes.

Le MOD local est un sous-ensemble du MOD global en termes :

- d'entités-types, de relations-types et de propriétés
- d'occurrences d'entités ou de relations; par exemple une agence (unité organisationnelle) ne gère que les contrats de son secteur.

L'expression d'un MOD local se représente pour chaque unité organisationnelle :

- par un schéma des entités, relations et propriétés utilisées,
- par un tableau précisant les éventuelles restrictions sur les occurrences disponibles.

Ce découpage du MOD global en MOD locaux permet d'apprécier le degré de partage ou de séparation des données d'un système d'information en fonction de l'organisation adoptée. On peut ainsi mettre en évidence :

- les données communes à l'ensemble du domaine,
- les données partagées entre certaines unités,
- les données privées à une unité.

Lorsqu'un partage existe entre plusieurs unités organisationnelles et en cas de répartition informatique envisagée, il peut être utile de préciser, si c'est possible, quelle unité organisationnelle ferait référence en cas de divergence entre le contenu des informations partagées.

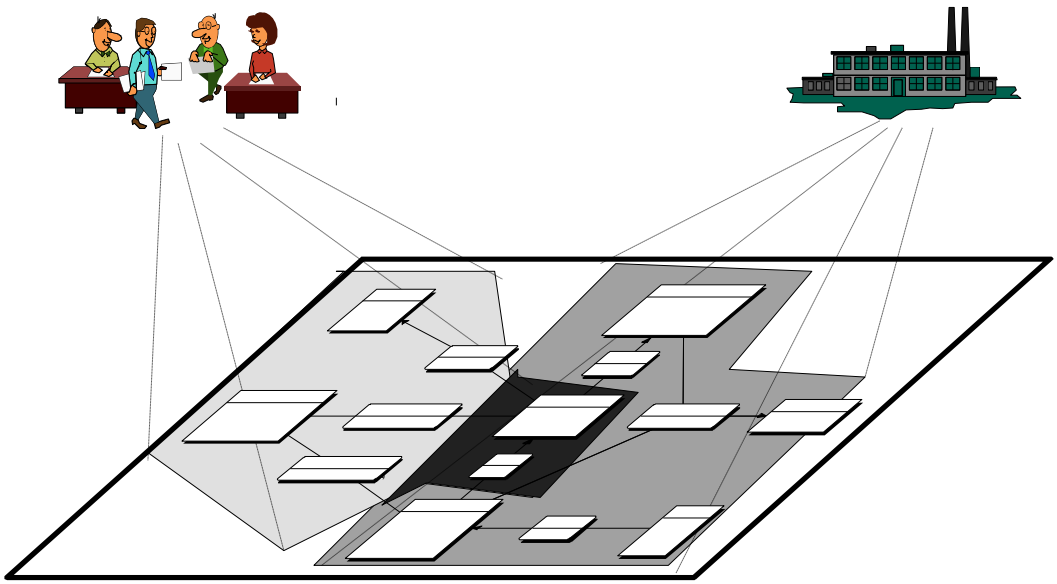


Figure 9.9 : Unités organisationnelles et MOD locaux

Accessibilité des données d'un MOD local

L'accessibilité des données d'un MOD local s'exprime par les actions élémentaires que peuvent effectuer sur ce sous-ensemble de données les traitements réalisés dans le site organisationnel.

Ces différents types d'accès, en lecture (L), en modification (M), en création (C) et en suppression (S) sont précisés sur le MOD local généralement sur un tableau récapitulant les restrictions de disponibilités, les partages et les actions autorisées.

La notion d'accessibilité d'un MOD local peut s'assimiler à un macro sous-schéma (voir « Sous-schéma organisationnel de données » dans la modélisation organisationnelle des traitements) au niveau d'une unité organisationnelle.

Sécurité des données

La sécurité des données définit des restrictions d'accès aux données mémorisées pour certaines catégories d'utilisateurs. Ces restrictions peuvent concerner un type d'action limité (L, M, C, S) soit aux entités, relations ou propriétés du MOD global ou local, soit à une sous-population des occurrences d'entités ou des relations La sécurité d'accès aux données comprend la limitation d'actions à certaines personnes (seul le responsable de la comptabilité peut modifier une écriture comptable) et intègre aussi les aspects de

confidentialité (accès à certaines informations du dossier de personnes dites « sensibles »).

La sécurité d'accès s'exprime, selon les cas, au niveau du MOD global ou des MOD locaux, et passe par la définition de catégories ou profils d'utilisateurs. Pour chaque profil, on précise les éventuelles restrictions d'accès envisagées. En pratique, on présente ces restrictions sous la forme de tableau faisant référence aux schémas MOD comme celui de la figure 9.10. Selon les cas, la sécurité d'accès aux données peut s'exprimer par une restriction par rapport à une autorisation générale, ou une autorisation par rapport à une restriction générale

Profil utilisateur : Employé

Entité - Relation Propriété	restriction ou autorisation
CLIENT niveau découvert	Lecture seule autorisée

Profil utilisateur : Chef de service

Entité - Relation Propriété	restriction ou autorisation
CLIENT niveau découvert	Modification autorisée montant <= 10 000 F.

Profil utilisateur : Directeur

Entité - Relation Propriété	restriction ou autorisation
CLIENT niveau découvert	Modification autorisée tout montant

Figure 9.10 : Expression des restrictions et autorisations d'accès

Bien que similaires et pouvant conduire à des modes d'implémentation informatiques proches, la notion d'accessibilité aux données des unités organisationnelles et la notion de sécurité d'accès ont cependant des finalités différentes. L'accessibilité vise apprécier la répartition et le partage de l'utilisation des données en vue d'une éventuelle répartition informatique. La sécurité concerne les conditions d'accès aux données par des profils d'utilisateurs, même en l'absence totale de préoccupation de répartition organisationnelle ou informatique.

11

Confrontation données / traitements

Rôle et nécessité

Bien que visant le même objectif global, assurer la cohérence et la qualité du futur système d'information, il convient au préalable de distinguer confrontation données / traitements et validation, et ceci d'autant plus que, par le passé, ces deux notions furent souvent confondues.

La confrontation cherche à vérifier la cohérence entre les modélisations des données et des traitements; c'est une technique spécifique aux concepteurs et propre à la méthode Merise

La validation cherche à vérifier que les solutions proposées sont conformes aux besoins; c'est un travail entre l'équipe de projet et la maîtrise d'ouvrage (décideur ou futurs utilisateurs) représentée par le groupe de validation (voir « Groupe de validation » dans Structures et intervenants).

Dans le déroulement du cycle d'abstraction, le concepteur a alternativement élaboré les modèles de données et de traitements. On a alors constaté la relative indépendance entre données et traitements évoquée dans la première partie**.**

Le modèle conceptuel de données a été élaboré sans approfondir les conditions d'utilisation des informations modélisées.

Les modèles conceptuel et organisationnel de traitements ont pu être élaborés sans insister sur les informations utilisées.

Or, chaque opération ou tâche exprimée dans le modèle conceptuel ou organisationnel de traitements interagit avec les données mémorisées. Si, dans le processus d'élaboration d'un modèle conceptuel de données, la méthode Merise préconise une structuration des informations en fonction de leur signification, il est indispensable de s'assurer régulièrement que leur utilisation est compatible avec leur signification modélisée.

Enfin, la seule structuration des informations perceptible par l'utilisateur est

celle exprimée dans les messages, structuration fortement conditionnée par les traitements associés à ces messages.

On doit donc confronter :

- Une perception globale, unitaire, abstraite, représentée par le modèle conceptuel de données, et maîtrisée essentiellement par les concepteurs.
- Une juxtaposition de perceptions partielles, redondantes, contingentes à des traitements, représentées par les messages et partagées entre tous les utilisateurs.

Cette confrontation consiste schématiquement à :

- Vérifier si les traitements disposent bien des données nécessaires.
- Contrôler si les données sont utilisées dans les traitements.

Dans les deux cas, on s'assurera que la signification de la structuration des données est cohérente avec leurs utilisations dans les traitements.

Différents modes de confrontation

Dans son principe, la confrontation données / traitements sera toujours une simulation du fonctionnement des futurs traitements utilisant les futures données.

En pratique, plusieurs techniques de confrontation sont utilisées qui dépendent du degré de détail significatif et de la couverture du domaine, liés à la démarche. Elles seront assez grossières en étude préalable pour s'affiner en étude détaillée.

En étude préalable :

- Relecture croisée MCD / MCT
- Grille de cohérence globale MCD-MOD / MOT

En conception générale (première phase de l'étude détaillée)

- Grille de cohérence globale MCD-MOD / MOT

En conception détaillée (deuxième phase de l'étude détaillée)

- Confrontation détaillée entre MCD-MOD et les modèles externes (expression formalisée du contenu des messages, règles de traitement et actions d'une tâche).
- Maquettage impliquant le MCD-MOD et les futures tâches à automatiser. Cette technique peut, selon l'outil de maquettage utilisé,

contribuer très efficacement à la confrontation

Relecture croisée MCD / MCT

En étude préalable, et particulièrement dans les premiers travaux de conception des modèles conceptuels futurs, on vérifiera qu'il y a une bonne concordance entre le sous-ensemble représentatif du domaine retenu pour le MCD et celui retenu pour le MCT.

En relisant le MCT, on vérifie si les objets et associations évoqués dans les opérations sont modélisés dans le MCD. Inversement, en relisant le MCD, on vérifie que les entités et relations représentées trouvent une utilité dans le MCT.

Bien que techniquement très sommaire, cette relecture croisée s'avère en pratique d'une bonne efficacité et permet fréquemment de recadrer les deux modélisations.

Grille de cohérence globale MCD-MOD / MOT

Ce raisonnement consiste à contrôler si les tâches du MOT disposent des données nécessaires à leur fonctionnement (du moins en termes d'entités et de relations), et que les entités et relations du MCD/MOD sont utilisées dans les tâches. On utilise la technique de la grille de cohérence globale. Nous présentons la technique manuelle dont les principes peuvent par ailleurs s'automatiser à travers des outils d'aide à la conception.

Sur un tableau, on indique (voir figure 11.3)

- en colonne, les différentes entités et relations du MCD-MOD, dans un ordre quelconque;
- en ligne, les différentes tâches du MOT en respectant autant que possible la chronologie de ces traitements.

Sur chaque case du tableau, on indique les actions effectuées par chaque tâche du MOT sur toute entité ou relation du MCD-MOD:

Selon les traitements réalisés, la tâche peut comporter aucune, une ou plusieurs actions élémentaires sur les données. Si la tâche est entièrement manuelle, elle ne peut accéder à aucune donnée mémorisée; les cellules correspondantes restent alors vides.

Pour illustrer cette grille de cohérence, prenons le cas assurance déjà abordé, dont les figures 11.1 et 11.2 rappellent le MCD-MOD et le MOT.

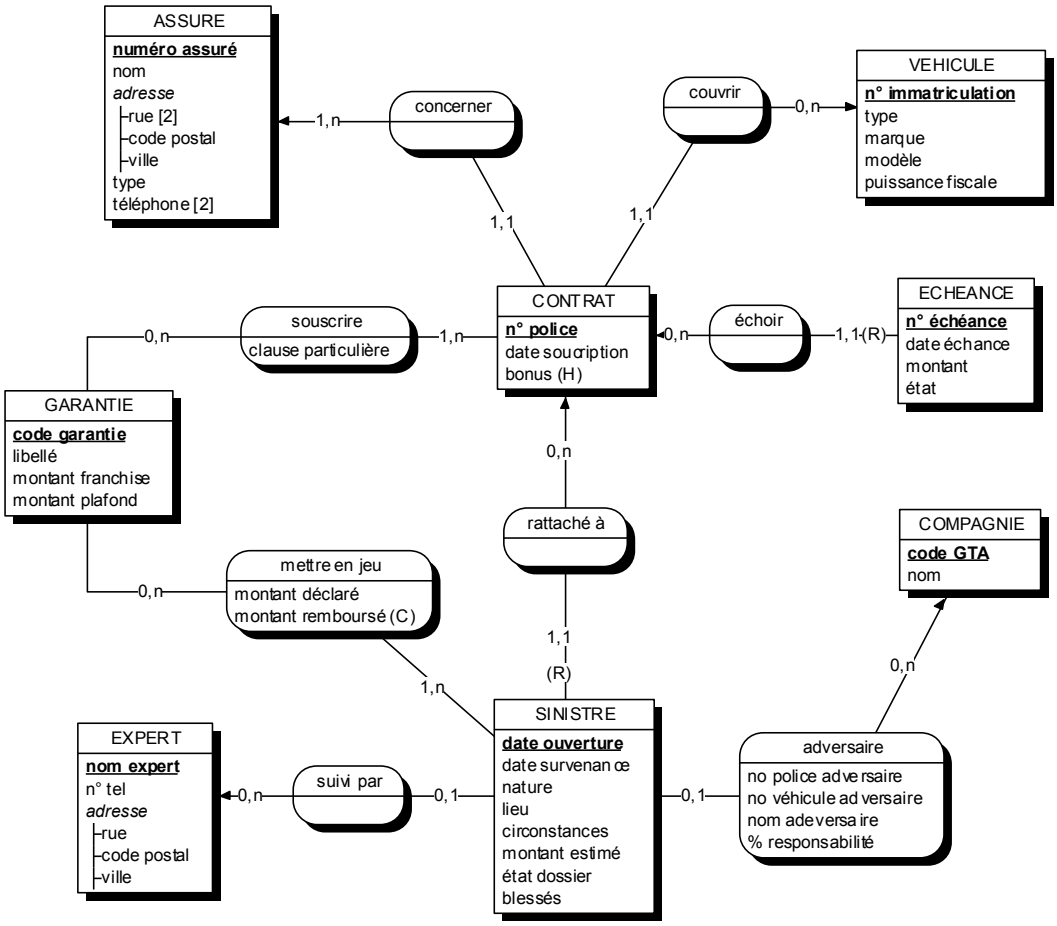


Figure 11.1 : MCD/MOD du cas assurance

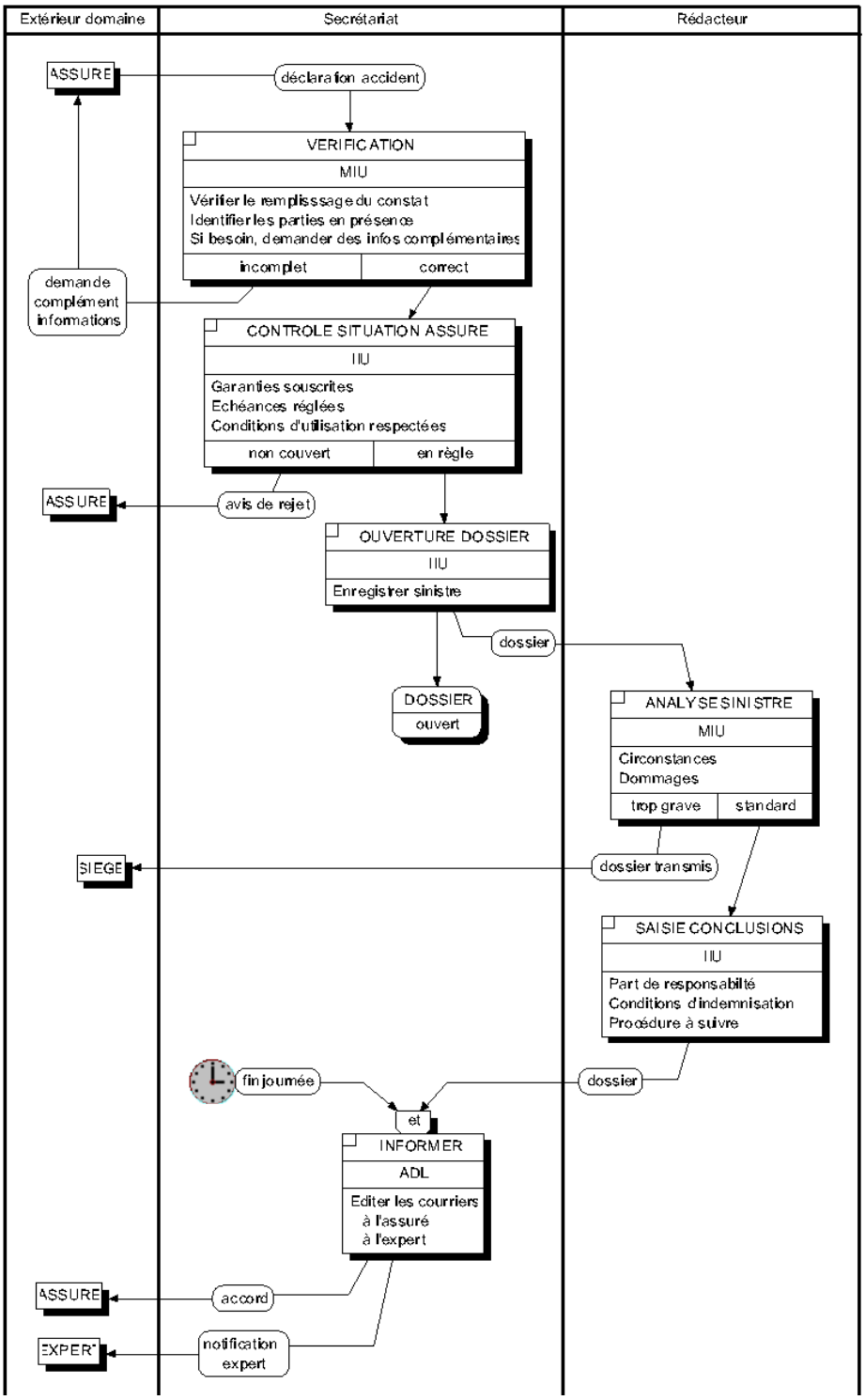


Figure 11.2 : MOT du cas assurance.

Pour chaque tâche de cette procédure, on précise la ou les actions sur les entités et relations du MCD (CREation, LECture, MODification, SUPpression), comme l'illustre la figure 11.3.

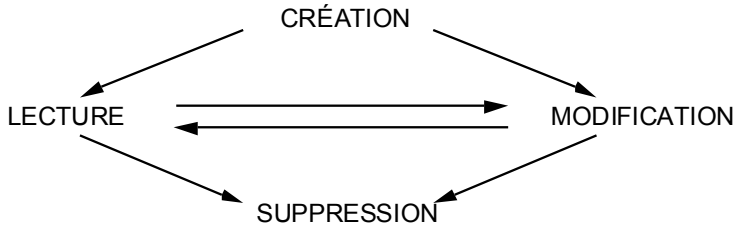
Entités et relations (liste partielle)

tâches	ASSURE	CONTRAT	GARANTIE	SINISTRE	souscrire	concerner	rattacher	mettre en jeu
Vérification								
Contrôle	LEC	LEC	LEC		LEC	LEC		
Ouverture				CRE			CRE	
Analyse								
Saisie				MOD				CRE
Informer	LEC	LEC	LEC	LEC			LEC	LEC

Figure 11.3 : Grille de cohérence MCD-MOD / tâches.

Lorsque le tableau est entièrement rempli, on procède alors à une *double relecture* :

- Pour chaque ligne, on s'assure que les actions élémentaires prévues sont bien cohérentes par rapport à la désignation de la tâche; au besoin, on procède aux ajustements nécessaires dans la description de la tâche.
- Pour chaque colonne, on s'assure que, pour chaque entité ou relation, on retrouve au niveau d'une occurrence la séquence générale suivante :



Le concepteur doit prêter attention à certaines situations dont quelques-unes peuvent révéler une grave anomalie.

- Entité ou relation sans aucune action

Cette situation peut provenir de plusieurs cas :

- Les entités ou relations ne sont pas utilisées par des tâches du sous-ensemble représentatif; il faut éventuellement réajuster la concordance du sous-ensemble représentatif (données ou traitement).
- Les entités ou relations avaient un intérêt pour des tâches qui sont manuelles; il faudra en tenir compte au niveau du MOD.
- Entité ou relation jamais créés

Cette situation peut provenir des cas suivants :

- La tâche chargée de la création n'appartient pas au sous-ensemble représentatif.
- L'action de création a été « oubliée ».
- Entité ou relation créés par plusieurs opérations ou tâches

Bien que cette situation soit théoriquement normale, si elle met en jeu plusieurs postes, elle peut alors poser des problèmes organisationnels (responsabilité de création).

- Entité ou relation avec propriété(s) non modifiée(s)

Aucune action de modification n'a été recensée pour cette entité ou cette relation, ses propriétés ne peuvent donc changer de valeurs, cela peut être volontaire dans le cas de données sensibles, sinon il s'agit probablement d'un oubli.

- Entité ou relation jamais supprimés

Le concepteur doit vérifier que la fréquence des créations de cette entité ou relation ne va pas conduire à une saturation de la mémoire. Cette situation est assez fréquente; on néglige trop souvent dans une étude le processus d'épuration de la mémoire courante (archivage, durée de vie des informations).

Cette technique simple permet de détecter rapidement un grand nombre d'incohérences entre données et traitements. En revanche, elle n'est pas adaptée à une présentation aux gestionnaires et utilisateurs du domaine. Il est recommandé de procéder au contrôle de cohérence global avant de présenter les modèles aux gestionnaires et utilisateurs du domaine.

Bien que cette technique puisse sembler lourde à appliquer manuellement, elle reste cependant réalisable sur des projets moyens. Le recours à des outils permet évidemment d'automatiser ces diagnostics.

Confrontation algorithmique détaillée

En conception détaillée, les techniques précédentes sont insuffisantes et l'on doit descendre au niveau des différentes propriétés. A ce stade, le MCD et le MOD sont complètement décrits (ou en voie d'achèvement), et les concepteurs peuvent spécifier plus précisément le contenu des tâches.

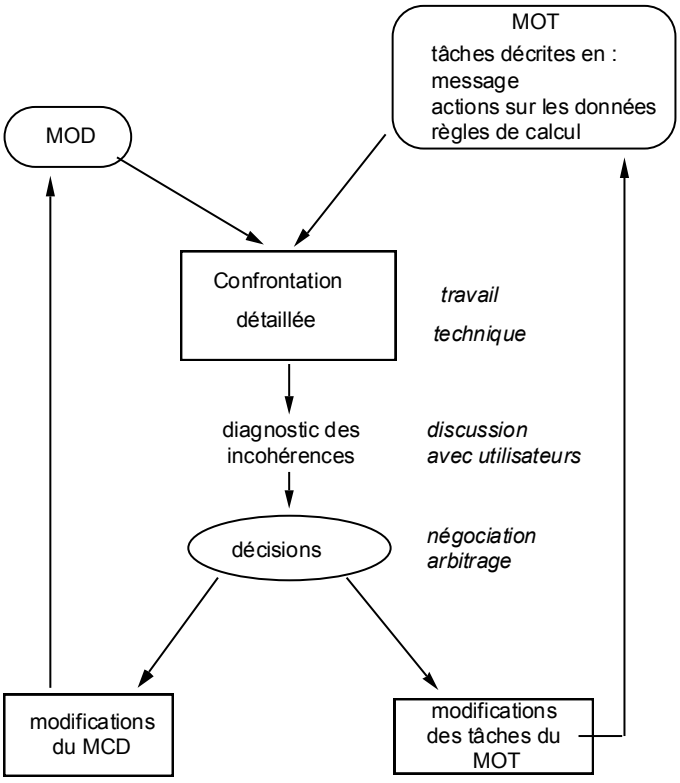
Cette analyse détaillée des tâches comprend :

- La description des messages en entrée et en sortie en précisant les informations contenues et leur structure; ainsi que leurs conditions de synchronisation ou d'émission ;

- Les règles de traitement appliquées sur ces informations;
- Les actions de consultation et de mise à jour des données mémorisées;

Cette description exprime la vision des informations contingente aux traitements effectués par la tâche, seule perception connue des utilisateurs. Cette perception modélisée est appelée *modèle externe*

Le processus de confrontation détaillée consiste à vérifier si, sous des apparences parfois très différentes, modèle organisationnel de données et modèle externe sont compatibles en signification et structure. Les principes de cette confrontation détaillée sont présentés sur la figure 11.4



.

Figure 11.4 : Principes de la confrontation détaillée.

Principes de la confrontation détaillée

Cette technique consiste à confronter le modèle organisationnel de données

avec les modèles externes associés à chaque tâche du modèle organisationnel de traitements (voir figure 11.5).

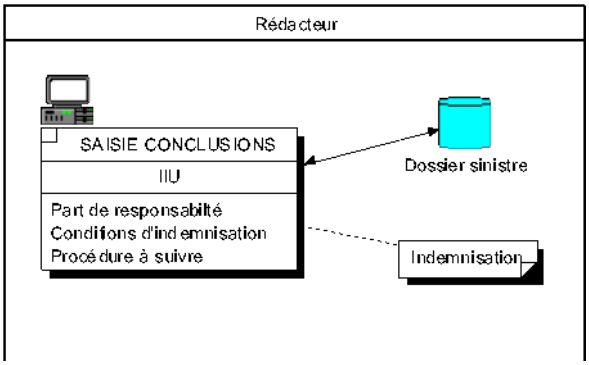


Figure 1156 : Tâche Saisie conclusions sinistre à confronter avec le MCD-MOD assurance (figure 11.1).

Cette tâche, à laquelle est associé un modèle externe, est-elle cohérente par rapport à un modèle organisationnel de données défini par ailleurs ?

Une tâche peut comporter plusieurs modèles externes, en particulier si elle émet plusieurs résultats conditionnels de nature et de contenu différents. En général, un modèle externe se compose :

d'un message dont le contenu est modélisé par une vue externe;

des règles de traitement;

des actions sur un sous-schéma de données.

Modélisation du contenu d'un message : vue externe

Chaque message est constitué d'un ensemble structuré d'informations représenté par une *vue externe de données*. La formalisation de cette structure peut s'effectuer soit en utilisant le formalisme entité – relation, soit en terme de groupes de données structurés ; dans les deux cas, seule la règle de vérification (non-répétitivité) est utilisée. Rappelons que dans l'expression formelle du contenu d'un message, il ne s'agit pas de reconstituer un « pseudo-modèle » conceptuel de données, mais d'exprimer strictement ce que l'on voit. On constate généralement que les types de structures présents sont assez simples: unitaire, hiérarchique, arborescent.

Dans le cas où des messages en entrée d'une tâche seraient soumis à une synchronisation, il faut modéliser un message fictif résultant de la composition des messages, en fonction de l'expression logique de synchronisation. Par exemple :

- 2 messages A et B synchronisés par une expression OU donneront naissance à deux vues externes.

- 2 messages A et B synchronisés par une expression ET ne donneront naissance qu'à un message fictif résultant de la fusion des messages, donc à une vue externe.
- 3 messages A, B et C synchronisés par une expression (A ET B) OU C produiront deux vues externes; l'une concernant (A ET B), l'autre concernant C.

La figure 11.6 illustre le contenu du message associé à la tâche Saisie conclusions, c'est à dire les conclusions issues de la tâche manuelle précédente d'Analyse du sinistre (voir figure 11.2)

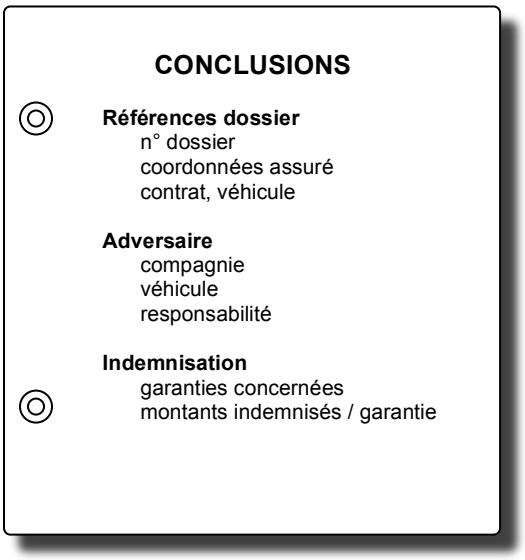


Figure 11.6 : Contenu du message Conclusions sur le sinistre.

En mettant en évidence les répétitions (règle de vérification), le message peut être modélisé comme sur les figures 11.7a et 11.7b.

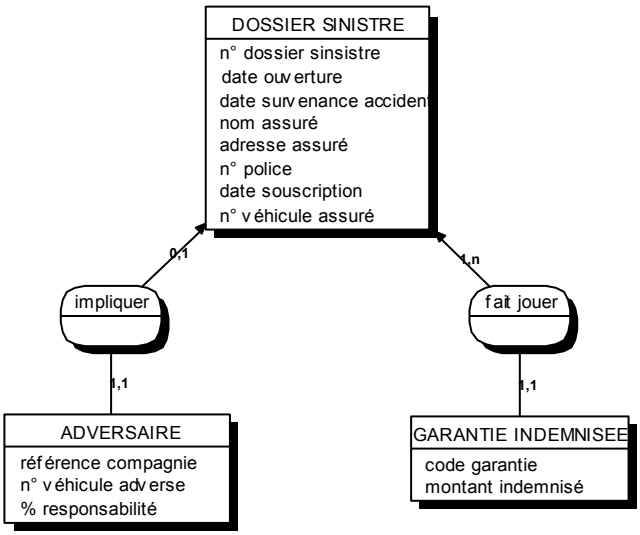


Figure 11.7a : Modélisation de la vue externe Conclusions sous la forme d'entités et relations externes

DOSSIER SINISTRE (1,1)
- n° dossier sinistre
- date ouverture
- date survenance accident
- nom assuré
- adresse assuré
- n° police
- date souscription
- n° véhicule assuré

ADVERSAIRE (0,1)
- Référence compagnie
- N° véhicule adverse
- % responsabilité

GARANTIE INDEMNISEE (1,n)
- code garantie
- montant indemnisé

Figure 11.7b : Modélisation de la vue externe Conclusions sous la forme de groupes de données

Expression des actions sur les données mémorisées

Les données mémorisées sont représentées par le modèle conceptuel ou organisationnel de données. L'accès à ces données ne peut se faire qu'à travers une tâche, données représentées par le sous-schéma conceptuel ou

organisationnel de données.

Comme on l'a vu précédemment (dans « Sous-schéma de données » chapitre Modélisation organisationnelle des traitements), on distingue quatre natures d'actions qui s'appliquent aux occurrences d'entité ou de relation. On obtient ainsi huit actions types :

- Créer entité; créer une nouvelle occurrence d'une entité avec au minimum la valeur de son identifiant et souvent des valeurs de quelques propriétés.
- Modifier entité; modifier des valeurs de propriétés d'une occurrence d'entité déjà existante, à l'exception de son identifiant. Cette action est également utilisée dans le cas où l'on affecte une première valeur à une propriété d'une occurrence déjà existante. Cette action intègre le fait que l'accès à l'occurrence à modifier doit être réalisé avant la modification.
- Supprimer entité; supprimer de la mémoire une occurrence d'une entité existante. Par définition, l'ensemble des valeurs des propriétés associées disparaît de la mémoire. La suppression d'une occurrence d'une entité implique que les occurrences des relations liées à cette entité ont été supprimées auparavant. Cette action de suppression intègre le fait que l'accès à l'occurrence à supprimer doit être réalisé avant suppression.
- Consulter entité; accéder à une occurrence d'une entité soit pour en consulter les valeurs des propriétés, soit pour y faire simplement référence, en particulier pour agir sur les relations.
- Créer relation; créer une nouvelle occurrence d'une relation entre des occurrences d'entités existantes. Éventuellement des valeurs de propriétés peuvent être données. La création d'une nouvelle occurrence de relation exige qu'au préalable les occurrences d'entités composant sa collection aient été consultées ou créées.
- Modifier relation; modifier des valeurs de propriétés d'une occurrence d'une relation déjà existante. Cette action est également utilisée dans le cas où l'on affecte une première valeur à une propriété d'une occurrence déjà existante. Cette action intègre le fait que l'accès à l'occurrence à modifier doit être préalablement réalisé avant la modification. Cette action ne s'applique pas dans le cas où l'on modifie la collection de l'occurrence de la relation. Il faut alors supprimer l'occurrence de l'ancienne relation pour créer l'occurrence de la nouvelle.
- Supprimer relation; supprimer de la mémoire une occurrence d'une relation existante, ainsi que l'ensemble des valeurs de ses éventuelles propriétés.
- Consulter relation; accéder à une occurrence de relation soit pour en

consulter le contenu des propriétés, soit pour parcourir la relation afin d'atteindre une occurrence d'entité.

L'ensemble de ces actions types ne fait aucune hypothèse sur la technique utilisée ultérieurement pour réaliser ces accès. Ainsi, on supposera que l'accès à une occurrence d'une entité peut être réalisé à partir d'une valeur de son identifiant ou par le parcours de relations binaires fonctionnelles, et que l'on peut sélectionner en fonction d'une expression logique un sous-ensemble d'occurrences d'entité ou relation à partir de toute propriété de ces concepts.

Algorithme de confrontation détaillée

L'analyse complète de ces différents algorithmes a été faite dans le cadre des recherches [Tardieu, Nanci, Heckenroth 76] [Tardieu, Nanci, Pascot 79]. En outre, nous pensons que la technique de maquettage peut, sous certaines conditions, se substituer à cette procédure dont l'aspect fastidieux est indéniable.

Nous présentons ici les principes généraux de l'algorithme de confrontation détaillée en écartant les cas particuliers dont l'intérêt théorique est certain, mais dont la fréquence reste exceptionnelle. Pour dérouler l'algorithme, on dispose au préalable :

- de la vue externe de données.
- du modèle conceptuel ou organisationnel de données;
- de la tâche pour laquelle on connaît la fonction, même si les règles de calcul et les actions ne sont pas toujours formellement exprimées;

Pour valider le modèle externe par rapport au modèle de données, on effectue les opérations décrites ci-après.

Équivalence propriété externe / propriété conceptuelle

A chaque propriété externe présente dans la vue externe, on recherche une correspondance dans le modèle conceptuel. Cette correspondance peut être :

- directe; il y a équivalence stricte entre les deux informations;
- via une relation; la propriété externe est équivalente à une propriété conceptuelle vue à travers une relation (généralement une relation binaire fonctionnelle);
- via une règle de traitement; on remplace alors la propriété externe par les propriétés conceptuelles évoquées dans la règle de traitement. Dans certains cas, surtout dans les restitutions, la règle génère une véritable structure complémentaire au modèle externe initial; le concepteur a alors intérêt à expliciter cette structure de calcul (généralement des

sommations) avant de poursuivre son algorithme de confrontation détaillée.

Certaines règles de traitement évoquent des propriétés conceptuelles sans se traduire par des propriétés externes; il s'agit fréquemment de règles d'affectation de valeurs par défaut ou automatiques.

Pour chacune des propriétés conceptuelles ainsi évoquées, on relève l'entité ou la relation conceptuelles d'accueil. On note également s'il s'agit d'un identifiant et si la propriété externe était optionnelle dans le message.

Dans certains cas, des propriétés externes différentes correspondent aux mêmes propriétés conceptuelles mais pour des occurrences distinctes. On prendra alors en compte ces deux occurrences du même concept en prenant soin de les distinguer par des indices. Par la suite, on les traitera comme des concepts différents.

Si l'on échoue dans cette première étape, la tâche représentée par son modèle externe est incompatible avec le modèle conceptuel. Il s'agit fréquemment de l'oubli d'une propriété dans le modèle conceptuel de données. C'est à cette étape que le modèle conceptuel de données s'enrichit progressivement en propriétés.

A l'issue de cette étape, pour chaque entité externe ou groupe de données composant le message, on dispose d'un ensemble d'entités et de relations conceptuelles évoquées par l'équivalence des propriétés. Ces entités et relations ne sont pas nécessairement interreliées. L'étape suivante va chercher à reconstituer des sous-ensembles connexes du modèle conceptuel correspondant à chaque entité externe.

Correspondance entité externe / sous-ensemble conceptuel

Cette opération s'applique à chaque entité externe ou groupe de données de la vue de données. Si parmi les entités et relations conceptuelles évoquées précédemment, on enregistre certains éléments isolés, le concepteur cherche à les relier à des éléments cités. On utilise des relations du modèle conceptuel non encore directement citées (simple ou par composition). Le concepteur devra s'assurer que la signification des relations utilisées correspond à celle du contexte de la tâche. Lors de cette opération, le concepteur peut mettre en évidence l'absence de certaines relations conceptuelles ou une divergence de signification.

Une fois le sous-ensemble reconstitué, il faut s'assurer que cette structure conceptuelle est équivalente à la structure d'entité externe. Cette vérification introduit la notion de *pivot*. Le pivot d'une entité externe est une entité ou relation conceptuelle du sous-ensemble correspondant, tel que, à partir de l'occurrence de ce pivot, l'on peut déterminer une seule occurrence de chacun

des autres éléments du sous-ensemble. Dans ce cas, la règle d'unicité des propriétés de l'entité externe est complètement assurée par la structure conceptuelle correspondante. On peut dire également que les occurrences de l'entité externe sont de même distinguabilité que les occurrences du pivot.

La détermination du pivot s'appuie sur trois règles du formalisme entité-relation :

- une relation porteuse d'une dépendance fonctionnelle (généralement une relation binaire à cardinalité maxi = 1) permet, à partir d'une occurrence d'une entité, de faire référence à une seule occurrence d'une autre entité via cette relation;
- une occurrence de relation fait référence à une occurrence de chacune des entités qui la compose;
- à une occurrence de chacune des entités composant une relation, il n'y a au plus qu'une occurrence de cette relation.

La figure 11.8 illustre la correspondance entre les entités externes et le sous-ensemble conceptuel.

Les échecs rencontrés lors de cette étape concernent fréquemment l'unicité du pivot. Cela est souvent dû à une divergence de cardinalité dans les relations. Dans ce cas, sauf situations très particulières, il y a incohérence entre la structure de la vue externe et le modèle conceptuel de données. Le concepteur doit remédier à cette situation soit en modifiant la tâche, soit en remettant en cause le modèle conceptuel de données. Dans tous les cas, il s'agit d'un motif grave d'incohérence.

A l'issue de cette opération, le concepteur a réussi à reconstituer, pour chaque entité externe ou groupe de données, le sous-ensemble conceptuel correspondant. Il faut maintenant effectuer une opération semblable pour les éventuelles relations externes.

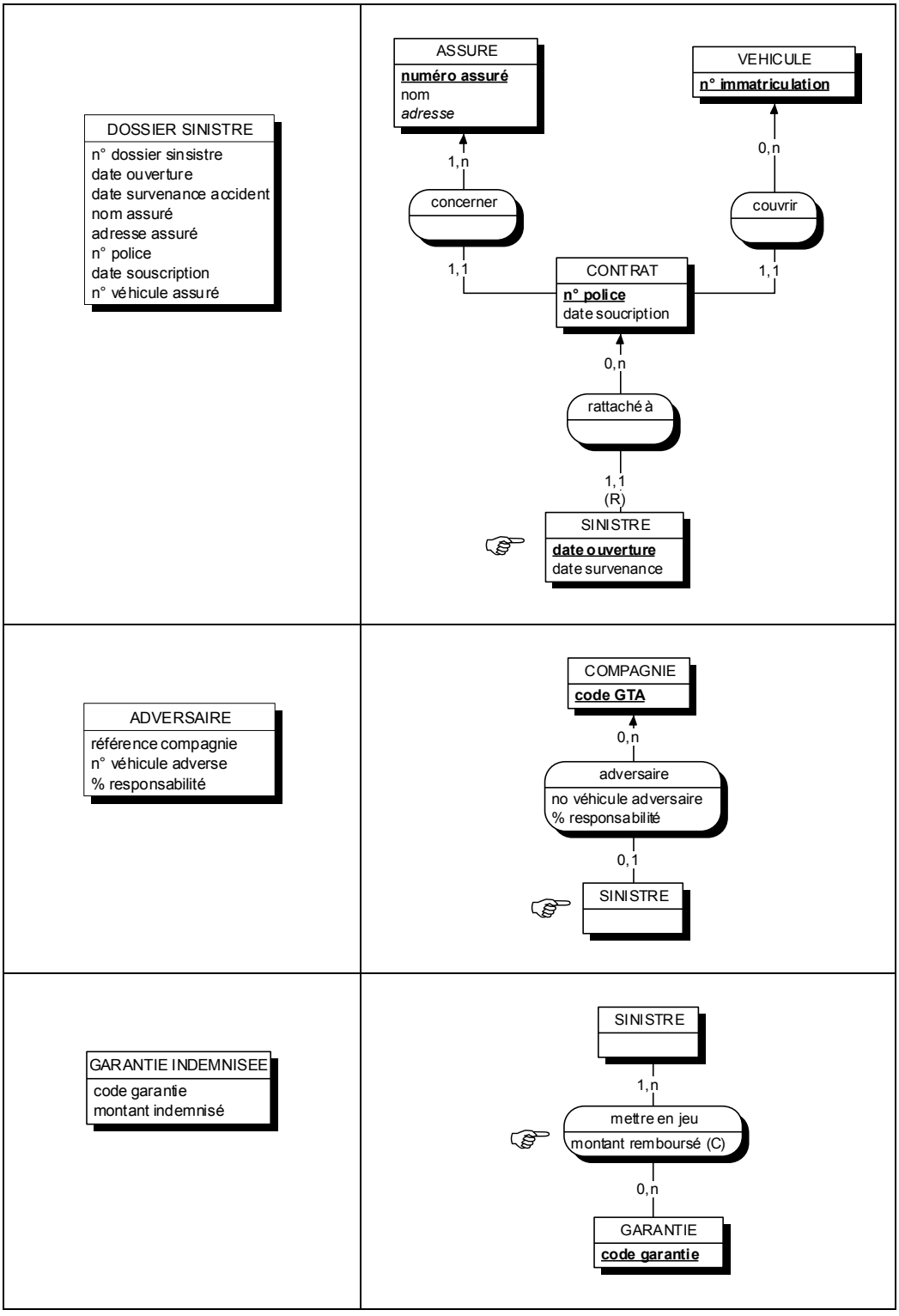


Figure 11.8 : Correspondances entités externes et pivots (☞)

Correspondance relation externe / relation conceptuelle

Les relations externes sont généralement binaires fonctionnelles dues à la structure hiérarchique des vues externes. Il s'agit de retrouver quel élément du modèle conceptuel correspond à la relation externe. Il peut s'agir :

- d'une relation conceptuelle dont la collection doit être constituée des pivots des entités externes;
- d'une patte d'une relation conceptuelle si l'un des pivots est constitué par une relation conceptuelle;
- du partage d'entités conceptuelles communes aux deux structures correspondant aux entités externes reliées.

Dans tous les cas, il faut s'assurer que les cardinalités conceptuelles sont compatibles avec la cardinalité externe exprimée. Cette compatibilité concerne l'unicité opposée à la multiplicité.

Dans l'exemple de la figure 11.8, la relation externe Impliquer est équivalente au partage de Sinistre et la relation externe Fait jouer est équivalente à la patte Sinistre de relation conceptuelle Mettre en jeu.

A l'issue de cette opération, on a reconstitué d'une manière statique la correspondance entre le message et le modèle conceptuel de données. On retrouve là la notion de sous-schéma conceptuel de données associé à une tâche.

Cette séquence de confrontation est suffisante pour tous les modèles externes dits en consultation qui concernent généralement des restitutions d'informations. Pour les modèles externes en mise à jour, c'est-à-dire qui modifient le contenu de la mémoire, il faut maintenant s'assurer que les actions envisagées sont possibles.

Prise en compte des actions

A chaque entité ou relation conceptuelles évoquées, le concepteur va préciser les actions (créer, modifier, supprimer, consulter) correspondant aux intentions de la tâche. Ce choix peut être la source de discordances qui peuvent conduire jusqu'à la remise en cause du rôle de la tâche, voire de sa place dans le modèle organisationnel. Suivant les types d'actions retenus, plusieurs cas se présentent.

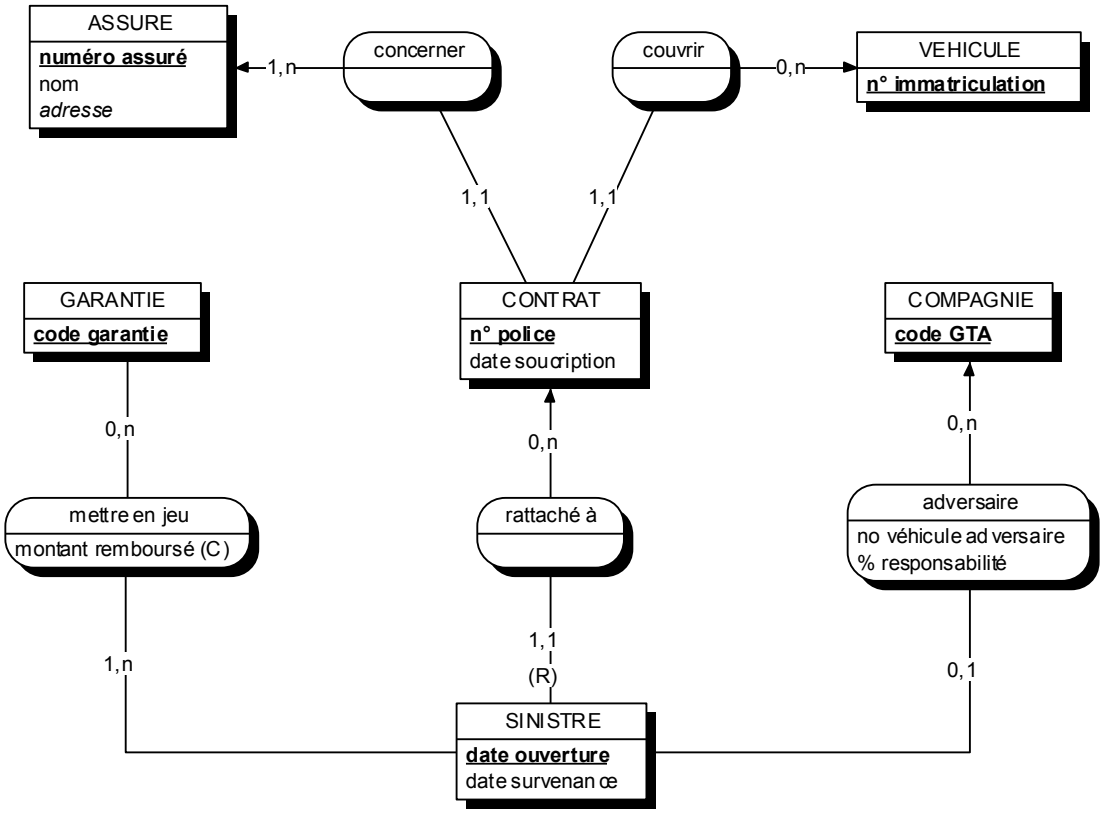
- Création, modification, suppression d'une occurrence d'entité; il faut impérativement que l'identifiant de l'entité soit présent dans les propriétés invoquées par le message.
- Consultation d'une occurrence d'une entité; le concepteur doit systématiquement se poser les questions suivantes :
- S'agit-il d'une redondance voulue dans le message ?

- L'entité auquel il est fait référence est-elle utilisée pour créer une occurrence de relation; surtout si l'identifiant de cette entité est explicitement citée dans le message ?

Cette dernière question permet de mettre parfois en évidence l'oubli d'une relation binaire fonctionnelle entre l'une des entités que l'on crée et l'entité que l'on consulte.

- Création, modification, suppression d'une occurrence de relation : il faut impérativement que les identifiants des entités composant la collection de cette relation soient présents dans les propriétés invoquées dans le message.
- Consultation ou parcours d'une occurrence de relation : s'il s'agit d'une relation binaire fonctionnelle, il suffit de connaître l'occurrence de l'entité origine (côté cardinalité maxi = 1); dans les autres cas, il faut connaître l'ensemble des identifiants des entités de sa collection.

Prenons l'exemple du sous-schéma conceptuel des données de la figure 11.9.



	1
ASSURE	Lec
concerner	Lec
CONTRAT	Lec
couvrir	Lec
VEHICULE	Lec
rattaché à	Lec
SINISTRE	Lec

	1
adversaire	Cré
COMPAGNIE	Lec
VEHICULE	Cré

	N
mettre en jeu	Cré
GARANTIE	Lec

Figure 11.9 : Prise en compte des actions.

On doit normalement retrouver les actions déjà envisagées dans la grille de cohérence

A l'issue de cette opération, le concepteur a pratiquement vérifié que le traitement envisagé par la tâche est possible. Il reste toutefois à vérifier si certaines notions exprimées dans le modèle conceptuel de données sont intégralement prises en compte dans la tâche, du moins pour ce qui peut la concerner.

Actions induites

Pour chaque occurrence d'entité créée, le concepteur doit examiner toutes les relations conceptuelles auxquelles participe cette entité. Si la cardinalité minimum est 1, alors une occurrence de la relation concernée doit être également créée dans cette même tâche. Cela peut parfois à une révision de la vue externe.

Pour chaque occurrence d'entité supprimée, le concepteur doit vérifier soit qu'il n'y a plus d'occurrence de relation reliée à cette entité, soit que les suppressions de ces occurrences interviendront dans cette tâche. Là encore, cela peut conduire à une révision de la vue externe, voire de l'objet de la tâche.

Rappelons également que, dans le cas de contraintes logiques interrelations, le concepteur doit vérifier que cette contrainte est respectée si l'une des relations concernées est mise à jour dans la tâche.

Redondance dans le message

Pour toutes les entités et relations conceptuelles consultées, dans un modèle externe en mise à jour, le concepteur doit s'interroger sur la présence, dans le message, de propriétés leur appartenant, autres que les identifiants. Théoriquement, ces informations n'apportent aucun élément nouveau pour la mémoire du système. On peut donc les considérer comme superflues, et l'on peut même proposer de fournir la valeur connue à l'utilisateur (pour des motifs de contrôle essentiellement). Techniquement, on pourra considérer ces informations comme des zones de sortie exclusives dans le futur écran associé

à la tâche.

Cependant, des différends peuvent surgir sur la nécessité de ces informations dans le message, révélant parfois une distorsion sur le rôle de la tâche et de ses actions associées.

Propriétés externes absentes

A chaque entité ou relation conceptuelles créées, le concepteur doit consulter la liste des propriétés conceptuelles décrivant cette entité ou cette relation. Il constate alors que certaines de ces propriétés ne sont pas évoquées directement ou indirectement dans le modèle externe. Il doit s'interroger sur cette absence, et s'assurer qu'elle est justifiée. Ce dernier contrôle ne peut pas conduire à une incohérence, tout au plus à l'enrichissement du message associé à la tâche.

L'ensemble de ces contrôles effectués sans échec permet de garantir la compatibilité entre le modèle conceptuel ou organisationnel de données et les vues externes associées aux tâches du modèle organisationnel de traitements. Il faut noter que cette confrontation détaillée permet d'obtenir :

- un enrichissement du modèle conceptuel ou organisationnel de données, en particulier de ses propriétés;
- une formalisation précise du contenu des messages associés aux tâches;
- une expression formalisée des règles de traitement;
- une expression des actions conforme à leur formulation conceptuelle.

Dans la pratique, cette confrontation s'effectue simultanément à la description détaillée de chacune des tâches dont elle permet d'obtenir l'essentiel de leur spécification. Rappelons enfin que :

- La confrontation algorithmique n'intervient qu'en conception détaillée.
- Le processus de confrontation doit se faire à modèle conceptuel de données (ou organisationnel) « constant »; à la suite d'une modification, on doit théoriquement revérifier tous les modèles externes précédemment validés.
- La confrontation détaillée peut remettre en cause indifféremment le modèle conceptuel ou organisationnel de données, les tâches et même le modèle organisationnel de traitements.

La confrontation détaillée est un préalable à la validation de la description détaillée des tâches..

Le maquettage, outil de confrontation

Le maquettage est d'abord une technique dont le rôle s'est développé pour présenter aux utilisateurs les interfaces détaillées du futur système et obtenir ainsi une meilleure validation. La maquette concrétise pour le gestionnaire ou l'utilisateur les spécifications externes de l'étude détaillée. Il peut donc vérifier directement si les solutions proposées sont conformes à ses besoins. Mais cela dépend du niveau de maquettage utilisé ; il ne faut pas confondre maquettage et prototypage.

La *maquette* est une représentation de la future application (ou seulement d'une partie) qui en possède la même apparence extérieure (ou proche). Elle facilite grandement la validation par les utilisateurs, des spécifications externes détaillées de l'interface (présentation et logique de dialogue), surtout dans son aspect ergonomique.

Le maquettage peut se présenter sous des degrés de sophistication variables :

- de simples descriptions graphique d'écran avec une interaction minimale (maquette statique),
- des écrans interactifs comportant l'essentiel de la logique du dialogue ainsi que les enchaînements (maquette dynamique).

Le maquettage joue pleinement son rôle dans le processus de confrontation des données et des traitements lorsque sa spécification permet d'associer :

- les informations utilisées dans l'interface et les données modélisées, l'ensemble des données de l'interface correspond ainsi à une vue externe
- les règles de traitements mise en jeu et exprimées sur les données modélisées .
- les actions exercées par la maquette sur les données mémorisées.

Un tel maquettage nécessite le recours à des outils adaptés. Même dans les cas les plus sophistiqués, la maquette ne peut être une application opérationnelle (ou une partie) utilisable même provisoirement par l'utilisateur. Elle doit rester un outil du concepteur pour représenter, illustrer et permettre de valider les spécifications de la future application.

Le *prototype* est une application, une partie d'application ou fonction particulière développée à titre expérimental. Il est destiné à vérifier la faisabilité d'une solution (ou d'un élément) dans des conditions d'utilisation les plus proches de la réalité (ou extrêmes). Le prototype peut porter sur l'évaluation d'un aspect fonctionnel, organisationnel ou technique. Le prototype, dans son champ d'expérience, comporte l'ensemble des fonctionnalités opérationnelles du futur système.

Conclusion

Dans le déroulement du cycle d'abstraction, nous avons jusqu'ici parcouru:

- la modélisation conceptuelle des traitements;
- la modélisation conceptuelle des données;
- la modélisation organisationnelle des données;
- la modélisation organisationnelle des traitements;
- la modélisation du cycle de vie des objets et des objets métiers,
- la confrontation données / traitements.

Ces différents raisonnements permettent la conception du système d'information organisationnel (SIO) et constituent l'aspect métier. Nous avons pu constater que les notions utilisées n'impliquaient aucune connaissance informatique et que les problèmes abordés relevaient avant tout de la gestion du domaine : fonctionnement général, signification des informations, organisation des ressources.

Il importe donc qu'au sein de l'équipe de conception le rôle des utilisateurs gestionnaires soit prépondérant. Nous reviendrons sur ces rôles ultérieurement.

Le déroulement du cycle d'abstraction se poursuit par :

- la modélisation logique des données;
- la modélisation logique des traitements;
- la modélisation physique des données et des traitements.

Ces raisonnements permettent la conception du système d'information informatisé (SII) et constituent l'aspect informatique. Nous introduirons donc progressivement des notions techniques nécessaires à la résolution des problèmes abordés.

Troisième Partie

Les raisonnements de la méthode Merise : conception du Systèmes d'Information Informatisé (SII)

Partie 3

Les raisonnements de la méthode Merise : conception du Systèmes d'Information Informatisé (SII)

Préambule

La deuxième partie traitait de la conception du Système d'Information Organisationnel. Cette troisième partie est consacrée à l'étude du Système d'Information Informatisé (SII), plus précisément à l'articulation des modélisations et formalismes associés.

Cette partie précisera comment élaborer et exprimer les différents modèles, comment passer d'un niveau d'abstraction au suivant et transformer les différents modèles et enfin, comment aborder toute optimisation :

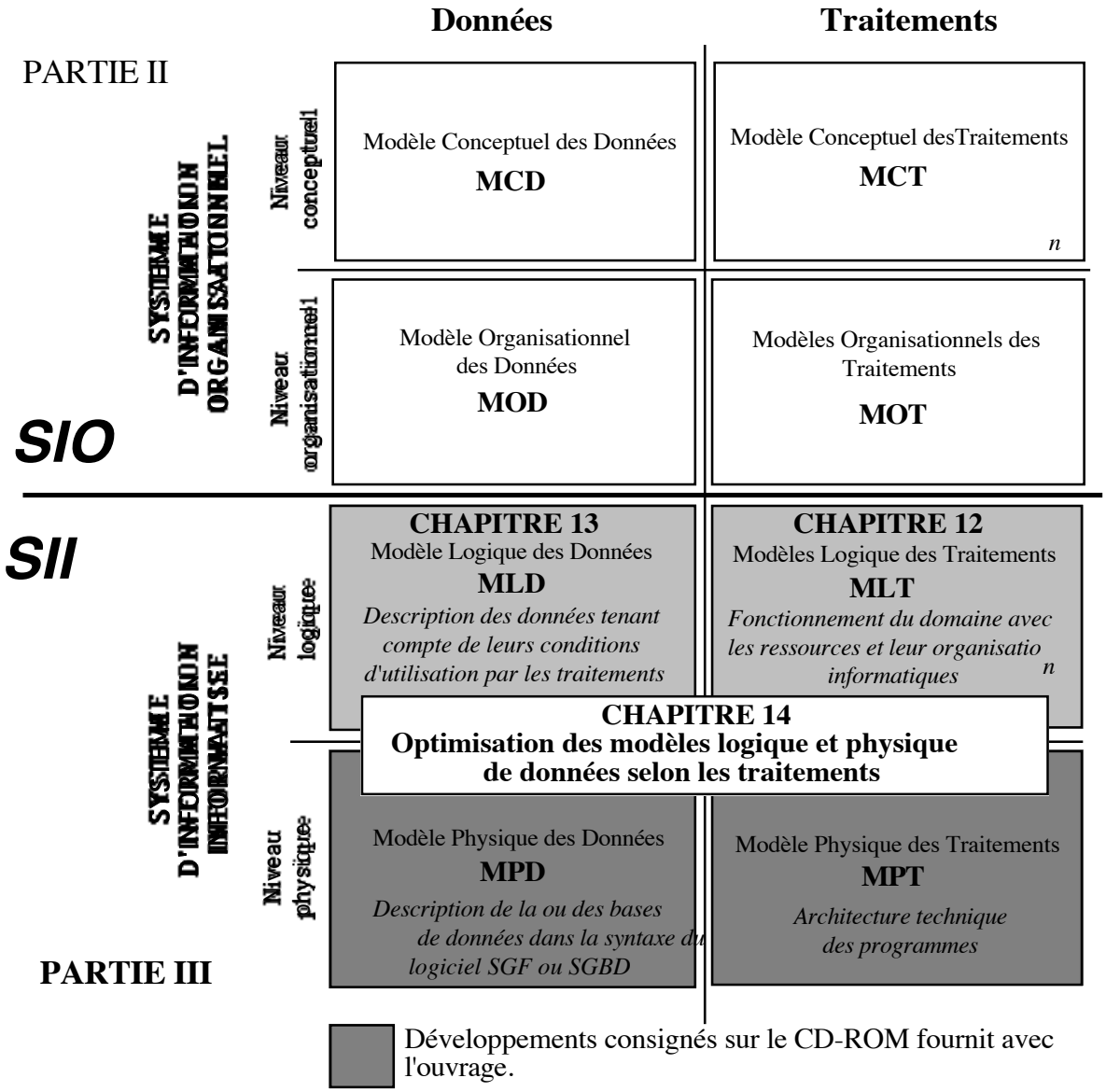
- Chapitre 12 : modélisation logique des traitements.
- Chapitre 13 : modélisation logique des données.
- Chapitre 14 : optimisation des modèles logiques et physiques de données.

Comme pour la deuxième partie, la présentation est effectuée ici sans référence aux autres cycles.

Nous rappelons, encore une fois, qu'une telle démarche reste théorique, et n'a de justification que pédagogique.

Dans la pratique, comme nous l'avons vu à la fin de la partie précédente, le parcours du cycle d'abstraction devra toujours se situer par rapport à une étape du cycle de vie, en particulier l'étude préalable, étude détaillée ou étude technique.

C'est ce que nous traiterons en détail dans la quatrième partie de l'ouvrage.



Les modèles du systèmes d'information informatisé.

12

Modélisation Logique des Traitements

Problématique de la modélisation logique des traitements (MLT)

La conception du système d'information organisationnel a conduit à l'élaboration des modèles conceptuels et organisationnels de traitements, MCT et MOT. Ces modèles ont permis de décrire le fonctionnement du SIO, en réponse aux stimuli en provenance de l'environnement du domaine d'activité étudié.

Le MCT a permis de décrire les fonctions majeures du domaine, sans référence aux ressources nécessaires pour assurer le fonctionnement ; on s'est concentré sur le quoi.

Le niveau d'abstraction suivant, appelé aux débuts de Merise « organisationnel - logique », se préoccupe du comment, c'est-à-dire des ressources (moyens techniques ou humains, espace, temps, données) et de leur mise en œuvre permettant d'assurer l'exécution des activités définies au niveau conceptuel.

Dix années de pratique ont révélé la nécessité de considérer deux perceptions et préoccupations différentes du comment, selon que l'on se situe du point de vue du gestionnaire (SIO) ou de l'informaticien (SII). Ainsi a progressivement émergé la distinction entre modèle organisationnel de traitements et modèle logique de traitements.

Le MOT se préoccupe d'une vision externe des moyens que l'entreprise va mettre en œuvre pour informatiser son système d'information. On s'intéressera à la répartition et à l'organisation des tâches entre l'homme et l'informatique, à la disponibilité des données. En résumé, le gestionnaire se pose la question : Comment vais-je informatiser et organiser les activités de mon domaine ?

Le MLT se préoccupe d'une vision interne des moyens que l'informaticien va utiliser pour construire le logiciel correspondant aux activités informatisées définies dans le MOT. On parlera d'enchaînement de transactions, de découpage en modules, de répartition des données et traitements informatisés.

L'informaticien se pose la question : Comment vais-je concevoir mon logiciel par rapport aux fonctions demandées ?

Le passage du système d'information organisationnel (SIO) au système d'information informatisé (SII), c'est le passage de solutions d'organisation à des solutions informatiques. Ces solutions informatiques spécifieront de façon fine et opératoire le SII.

Les modèles logiques de traitements (MLT) ont pour objectif de décrire le fonctionnement du SII en réponse aux stimuli des événements associés aux tâches informatisées précisées dans les MOT du SIO, comme l'illustre la figure 11.2.

En résumé, la problématique de la modélisation logique des traitements c'est comment informatiser les activités prescrites dans la modélisation organisationnelle des traitements (phases, tâches) compte tenu :

- des ressources et contraintes logiciel et matériel,
- des principes généraux d'ergonomie.

Ces modèles logiques de traitements (MLT) doivent permettre la prise en compte de choix techniques liés soit à l'architecture, notamment la répartition des traitements et des données, soit au poste de travail lui-même. Pour l'architecture technique, c'est la mise en œuvre d'un système de gestion de bases de données (SGBD), de bases de données réparties, d'architectures client-serveur, ... Pour le poste de travail, c'est d'une façon plus générale la prise en compte de nouvelles tendances du génie logiciel, permettant une meilleure ergonomie du poste de travail en mettant à profit la richesse de nouvelles interfaces homme/machine ; c'est concevoir des applications respectant la séparation entre ces interfaces utilisateur et le noyau de l'application ; c'est la conception et la programmation par objets...

Ces MLT doivent spécifier avec rigueur et en détail le contenu des traitements informatisés associés à chaque tâche organisationnelle à informatiser afin de construire une ou plusieurs solutions informatiques.

La spécification des modèles logiques de traitements est fortement liée à l'architecture adoptée et surtout aux outils logiciels retenus pour la réalisation du SII. Ainsi sont apparus sur le marché, autour des systèmes de gestion de bases de données principalement relationnels, des environnement ou ateliers de quatrième génération permettant un développement rapide d'applications. Ces ateliers facilitent ce développement d'applications en proposant des objets logiciels de plus en plus complexes, véritable boîte à outils.

L'usage de ces environnements de développement de quatrième génération conduira à la formulation de MLT d'un très haut niveau de spécification et dans un formalisme approprié aux fonctionnalités de l'environnement retenu. Si le langage SQL est maintenant une norme à laquelle la quasi totalité des

SGBD relationnels se réfèrent, les environnements de développement de quatrième génération proposés par ces SGBD ne sont pas encore normalisés. Tout au plus peut-on constater une certaine convergence au niveau de la définition des objets complexes sur lesquels ces environnements s'appuient. Cet état de fait pose d'ailleurs des problèmes de portabilité des applications d'un environnement à l'autre.

Citons pour exemple les ateliers de développement (AGL – ateliers de génie logiciel) proposés autour de SGBD relationnels du marché. La mise en œuvre de tels ateliers conduit généralement à spécifier des MLT dans un formalisme de type graphe, enchaînant des objets complexes parfois appelés «frames». Ces frames présentent un aspect statique, les «formes», sortes d'écrans présentant des données, et un aspect dynamique ou comportemental gérant des menus en fonction d'événements, permettant d'activer des procédures sur la base de données, assurant l'enchaînement de frames entre eux.

Aussi, il est assez difficile de proposer un formalisme universel pour l'élaboration de MLT. Néanmoins, un certain nombre de tendances du génie logiciel s'affirment, par exemple le multifenêtrage, la séparation dans un traitement de la partie dialogue (interface homme/machine, machine/machine...) de la partie purement applicative. Signalons les travaux menés au sein de SEMA Group sur cette modélisation logique des traitements [Panet, Letouche, Peugeot 91] [Panet, Letouche 94]. La prise en compte de ces tendances nous conduit à proposer le formalisme développé dans le paragraphe suivant.

Formalisme de modélisation des traitements au niveau logique

Comme pour l'expression des modèles organisationnels de traitements, le formalisme proposé pour l'expression de modèles logiques de traitements est fondé sur le formalisme général de modélisation des traitements de Merise. Il s'agit d'adapter les concepts types du formalisme général aux préoccupations de ce niveau logique.

Pour décrire le niveau logique, le formalisme des traitements utilise les concepts suivants :

- la machine logique,
- l'événement/résultat – message,
- l'état,
- l'unité logique de traitements (ULT),
- la procédure logique.

La machine logique

Une machine logique type, ou machine logique, est définie comme un ensemble de ressources informatiques (matériel et logiciel) capables d'exécuter des traitements informatiques de façon autonome.

Bien que confondues dans la grande majorité des cas, il convient parfois de faire la distinction entre machine logique et machine physique. Une machine physique est un ensemble de matériels permettant d'assurer les fonctions de base de l'informatique (exécution de logiciel, mémorisation, entrées/sorties).

Ainsi une machine logique peut être :

- équivalente à une machine physique
  - micro autonome ou en réseau,
  - serveur,
  - mainframe ou mini avec terminaux passifs.
- composée de plusieurs machines physiques
  - mini et micro en émulation terminal passif,
  - mainframe et machine base de données.
- une partie de machine physique
  - machine virtuelle sur un mainframe.

Précisons que l'architecture courante client / serveur est constituée de deux machines logiques basées sur deux machines physiques.

Une machine logique type est décrite par ses caractéristiques techniques : type, puissance, capacité, ...

Une machine logique type peut se matérialiser par plusieurs occurrences sur le terrain qui ont, par définition, le même comportement. Par exemple, la machine logique type micro client est en soixante exemplaires.

A l'instar des postes du modèle organisationnel de traitements, les machines logiques permettent d'exprimer la répartition des traitements informatisés. La représentation graphique des machines logiques reprend donc celle des postes, à savoir des colonnes dans lesquelles sont représentées les unités logiques de traitement exécutées par la machine logique.

Remarque : La représentation graphique des machines logiques peut être optionnelle dans les cas suivants :

- une seule machine logique,
- répartition des traitements par nature d'ULT correspondant à des machines logiques dédiés (figure 12.1); exemple, d'une architecture client / serveur où les ULT d'accès aux données seraient toutes réalisées

sur le serveur de données et les ULT de présentation, de dialogue et de calcul seraient toutes réalisées sur la machine cliente (voir MLT répartis)

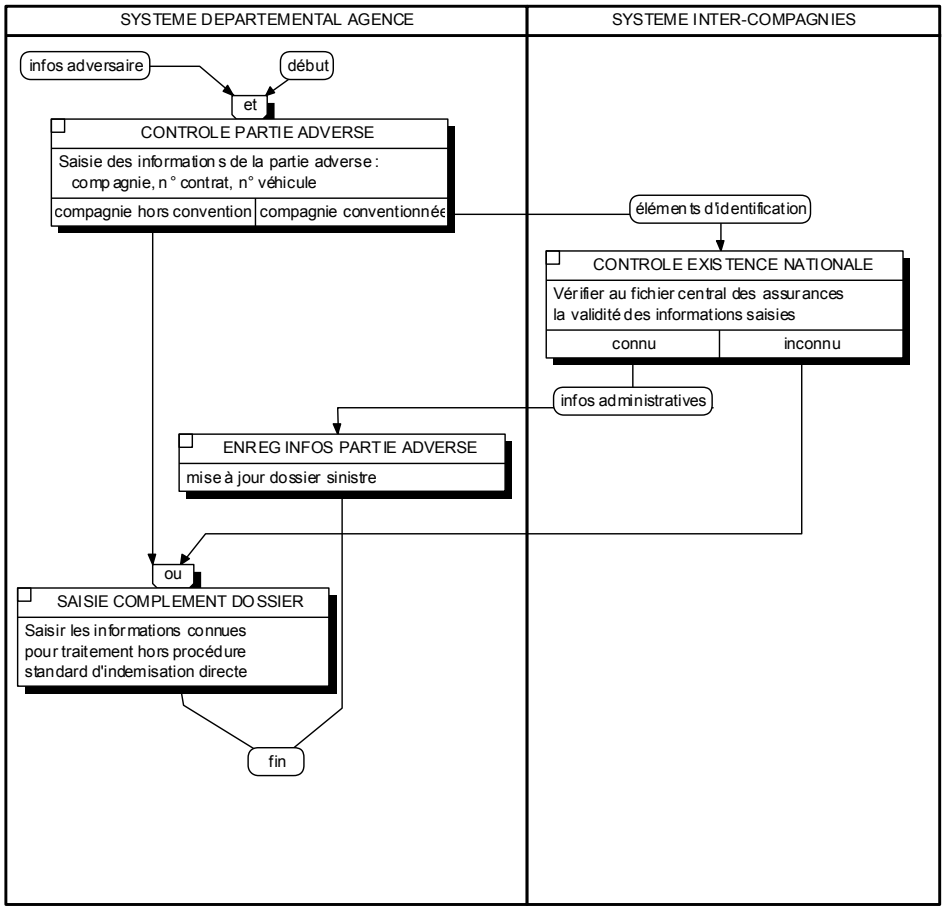


Figure 12.1 : Répartition des traitements entre des machines logiques

Le site

Le site est le lieu où sont physiquement installées les machines physiques, support des machines logiques. Dans la modélisation logique des traitements, la prise en compte des sites n'interviendra qu'à travers la répartition des machines logiques.

Par exemple, dans le cas d'une machine logique constituée d'un mainframe et de terminaux répartis sur tout le territoire, le site sera la localisation du mainframe.

L'événement/résultat-message

Dans le formalisme général de modélisation des traitements proposé par la méthode Merise, événements et résultats représentent l'échange de stimuli et de réponses par rapport au système d'information. Cependant, la nature des

événements et résultats est différente entre les niveaux conceptuel et organisationnel (S.I.O.), et le niveau logique (S.I.I.). Pour le système d'information organisationnel (S.I.O.), événements et résultats traduisent généralement les échanges du domaine avec son environnement ou entre les postes. Pour le système d'information informatisé (S.I.I.), événements et résultats expriment les échanges entre le S.I.O. (c'est à dire les utilisateurs) et le S.I.I. (figure 12.2).

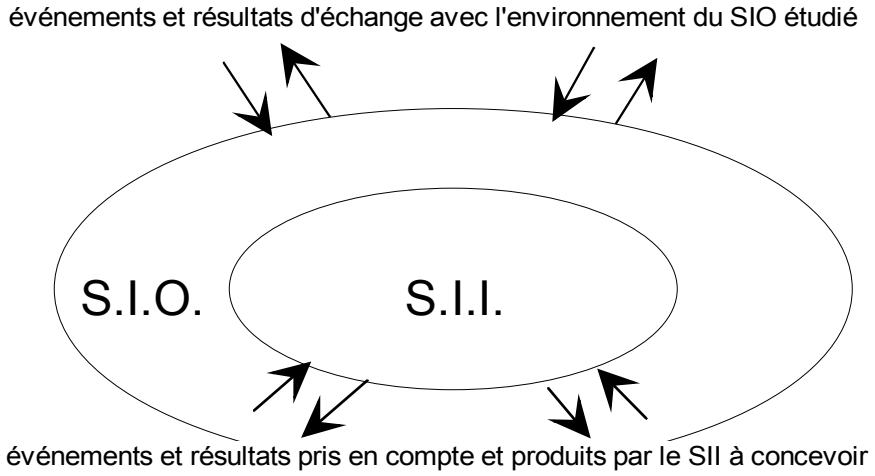


Figure 12.2 : Evénements et résultats du SIO et du SII

En conséquence, bien que reconduisant la symbolisation graphique précédemment utilisée, les événements et résultats du niveau logique pourront représenter :

- des apports ou production d'informations entre le SIO et le SII, où l'on retrouvera certains événements ou résultats du S.I.O.;
- des échanges entre des machines logiques ou des unités logiques de traitements;
- le lancement ou la fin des traitements informatisés déclenchées à l'initiative de l'utilisateur dans le cadre de ses tâches; on utilise ainsi fréquemment l'événement "début procédure x" et le résultat "fin procédure x".

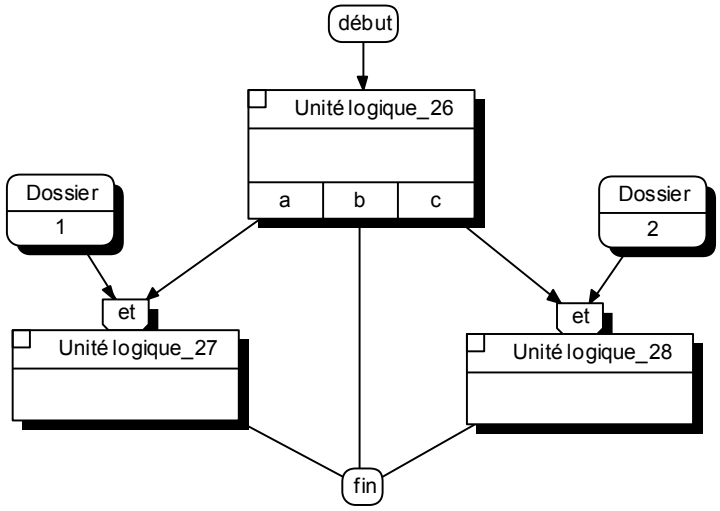
Remarque : Dans la modélisation logique des traitements, les événements et résultats n'ont pas d'acteurs émetteurs ou destinataires explicitement représentés. En fait, ce rôle d'acteur est joué par la ressource humaine impliquée dans les tâches interactives; les tâches automatisées étant généralement déclenchées par des événements temporels ou décisionnels qui n'ont déjà pas d'acteur formalisé au niveau organisationnel (implicitement le système de pilotage).

L'état

La modélisation des états reste identique à celle utilisée aux niveaux conceptuel et organisationnel. Ils expriment des conditions préalables ou des résultats conditionnels d'une unité logique de traitement.

Dans la modélisation logique des traitements, il existe deux tendances pour l'expression des états :

- Certains choisissent de ne pas représenter les états dans les modèles. Ils considèrent que, les états étant mémorisés dans les données, la prise en compte des états préalables ou résultants relève de règles de traitement faisant partie intégrante de l'unité logique et qui sont associées à des actions sur le sous-schéma de données (en lecture ou mise à jour). Ils argumentent que, lors de la réalisation, la gestion des états sera inclue dans la programmation associée à l'unité logique.
- D'autres optent pour une représentation explicite des états sur les modèles. Ils avancent qu'en particulier l'expression des états préalables permet de mieux formaliser la dynamique potentielle des enchaînements entre différentes ULT. Ils mettent en avant la possibilité ultérieure, dans les environnements orientés objets, de gérer les enchaînements extérieurement au noyau applicatif (figure 12.3).



Dans l'ULT 26 le passage vers l'ULT 27 ne sera disponible que si le dossier est à 1 etc....

Figure 12.3 : Expression des états pour un enchaînement dynamique

L'unité logique de traitement

L'unité logique de traitement type, ou unité logique (ULT), modélise un

ensemble de traitements informatiques perçus comme homogènes en termes de finalités.

L'unité logique de traitement se défini également par rapport à la cohérence des données du SII. Avant son démarrage, les données doivent être cohérentes (c'est à dire respecter l'ensemble des contraintes définies dans la base); durant son déroulement, les différentes actions sur les données envisagées peuvent provisoirement enfreindre cette cohérence; à l'issue de son exécution, quelque soit les conditions de sortie modélisées, l'ensemble des données mémorisées doivent retrouver une cohérence. Dans le cas où une ULT est interrompue ou suspendue, son contexte doit être conservé. Notons que cette notion de cohérence prépare les préoccupations de mécanismes de mises à jour concurrentielles et de verrouillages (COMMIT, Logical Unit of Work dans les SGBD relationnels).

Ce respect de la cohérence et l'homogénéité du traitement envisagé vont guider le concepteur dans le découpage en ULT.

Une unité logique de traitement peut modéliser par exemple :

- une transaction dans un système transactionnel classique,
- une boite de dialogue,
- une édition,
- un module dans une chaîne batch.

La symbolisation graphique de l'ULT reprend celle du formalisme général des traitements de Merise déjà utilisée pour l'opération et la tâche avec les concepts secondaires associés : synchronisation, description et conditions d'émission.

Description et composition d'une unité logique de traitement

L'ULT est décrite par l'ensemble des traitements informatiques homogènes à réaliser qui peuvent être décomposés selon leur nature :

- Interface,
- Traitements,
- Données.

Ces trois niveaux, aujourd'hui communément admis et particulièrement utilisés dans la répartition, peuvent également se décomposer en fonctions suivantes :

- *présentation* externe des données utilisées,
- règles de gestion et de contrôle associées à la présentation ou *logique de dialogue*,
- algorithmique générale de l'ULT ou *logique fonctionnelle*,
- *règles de calcul* ou procédures à appliquer,
- *accès aux données* mémorisées à travers un sous-schéma de données,

- *enchaînements* conditionnels vers d'autres ULT ou résultats produits représentés par les conditions de sortie.

La figure 12.4 illustre l'articulation de ces différents composants :

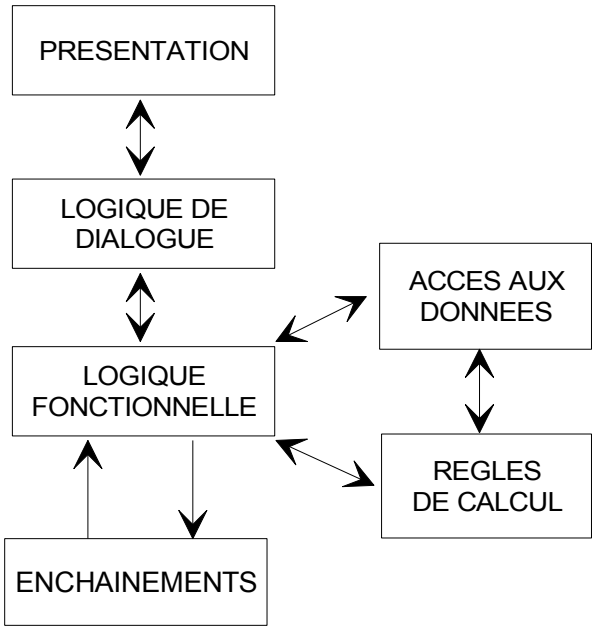


Figure 12.4 : Articulation fonctionnelle de l'ULT.

Suivant le degré de détail souhaité, le concepteur peut représenter une ULT soit au sein d'une procédure organisationnelle (figure 12.7) avec une description sommaire des traitements à réaliser mais en privilégiant l'enchaînement, soit en "gros plan" avec l'expression détaillée de la présentation, de la logique de dialogue, de la logique fonctionnelle, des règles, du sous-schéma de données et des enchaînements. Dans ce dernier cas, la représentation graphique n'a certes qu'un intérêt limité et de nombreux concepteurs en font d'ailleurs l'économie; il ne faut pas pour autant hâtivement en déduire que la modélisation logique des traitements n'a que peu d'intérêt dans la méthode Merise.

Remarque : Toutes les ULT ne comprennent pas systématiquement tous les composants sus-cités. Dans certains cas (décomposition, répartition, nature de l'ULT) certains composants peuvent être absents.

La présentation

La présentation, associée à sa logique de dialogue, constitue l'interface homme - machine. Cette composante de l'unité logique de traitement est un aspect très important de la conception d'un système d'information informatisé. Elle est le point de contact privilégié entre l'utilisateur et la partie informatisée de son système d'information; aussi son ergonomie, parfois négligée dans le passé, doit

être particulièrement étudiée. Un paragraphe spécifique aborde plus loin les différents aspects de l'ergonomie des interfaces homme - machine.

La présentation est la partie externe visible (voire auditive) de l'interface. Selon les types d'environnement, elle se concrétise sous la forme :

- d'un écran utilisant des objets alphanumériques activables par l'intermédiaire d'un clavier,
- d'une fenêtre utilisant des objets alphanumériques ou graphiques activables par l'intermédiaire d'un clavier ou d'une souris,
- d'une édition sous forme d'état ou de formulaire.

La formalisation de la présentation s'effectue par un dessin d'écran, de fenêtre ou d'édition. auquel est joint un descriptif précisant le contenu et les caractéristiques de chaque objet.

Dans l'élaboration de la présentation, le concepteur doit également tenir compte :

- des normes et standards de la profession (AFNOR, CUA, OSF),
- des éventuels guides de style spécifique à son entreprise,
- des possibilités et contraintes liées aux outils de réalisation.

La validation d'une présentation par les utilisateurs peut certes être réalisée sur papier. Toutefois, l'aspect démonstratif d'une maquette est un atout indéniable et est aujourd'hui la solution recommandée.

La logique de dialogue

La logique de dialogue comprend l'ensemble des règles de gestion et de contrôle associées à la présentation. En sont exclus, les algorithmes et les accès aux données.

Les règles de gestion de l'interface expriment :

- des actions sur le clavier,
- des actions sur des objets graphiques,
- la dynamique de la présentation.

Les règles de contrôle de l'interface correspondent à :

- des contrôles sur les données de la présentation (sans faire d'accès aux données de la base)
- des calculs élémentaires sur les donnée de la présentation.

La formalisation de la logique de dialogue s'effectue sous une forme textuelle (formulaire ad hoc ou pseudo-code proche d'un L4G).

Le maquettage dynamique (voir chapitre 18 de la partie IV) permet une validation simultanée de la présentation et du dialogue.

La logique fonctionnelle

La logique fonctionnelle représente l'algorithmique générale de l'ensemble des traitements à effectuer et constitue la "colonne vertébrale" de l'ULT. Son rôle central de coordination est illustré par sa position dans l'articulation des différentes fonctionnalités (figure 12.4).

Outre la gestion de l'enchaînement des traitements au sein de l'ULT, la logique fonctionnelle assure les échanges :

- avec la partie logique de dialogue (appel, transfert des données),
- avec la partie accès aux données (demande, récupération),
- avec la partie règles de calcul (lancement, récupération),
- avec la partie enchaînements (appel d'autres ULT, retour d'ULT appelée).

La formalisation de la logique fonctionnelle peut s'effectuer sous une forme graphique (type organigramme ou MLT analytique décomposé) ou sous une forme textuelle (langage naturel ou pseudo-code proche d'un L4G).

Les règles de calcul

On reconduit, au niveau de l'ULT, la notion de règle de calcul telle qu'elle a déjà été formalisée dans la modélisation organisationnelle des traitements. Précisons que les règles de contrôle et de calcul élémentaires sont déjà exprimés dans la partie logique de dialogue. En conséquence, ne subsistent dans cette partie que les règles :

- présentant une algorithmique suffisante,
- ne nécessitant pas d'interaction directe avec la présentation,
- échangeant éventuellement des données avec la partie sous-schéma.

Ces règles de traitement ou procédures s'expriment par un algorithme, ensemble d'expressions arithmétiques et/ou logiques enchaînées suivant une structure de calcul.

Ces règles de traitement comportent :

- des variables représentées ici par des informations provenant du dialogue ou du sous-schéma de données;
- des constantes dont les valeurs sont propres au contexte ;
- des opérateurs arithmétiques ou logiques,
- des règles de traitements, permettant ainsi une construction arborescente des règles de traitement avec possibilité de réutilisation de règles élémentaires.

La méthode Merise ne propose pas à ce jour de formalisme propre pour la

description des algorithmes. Nous conseillons d'utiliser les concepts et règles de la programmation structurée qui privilégie les trois structures de contrôle suivantes :

- le bloc,
- l'alternative,
- la répétitive.

Le contenu détaillé de la règle peut être exprimé sous la forme de langage structuré (naturel ou pseudo-code proche d'un L4G).

Les règles spécifiques définies dans cette partie peuvent être "catalogués" comme le proposent aujourd'hui les environnement client / serveur de 2ème génération.

Le sous-schéma logique de données

L'accès aux données comprend :

- le sous-schéma logique de données,
- les actions effectuées sur les données mémorisées,
- les contrôles de cohérence et d'intégrité des données.

Un sous-schéma logique de données (voir figure 12.5) est un sous-ensemble de tables et d'attributs défini sur le modèle logique de données (MLD), et associé à une unité logique. Cette ULT effectue des actions (mise à jour ou lecture) sur les occurrences de ces tables, tout en respectant la cohérence et l'intégrité de la base. Le sous- schéma logique est également appelé vue logique.

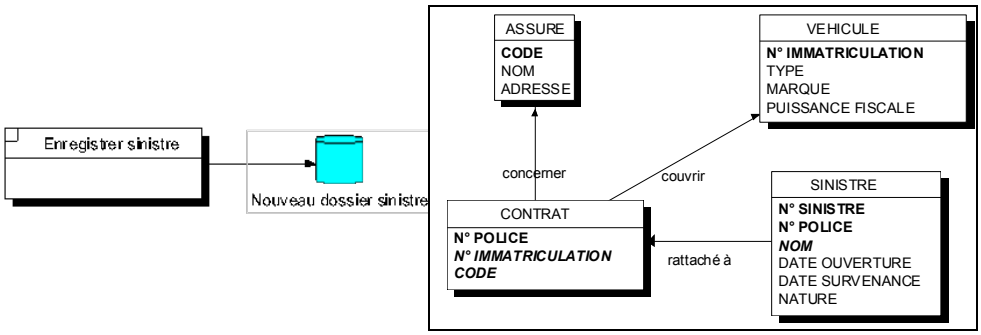


Figure 12.5 : Sous schéma de données logique associé à l'ULT Enregistrer sinistre

Sur le diagramme d'une procédure logique, le sous-schéma logique peut être représenté graphiquement par une icone associée à son nom (voir figure 12.5). Dans une description détaillée, on exprime le sous-schéma logique en associant le schéma de données à un tableau récapitulatif des actions types envisagées (voir figure 12.6).

Table	Action
ASSURE	Lire
CONTRAT	Lire
VEHICULE	Lire
SINISTRE	Créer

Figure 12.6 : Actions sur le sous-schéma de données « Nouveau dossier sinistre »

La cohérence et l'intégrité des données sont assurés, dans une proportion variable suivant les technologies utilisées :

- par le système de gestion de base de données,
- par des règles spécifiques exprimées dans la partie accès aux données.

Remarque : Bien qu'il soit théoriquement normal d'associer ULT et sous-schéma logique, de nombreux praticiens souhaitent conserver l'expression du sous-schéma dans le formalisme entité - relation, du moins dans l'étape d'étude détaillée au contact avec l'utilisateur.

Les enchaînements

Ils assurent les liaisons entre les différentes ULT d'un MLT. Ils représentent :

- les origines des appels de l'ULT (événements logiques),
- les liaisons conditionnelles vers d'autres ULT (résultats logiques).

Ces enchaînements peuvent s'appuyer soit sur les procédures logiques, soit sur la structure logique de l'application (voir ci-après).

L'enchaînement prend en charge le transfert d'informations éventuellement nécessaire entre les ULT. Ce rôle deviendra d'ailleurs important dans le cas de répartition des traitements qui conduira à une segmentation et une spécialisation des ULT.

Remarque : Il ne faut pas confondre l'enchaînement des différents traitements au sein d'une ULT exprimé dans la logique fonctionnelle, et l'enchaînement entre des ULT distinctes exprimé ici.

La procédure logique

C'est un enchaînement d'ULT réalisant l'informatisation d'une tâche ou phase du modèle organisationnel.

Le début de la procédure représente l'appel par l'utilisateur du menu ou de la fonction de l'application correspondant à la tâche. La fin de la procédure correspond au retour au menu de l'application permettant le lancement d'une autre procédure. Au sein de la procédure, les ULT disponibles et leurs enchaînements correspondent à la résolution de l'activité organisationnelle associée. La figure 12.7 illustre la procédure logique associée à la tâche Contrôle situation assuré présentée dans la procédure organisationnelle figure

8.9.

Remarque : Au niveau logique, il convient de distinguer deux présentations d'enchaînement d'ULT :

- l'enchaînement désigné par procédure logique qui représentent les traitements informatisés nécessaires à l'exécution d'une tâche ou phase organisationnelle;
- l'ensemble des enchaînements possibles entre les ULT correspondant à la structure logique de l'application qui est une représentation descriptive informatique.

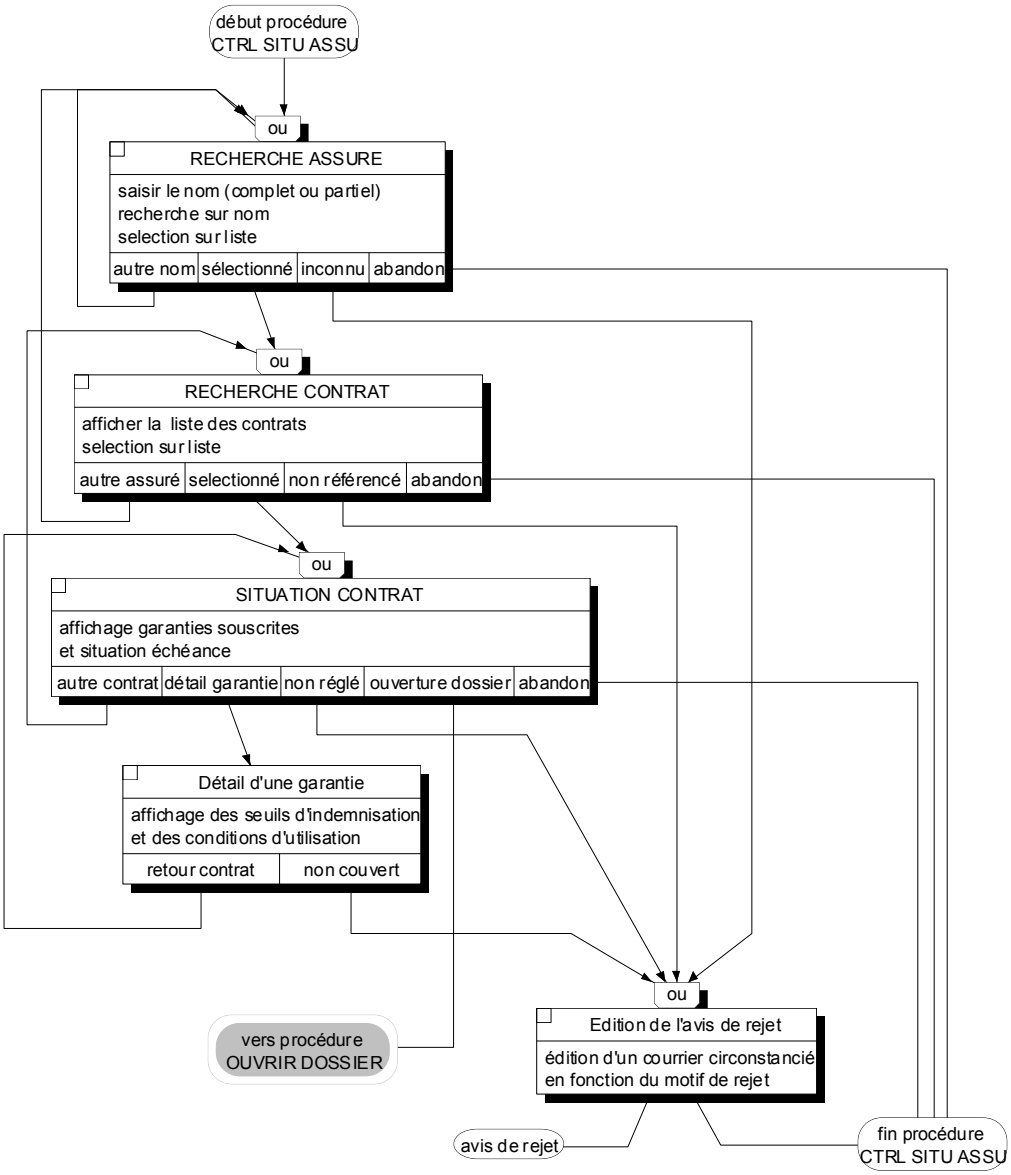


Figure 12.7 : Procédure logique « Contrôle situation assuré »

La présentation en procédure logique n'a pratiquement d'intérêt que dans le cas d'enchaînements suffisamment complexes d'ULT. Dans le cas où la procédure se réduit à une ou deux ULT, sa formalisation est superflue; on se satisfait alors de la description du contenu de chaque ULT (présentation, dialogue, règles, actions sur les données, enchaînements). Cette simplification de la présentation des procédures logiques est particulièrement constatée en cas d'utilisation de maquettage dynamique.

Cohérence sous-schéma organisationnel et sous-schémas logiques de données

Dans le cadre d'une procédure logique, associée à une tâche organisationnelle, il est nécessaire de contrôler la cohérence entre le sous-schéma de données organisationnel de la tâche et les sous-schémas logiques des ULT constituant la procédure logique.

On vérifie que la réunion des sous-schémas logiques des ULT de la procédure logique issue de la tâche, est équivalente à la transformation logique du sous-schéma conceptuel/organisationnel de données associé à la tâche.

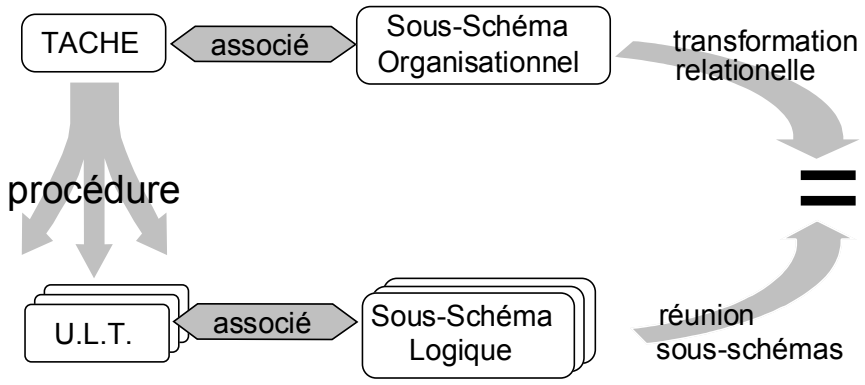


Figure 12.8 : Cohérence sous-schéma organisationnel et logiques

Conception des modèles logiques de traitements

Dans le processus de conception du SII, la méthode Merise présente une différence importante d'approche entre les données et les traitements. L'élaboration d'un modèle logique de données sera obtenue de façon algorithmique à partir de la modélisation conceptuelle et organisationnelle des données (voir chapitre 13). Par contre, la construction d'un modèle logique de traitements exige dans tous les cas une réflexion, une création, une invention; elle ne peut pas être directement et automatiquement déduite des modélisations effectuées dans le SIO; tout au plus la modélisation organisationnelle des traitements pourra servir au concepteur de cadre contextuel de réflexion.

Pour élaborer des modèles logiques de traitements, nous proposons d'abord trois approches qui s'avèrent complémentaires dans une mise en oeuvre pratique et possèdent chacune des avantages et des inconvénients :

- la décomposition des tâches du MOT,
- la recherche de réutilisation d'ULT,

- la conception d'ULT autour des données.

Par ailleurs, l'état de l'art des années 90 en génie logiciel propose de nouvelles architecture logiques d'application permettant une approche plus modulaire de la modélisation logique des traitements, facilitant ainsi leur répartition en fonction des technologies disponibles.

Décomposition des tâches organisationnelles

Le concepteur élabore les procédures logiques à partir des préoccupations exprimées dans les phases ou tâches informatisées du MOT; il est essentiellement guidé par la description des tâches. Pour chaque tâche, il construit les enchaînements et contenus des ULT les plus appropriés à la tâche étudiée.

Dans cette approche, la procédure logique apparaît comme une décomposition de la tâche; on reconduit ainsi globalement le processus utilisé pour passer du MCT au MOT.

On peut supposer que la procédure logique présentée à la figure 12.7 a été imaginée suivant cette approche à partir de la tâche correspondante présentée à la figure 8.9.

Avantages :

On peut concevoir des ULT très proches des activités formulées dans la description de la tâche et bien adaptées aux conditions d'utilisation exprimées dans la procédure organisationnelle (tâches amont et aval, environnement du poste).

Inconvénients :

L'approche n'est praticable que dans des situations où sont établies des procédures organisationnelles, par exemple dans des systèmes d'information de production (voir chapitre 4 «Typologie des systèmes d'information»).

Les ULT ainsi conçues peuvent également être trop spécifiques à une organisation et n'offrir que peu de souplesse organisationnelle.

On risque enfin une démultiplication d'ULT similaires qui compliquera ultérieurement la réalisation.

Réutilisation des ULT

Cette approche s'inspire de principes préconisés en génie logiciel visant à limiter la multiplication de fonctions applicatives similaires voire identiques afin d'améliorer l'économie du développement et la maintenance.

Dans cette approche, d'une part, le concepteur recherchera dans le MOT des tâches dont la description est similaire ou proche; il pourra ainsi concevoir des ULT utilisables en commun par ces différentes tâches.

D'autre part, lors de la construction du MLT, éventuellement par procédure logique, il s'efforcera de réutiliser des ULT déjà existantes quitte à procéder éventuellement à leur amendement. Cette "banque d'ULT" s'enrichira progressivement au sein du même projet, voire au delà d'un projet spécifique.

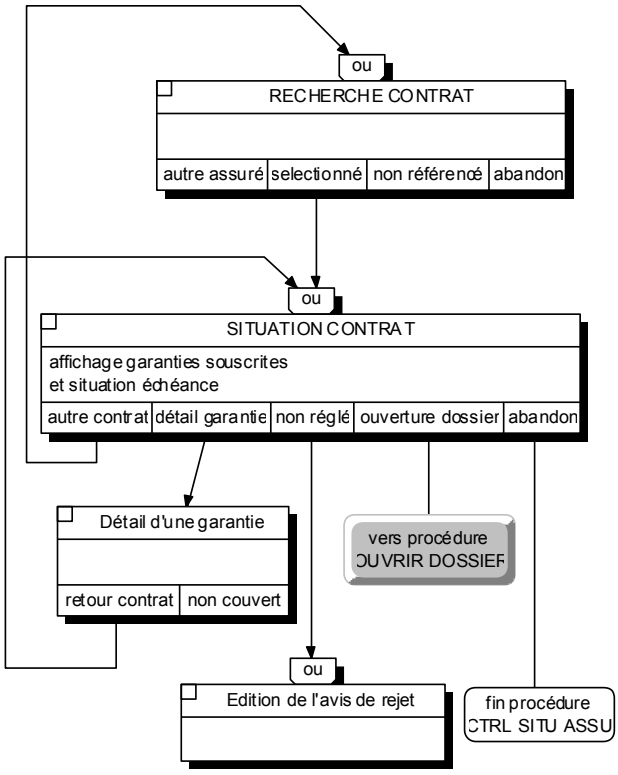
Avantages :

Une diminution du nombre d'ULT spécifiées est normalement espérée, allégeant les charges de réalisation et de maintenance ultérieures.

Inconvénients :

La mise en commun de préoccupations certes proches mais parfois légèrement différentes par leur contexte peut conduire à une moindre adaptation de l'ULT à la tâche associée.

L'implication d'une même ULT dans différentes procédures logiques pose le problème illustré par la figure 12.9.



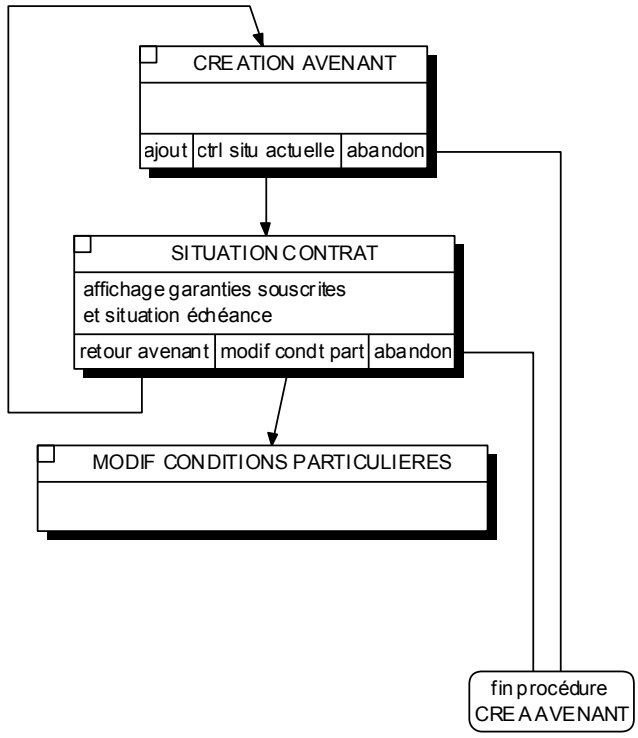


Figure 12.9 : ULT multi procédure

L'ULT Situation contrat est partagé par la procédure Contrôle situation assuré (présentée intégralement figure 12.7) et dans la procédure Avenant. Bien qu'ayant un contenu identique, cette ULT est soumise à des conditions

contextuelles d'appel et d'enchaînement différentes. En fait, il faut définir une ULT Situation contrat de "fusion" avec l'ensemble des appels et enchaînements possibles.

Dans la pratique, pour pouvoir réaliser cette réutilisation d'ULT multi-procédures, les différences contextuelles entre les ULT doivent être faibles. Nous suggérons les critères suivants :

- présentation identique,
- même sous-schéma avec éventuellement des actions différentes,
- règles partagées identiques, quelques règles spécifiques,
- appels et enchaînements différents.

La trop grande multiplicité de contexte d'utilisation d'une même ULT peut cependant compliquer ultérieurement le travail du réalisateur.

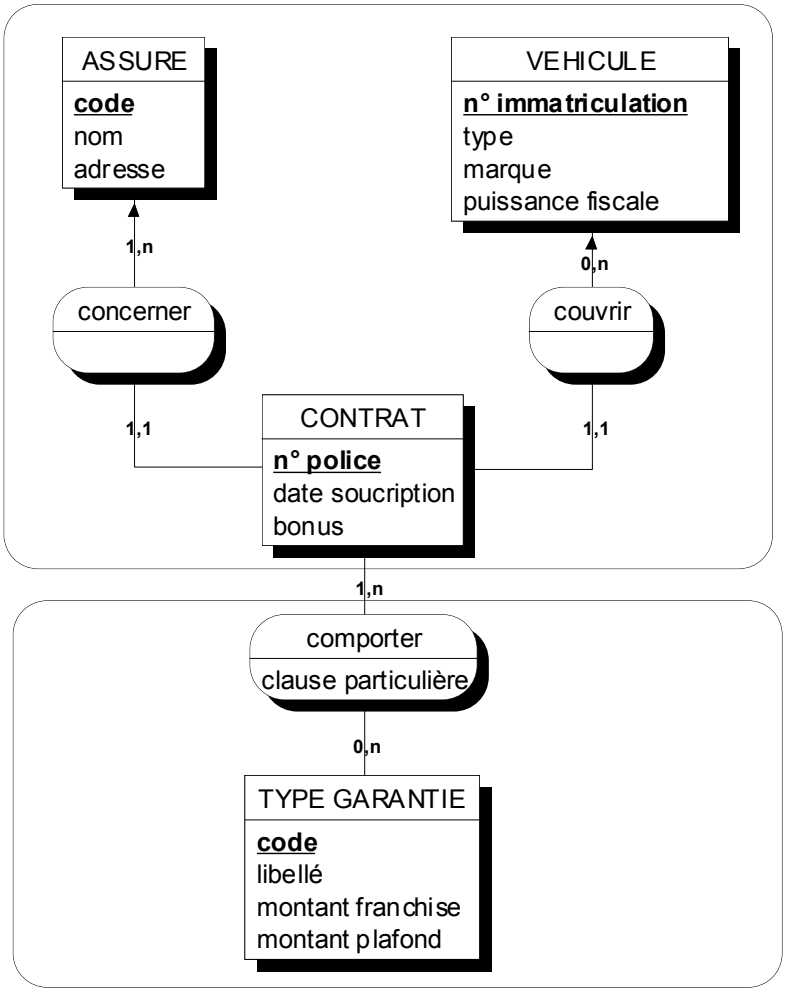
Conception des ULT autour des données

Cette approche cherche à être la plus indépendante possible du contexte d'organisation en s'appuyant sur des ensembles de données perçus comme stable par l'utilisateur, référant à des objets utilisés dans ses activités courantes.

Ces objets utilisateurs correspondent généralement au concept d'entité externe (voir Confrontation détaillée Données Traitements Chapitre 10), c'est à dire une entité jouant le rôle de pivot liée à des entités par des relations binaires fonctionnelles. Des objets utilisateurs complexes constituant de véritables modèles externes, peuvent se composer d'une entité externe principale (désignant l'objet) et d'une entité externe dépendante formant une structure de composition Cette notion d'objet utilisateur, également évoquée par dans Merise/2 [Panet, Letouche 94] s'apparente aussi à celle d' "objet naturel" [Bres 91].

Dans cette approche, le concepteur recherchera, dans le MCD ou le MOD, des entités dont l'appellation évoque des notions principales du domaine autour desquelles il construira ces modèles externes. Il associera à ce sous-schéma une ULT qui permettra d'effectuer les actions de base (création, modification, suppression, consultation).

La figure 12.10 présente quelques modèles externes correspondant à des objets utilisateurs définissables a priori sur le modèle conceptuel de données du cas Assurance (figure 10.3)


ASSURE
code
nom
adresse
VEHICULE
n° immatriculation
type
marque
puissance fiscale
1,n
0,n
concerner
couvrir
1,1
1,1
CONTRAT
n° police
date soucription
bonus
1,n
comporter
clause particulière
0,n
TYPE GARANTIE
code
libellé
montant franchise
montant plafond

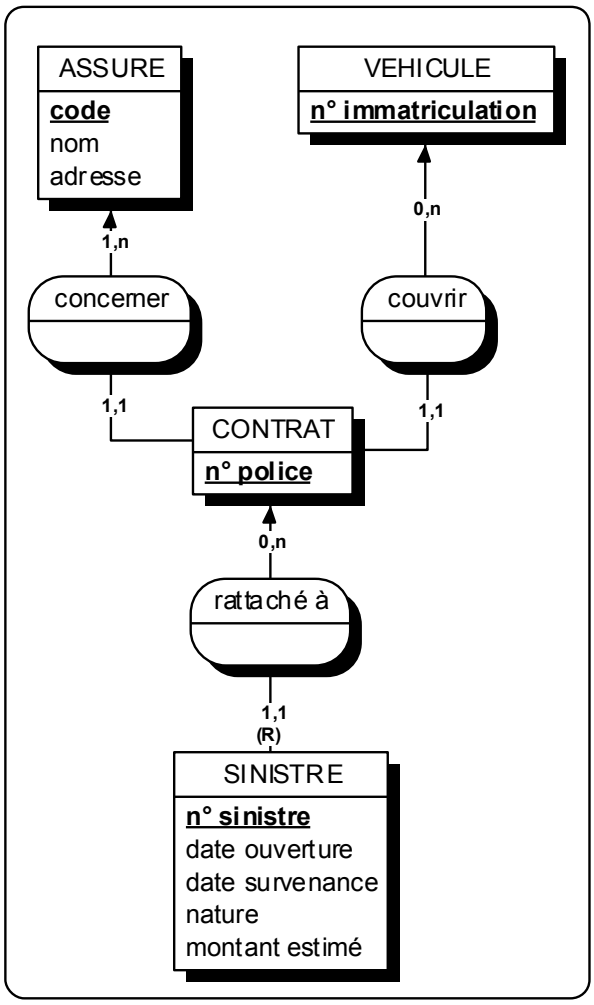


Figure 12.10 : Exemples d'objets utilisateurs

Avantages :

Ces ULT peuvent servir d'éléments de base pour constituer une approche par réutilisation.

Les ULT ainsi définies peuvent être directement implémentées dans des environnement graphiques en dehors d'approche procédurale.

Inconvénients :

Cette approche laisse évidemment la construction des procédures logiques à l'initiative des utilisateurs qui doivent maîtriser les enchaînements adéquats au problème à résoudre.

Modularité du MLT

ULT et Architecture logique d'application

Nous avons vu en introduction que les modèles logiques de traitements devaient pouvoir prendre en compte les nouvelles tendances informatiques tant en matière de répartition des données et des traitements qu'en matière d'interface homme/machine liée au poste de travail.

Il s'agit en conséquence de pouvoir concevoir des applications respectant la séparation de ces interfaces utilisatrices du noyau de l'application, afin d'avoir une indépendance entre le dialogue interactif (présentation, interaction) et le noyau de l'application. Cette indépendance est nécessaire pour :

- Développer plusieurs interfaces différentes pour un même noyau applicatif, permettant alors de travailler par exemple sur un parc matériel hétérogène au niveau des postes de travail.
- Maintenir le noyau applicatif sans avoir à toucher à l'interface et inversement.

L'approche que nous préconisons s'appuie sur un modèle d'architecture logique d'application proposé dans Merise/2 [Panet, Letouche 94] et repris de SAA d'IBM, qui distingue trois modules de base:

- l'interface homme - machine,
- le noyau applicatif,
- le guidage fonctionnel.

Dans une architecture classique transactionnelle (type CICS, TDS,...) ces modules recouvrent les fonctionnalités de base de l'ULT précédemment décrites comme le présente la figure 12.11. Une architecture plus récente liée aux environnements graphiques, est représentée par la figure 12.12.

La répartition des traitements entre plusieurs machines logiques permettra par ailleurs (voir MLT répartis) de nouvelles variantes d'architecture logique d'application.

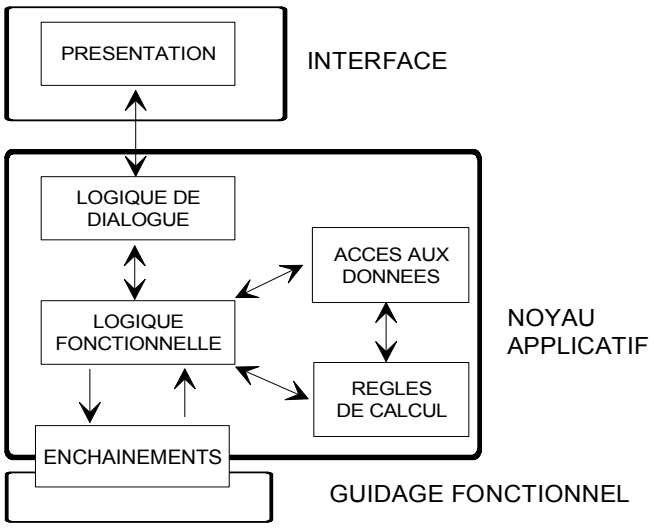


Figure 12.11 : Architecture logique d'application classique et fonctionnalités d'une ULT

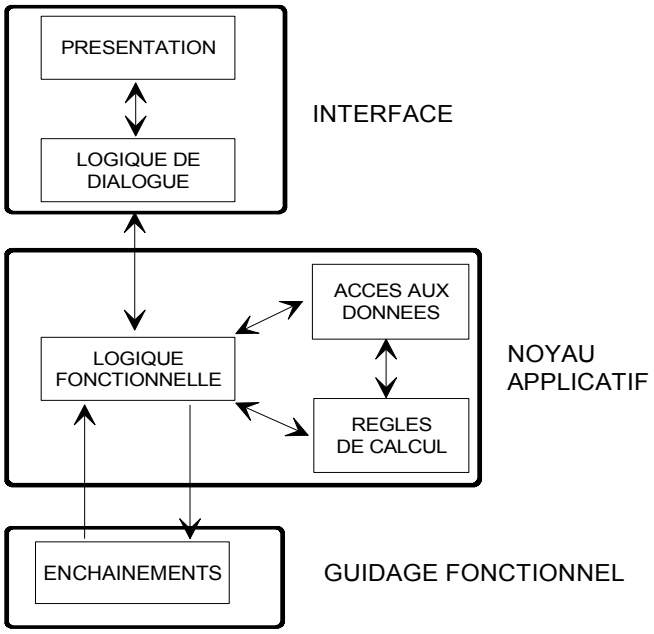


Figure 12.12 : Architecture logique d'application graphique et fonctionnalités d'une ULT

Décomposition des ULT par nature

Nous avons vu d'une part que la description d'une ULT pouvait s'analyser selon différents composants de nature différentes (voir figure 12.4), et d'autre part que l'architecture logique d'application se décomposait en trois modules. La réalisation de ces modules et de leurs communication ainsi que la spécification détaillée de l'ULT nécessite parfois de pouvoir formaliser plus

finement la logique interne de l'ULT.

Ce détail est obtenu par la décomposition de l'ULT en ULT plus fines. Nous retrouvons ici le principe de décomposition hiérarchique (refinement) présenté dans la première partie (voir chapitre 2 « Modularité des modèles conceptuels de traitements »), également appelé MLT analytique dans Merise/2.

Dans cette décomposition, les ULT élémentaires doivent respecter les règles suivantes :

- unité de nature (présentation, dialogue, logique fonctionnelle, accès aux données, règles, enchaînement)
- unité de machine logique en cas de répartition.

La formalisation est celle d'un enchaînement d'ULT, similaire à celui de la procédure logique où l'on précise la nature de chaque ULT élémentaire. Ce diagramme enchaînement des ULT élémentaires constitue déjà en lui même une partie de la logique fonctionnelle de l'ULT.

Les ULT élémentaires sont éventuellement réutilisables et peuvent ainsi intervenir dans la composition de plusieurs ULT.

Ces ULT élémentaires sont également appelées par d'autres auteurs primitives (de dialogue, d'accès aux données, de calcul).

La figure 12.13 illustre la décomposition de l'ULT Recherche assuré présente dans la procédure logique Contrôle situation assuré (figure 12.7); la nature de chaque ULT élémentaire est indiquée en abrégé.

Ultérieurement, en fonction du type d'architecture logique d'application retenu, le concepteur rassemblera dans les différents modules, les ULT élémentaires selon leur nature. Ainsi, sur la base d'une architecture type présentée à la figure 12.12, le module d'interface comporterait les ULT élémentaires de présentation et de dialogue; les autres ULT constituant le noyau applicatif. Les liens entre des ULT élémentaires appartenant à des modules différents se traduiraient par des échanges entre modules. On constatera qu'avec une architecture classique, l'ensemble des ULT (hormis la présentation) se retrouveraient dans le noyau applicatif.

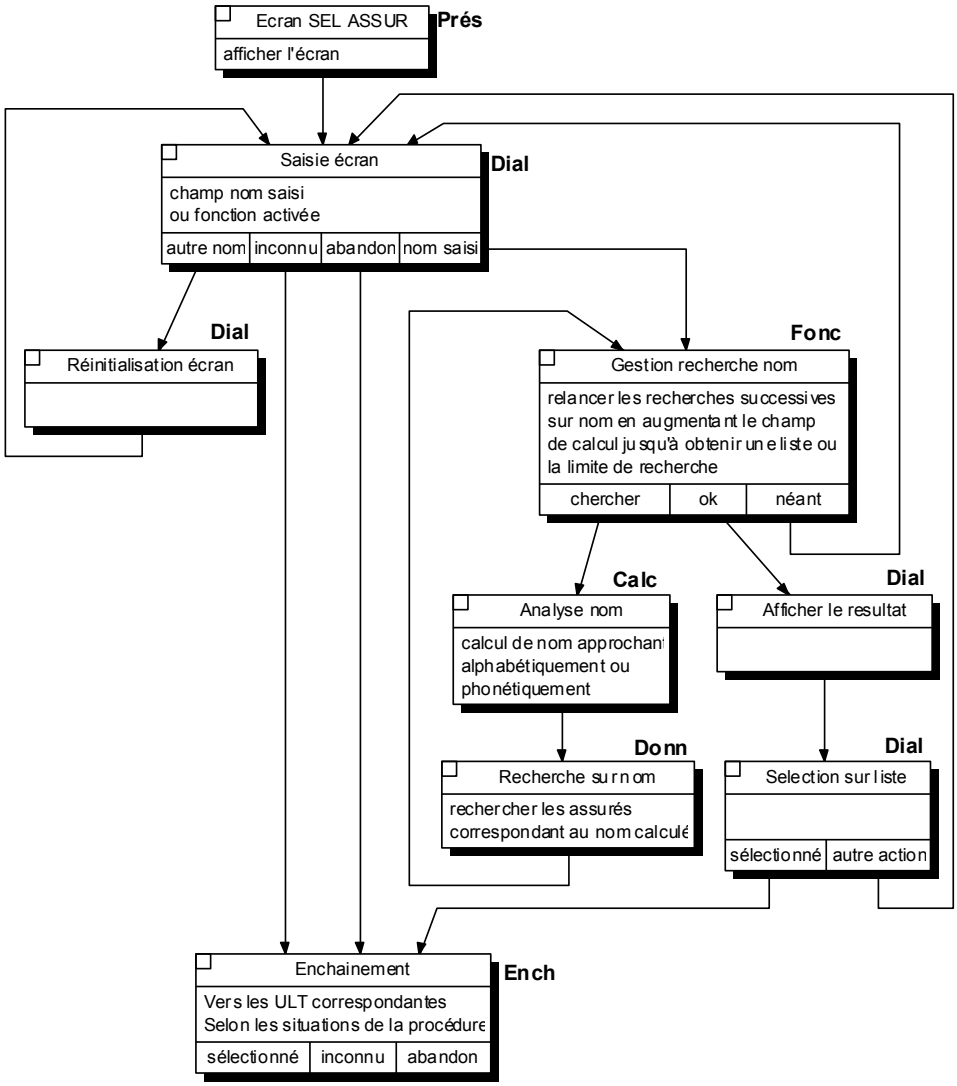


Figure 12.13 : Décomposition de l'ULT Recherche assuré en ULT élémentaires

Remarque : La décomposition d'une ULT est en pratique réservée à la formalisation

- d'ULT à contenu assez complexe,
- d'architecture logique d'application nécessitant une distinction des modules,
- de répartition liée à des traitements coopératifs.

Nous avons vu dans la deuxième partie, relative à l'étude du SIO (système d'information organisationnel), un premier type de répartition des traitements exprimée dans les MOT. Cette répartition portant sur les tâches s'effectuait tout d'abord en répartissant les tâches dans des postes de travail, et ensuite en définissant pour chaque tâche un type d'automatisation, d'informatisation (voir début du chapitre 8).

La répartition logique des traitements (voir figure 12.14) concerne le SII (système d'information informatisé) et porte sur les unités logiques de traitements associées aux tâches informatisées du MOT. Cette répartition logique est rendue possible par les possibilités de communications de plus en plus faciles entre matériels et logiciels hétérogènes, selon des protocoles normalisés.

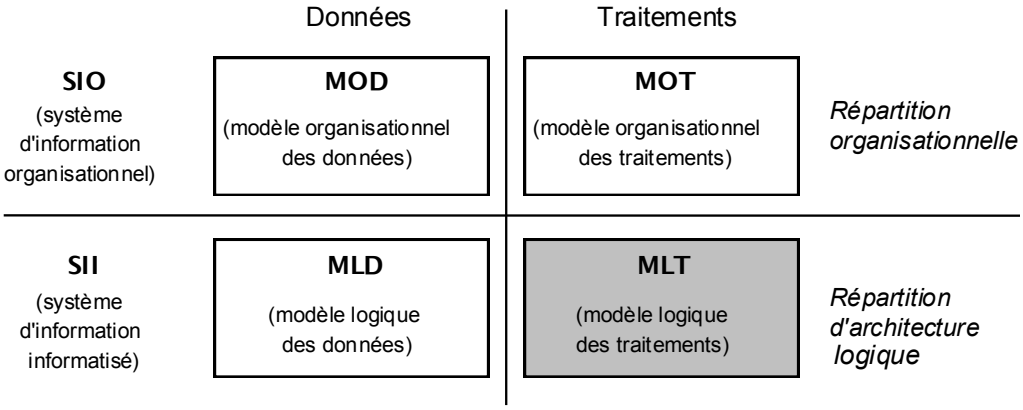


Figure 12.14 : Répartition logique des traitements.

Ces possibilités de communications font que des solutions de répartition des traitements permettent par exemples de mettre en commun des informations sur des serveurs dédiés, d'accéder à partir d'une machine à des données réparties ou de mettre à profit l'ergonomie des interfaces graphiques disponibles sur les environnements micro.

Démarche de répartition

Pour construire un MLT réparti, nous suggérons d'adopter la démarche suivante.

Elaborer un MLT non réparti

La construction de ce premier modèle s'effectue sans tenir compte de la future répartition ; elle est avant tout tournée vers l'utilisateur d'une part dans la définition de l'interface et des fonctions à: assurer, d'autre part dans la réponse donnée aux tâches organisationnelles.

Cette modélisation est commune à tout projet. Elle reprend tous les principes exposés dans les paragraphes précédents, avec en particulier l'analyse du contenu de chaque ULT selon ses différentes fonctionnalités. Cette distinction entre :

- la présentation,
- la logique de dialogue,
- la logique fonctionnelle,
- les règles de traitement,
- l'accès aux donnée,
- les enchaînements,

permet d'abord, dans tous les cas, une meilleure structuration de la future application distinguant ses différents composants (interface, noyau applicatif, guidage). Elle prépare ensuite la future répartition lorsqu'elle s'effectuera par nature de fonction (cas d'une architecture client / serveur spécialisée par nature).

Définir une architecture matérielle

La problématique de conception et de formalisation de l'architecture technologique des systèmes d'information ne figure pas dans la démarche et raisonnements de la méthode Merise. D'ailleurs, des méthodes spécifiques sont venu ultérieurement s'adapter à Merise, en particulier la méthode TACT [Alalk, Lalanne 89].

Cette méthode permet, entre autre, de définir :

- les *machines logiques* et leurs caractéristiques techniques,
- les *sites logiques*,
- les ressources d'environnement (systèmes d'exploitation, logiciel de développement, communications).

Dans certains cas, les machines logiques peuvent être spécialisées autour d'une fonction particulière ou service : gestion des données, gestion de la présentation, gestion des impressions, gestion des communications. Ces machines logiques dédiées sont appelées *serveurs logiques*.

Répartir les traitements

La répartition des traitements formalisés en MLT s'effectue en affectant les différentes ULT aux machines logiques qui les prennent en charge. La formalisation s'exprime à travers des procédures logiques dont les ULT sont distribuées suivant les machines logiques. La figure 12.1 illustre une présentation de MLT réparti.

Actuellement, les répartitions les plus couramment utilisées sont celles

permises par les architectures client / serveur, en particulier celles de serveur de données. Dans le cas où un serveur dédié prend en charge la totalité des fonctionnalités de même nature au sein d'une application, (par exemple la gestion des données), on pourra faire l'économie d'une représentation du MLT réparti.

Modalités de répartition

La répartition des traitements et des données d'une application entre différentes machines existe depuis le courant des années 80. Toutefois, les années 90 ont vu ces possibilités se développer grâce en particulier à la normalisation des protocoles de communication aux différents niveaux (middleware) et se concrétiser par l'expansion des architectures clients serveurs

Répartir des traitements et les données accédées, c'est les installer sur des machines logiques distinctes. Selon les architectures technologiques, on disposera de différentes modalités de répartition.

Traitements coopératifs

Un traitement est coopératif lorsque une ULT primaire (non décomposée en ULT élémentaires, c'est à dire correspondant à une transaction utilisateur) est exécutée sur plusieurs machines logiques. Cela exige des mécanismes système de synchronisation entre les parties de traitements effectuées sur les différentes machines. Les applications client / serveur sont, par définition, des traitements coopératifs.

Une application pourra pourtant être répartie mais sans traitement coopératif si des ULT primaires différentes sont exécutées sur des machines différentes (d'une façon transparente pour l'utilisateur); par exemple la procédure d'immatriculation nationale d'un nouvel assuré est réalisée sur le système national dédié alors que le reste de l'application fonctionne sur un système départemental

Données synchronisées

La question de synchronisation des données ne concerne que la répartition d'informations identiques (au niveau organisationnel) sur des machines logiques différentes. Ces données seront synchronisées si l'identité des valeurs est maintenue en temps réel sur les différentes machines logique. En règle générale, cette synchronisation nécessite des dispositifs systèmes sophistiqués (commit en deux phases).

Des données pourront toutefois être réparties sans être synchronisées. L'une des machines logiques sera la référence mise à jour en temps réel, les autres ne disposeront que de copies (clichés) dont la mise à jour pour synchronisation pourra être réalisée en différé (quotidiennement par exemple).

Client - serveur

Nous avons évoqué à plusieurs reprises l'architecture client - serveur comme champ d'application privilégié de la répartition. Etant donné l'importance prise par ce type d'architecture technologique dans le développement des systèmes d'information, nous avons souhaité consacrer un chapitre spécial à l'utilisation de la méthode Merise dans le cadre du client - serveur.

Aujourd'hui, toutes les approches client - serveur sont basées sur la distinction de trois composants : la présentation, les données et les traitements. La figure 12.15 illustre une correspondance entre ces trois composantes et la décomposition analytique que nous avons proposée pour l'ULT..

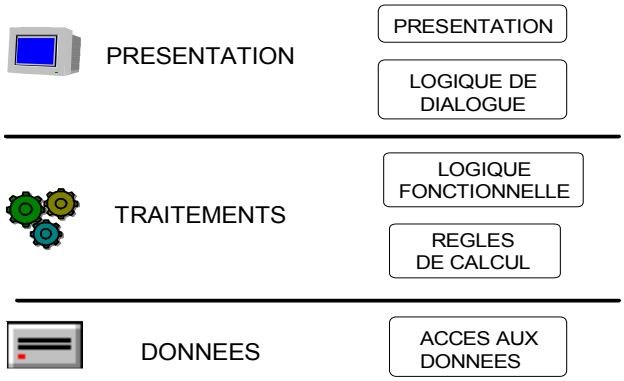


Figure 12.15 : Les composants d'une application client / serveur

13

Modélisation Logique des Données

Problématique de la modélisation logique des données

La méthode Merise propose une modélisation logique, puis physique des données. La modélisation logique des données est une représentation des données, issue de la modélisation conceptuelle puis organisationnelle des données. Elle est exprimée dans un formalisme général et compatible avec l'état de l'art technique, et tient compte des aspects coût/performance liés aux traitements.

La modélisation logique des données conduira aux opérations suivantes :

- Transformation du MOD, exprimé en formalisme entité-relation, en un MLD (modèle logique de données) exprimé dans un formalisme logique adapté au SGBD (voire système de gestion de fichiers) envisagé;
- Quantification en volume du modèle logique;
- Valorisation de l'activité générée par les modèles externes associés aux traitements (tâches du MOT);
- Optimisation générale.

Le modèle logique sera ensuite transformé et adapté en fonction des spécificités du langage de définition de données spécifique à l'outil (par exemple le SGBD) retenu, pour devenir modèle physique de données (voir figure 13.1).

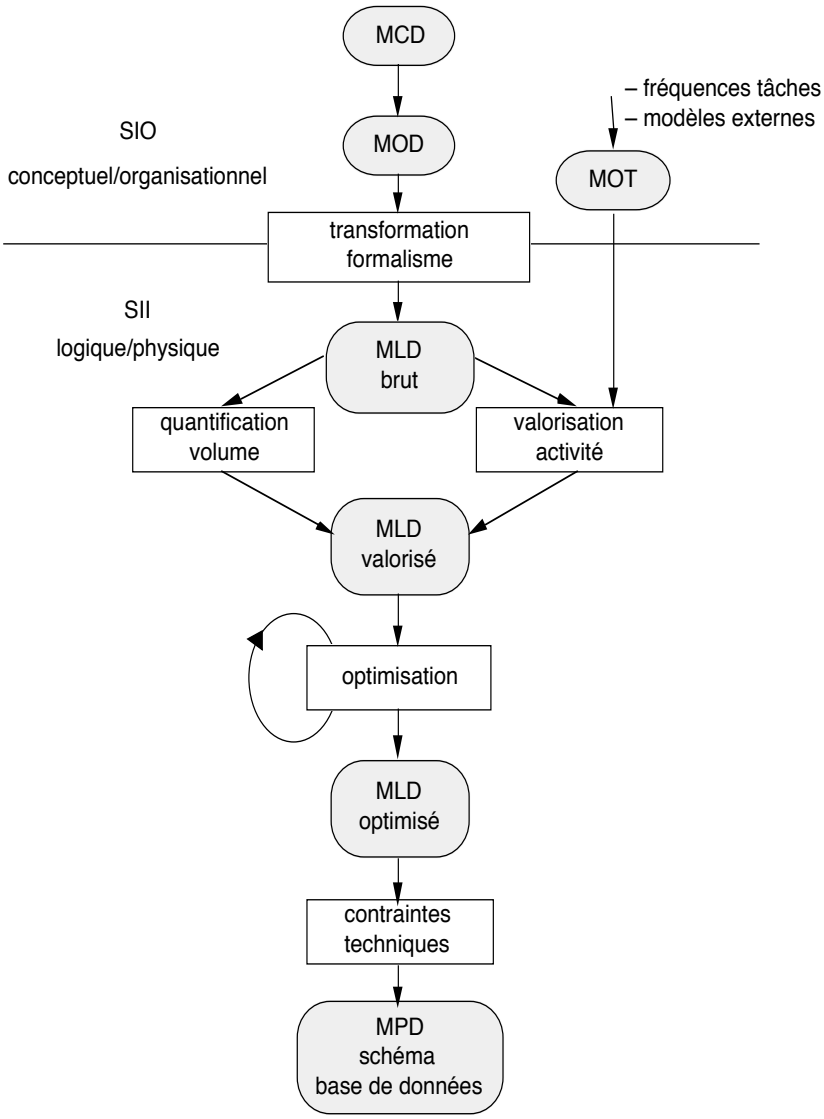


Figure 13.1: Le passage au modèle logique de données.

Notons que, si la frontière entre modèle conceptuel/organisationnel et modèle logique est nettement marquée, la frontière est plus floue entre modèle logique et modèle physique, à tel point que l'on nomme parfois modèle interne de données la réunion des modèles logique et physique.

Deux modèles (ou plutôt formalismes) théoriques de bases de données sont disponibles pour la représentation du modèle logique de données : le modèle relationnel et le modèle navigationnel (notamment Codasyl). A ces deux modèles sont associés, respectivement, les systèmes de gestion de bases de

données (SGBD) relationnels et navigationnels.

Les SGBD Relationnels se sont progressivement imposés au point de constituer, en cette fin des années 90, la quasi totalité de l'offre du marché (en particulier avec l'offre client/serveur). Aussi, nous ne traiterons dans cet ouvrage que du modèle et des SGBD relationnels, en mettant l'accent sur le passage de la modélisation conceptuelle/organisationnelle des données à la modélisation logique. Le chapitre suivant, le chapitre 14, traitera de l'optimisation tant au niveau logique qu'au niveau physique des modèles de données relationnels.

Modèle logique de données relationnel

Le modèle relationnel a été défini par E.F. Codd en 1970 [Codd 70]. Il a rapidement connu un engouement dans les milieux universitaires internationaux par la rigueur qu'il apportait dans la modélisation des données (théorie des ensembles, logique formelle), et l'ouverture qu'il offrait au travers des langages assertionnels. Actuellement, la plupart des systèmes de gestion de bases de données commercialisés sont de type relationnel.

Le modèle relationnel présente deux aspects fondamentaux : une algèbre permettant de manipuler des tables ou relations et ensuite une démarche de conception permettant de définir une collection de relations. Dans le cadre de cet ouvrage, nous ne présenterons de ce modèle que l'essentiel, nécessaire pour conduire l'informatisation d'un système d'information. De nombreux ouvrages ont été consacrés au modèle et aux systèmes relationnels, citons notamment [Date 85] [Delobel 82] [Gardarin 82] [Bouzeghoub, Jouve, Pucheral 90].

Concepts de base du modèle relationnel

Concepts structuraux

Le modèle relationnel s'appuie sur trois concepts structuraux de base : le domaine, la relation et l'attribut. Nous allons présenter simplement ce modèle ; pour une définition plus formelle, nous renvoyons le lecteur aux ouvrages précédemment indiqués.

La *relation*concept central du modèle, peut être définie grossièrement comme un tableau de données. Les colonnes de ce tableau sont appelés les *attributs* de la relation. Chaque attribut peut prendre des valeurs dans un *domaine*. Par exemple, le domaine des entiers E = {0, ± 1, ± 2...} ; le domaine des booléens D1 = {0, 1} ; des couleurs D2 = {vert, bleu, blanc, rouge} ; le domaine des caractères... Les lignes de ce tableau, occurrence de la relation, seront appelées *tuples* ou *n-uplets*.

Notons que le concept de relation utilisé ici n'a pas la même signification que celui utilisé dans le formalisme entité-relation. Aussi préférerons-nous appeler

ce concept *table*.

On définit aussi pour la table les notions suivantes :

- cardinalité (nombre de lignes ou tuples de la table) ;
- degré (n, nombre de colonnes ou d'attributs de la table).

Schéma d'une table

Le schéma d'une table permet de définir une table. Il est constitué du nom de la table suivi de la liste de ses attributs avec leurs domaines de valeurs ainsi que de l'ensemble des contraintes d'intégrité associées à la table (nous définirons plus loin ces contraintes).

Par exemple, une table Pièce décrivant des pièces aura comme schéma :

Pièce (n° : entier ; nom : car (10) ; couleur : couleur ; poids : réel ; ville : car (10)).

Extension d'une table

Étant donné une table, définie par son schéma, une extension de cette table sera un ensemble de lignes ou tuples (ou occurrences), définies par les valeurs prises par les attributs.

Par exemple, une extension de la table Pièce pourrait être :

PIÈCE	n°	nom	couleur	poids	ville
tuple 1	P1	écrou	rouge	14	Londres
tuple 2	P2	boulon	vert	17	Paris
tuple 3	P3	vis	bleu	12	Rome
tuple 4	P4	vis	rouge	14	Londres
tuple 5	P5	rivet	bleu	12	Paris
tuple 6	P6	clou	vert	10	Londres

Cardinalité de la table PIECE = 6 (6 tuples) ; degré de la table Pièce = 5 (5 attributs)

Notons que, pour une extension, l'ordre des tuples ou l'ordre des attributs ne sont pas significatifs.

Base de données relationnelle

On définira une base de données relationnelle comme un ensemble de tables. Le schéma de la base de données sera l'ensemble des schémas des tables la composant.

Prenons l'exemple de cette base de données relationnelle :

FOURNISSEUR	n°fournisseur	nom_four	ville
	F1	Dupont	Paris
	F2	Bergeron	Québec
	F3	Thomas	Genève

FOURNIT	n°fournisseur	n°pièce	délais
	F1	P1	20
	F1	P3	10
	F2	P1	20
	F2	P4	30

PIECE	n°pièce	nom_pièce	dépôt
	P1	table	Québec
	P2	fenêtre	Genève
	P3	chaise	Montréal
	P4	porte	Paris

Sans préciser les domaines de chacun des attributs, les schémas relationnels des différentes tables sont :

- Table FOURNISSEUR (n°fournisseur, nom_four, ville) ;
- Table PIECE (n°pièce, nom_pièce, dépôt) ;
- Table FOURNIT (n°fournisseur,n°pièce, délai).

Les contraintes d'intégrité

Une contrainte d'intégrité est une assertion qui doit être vérifiée par les valeurs d'attributs de tables constituant une base de données. Les deux principaux types de contraintes d'intégrité sont la contrainte d'unicité de valeur, qui nous permettra de définir la clé primaire d'une table, et la contrainte référentielle permettant de relier deux tables. Nous aurons l'occasion de revenir plus précisément sur ces contraintes d'intégrité et d'en évoquer d'autres types.

Contrainte d'unicité de valeur, clé primaire d'une table

Les valeurs prises par un attribut ou une composition d'attributs d'une table peuvent être déclarées uniques pour toute extension de cette table. Ce ou ces attributs permettent alors d'identifier de façon unique chaque tuple de la table. On parle alors de *clé primaire simple* (un seul attribut) ou *clé primaire composée* (plusieurs attributs).

Par exemple, dans la base relationnelle précédente, les clés primaires des tables Fournisseur, Fournit et Pièce sont les suivantes (attributs soulignés) :

- Table FOURNISSEUR (<u>n°fournisseur</u>, nom_four, ville) : n°fournisseur = clé primaire simple;

- Table PIECE (n°pièce, nom_pièce, dépôt) : n°pièce = clé primaire simple;
- Table FOURNIT (n°fournisseur, n°pièce, délai) : n°fournisseur = clé primaire composée.

Cette notion de clé primaire est équivalente à la notion d'identification définie dans le formalisme entité-relation.

Contrainte référentielle

Une contrainte référentielle est un lien sémantique défini entre deux tables. Ce lien est réalisé par une duplication de la clé primaire d'une table dans une autre table. Cette clé dupliquée est appelée *clé étrangère.*

Par exemple, pour la base relationnelle précédente, dans la table FOURNIT (n°fournisseur, n°pièce, délai), *n°fournisseur* et *n°pièce* sont chacune des clés étrangères provenant des tables FOURNISSEUR et PIECE.

Problèmes liés à la conception de schémas relationnels

Considérons par exemple une table P concernant des propriétaires de véhicules et les attributs de cette table.

- nom : nom de la personne propriétaire ;
- date : date d'acquisition du véhicule ;
- tél : dernier téléphone du propriétaire ;
- n° immat : numéro d'immatriculation du véhicule ;
- marque : marque du véhicule ;
- type : type du véhicule ;
- cv : puissance fiscale du véhicule ;
- coul : couleur du véhicule.

Soit une extension de la table P :

P.	nom	date	tél	n° immat	marque	type	cv	coul
tuple1	Durand	10/2/88	422775	540HB75	Renault	R25S	9	bleu
tuple 2	Dupont	8/10/88	665241	301UP75	Peugeot	405GR	7	vert
tuple 3	Pagnol	7/7/89	912458	741XD13	Citroën	AX11	4	blanc
tuple 4	Pagnol	21/4/90	912458	848HP13	Volvo	245	8	gris
tuple 5	Duval	15/8/90	425896	120DR75	Citroën	BX17	6	blanc
tuple 6	Martin	10/7/90	522684	956BV42	Renault	R25S	9	rouge
tuple 7				952KV13	Volvo	850	9	verte
tuple 8	Pascot		656458					

Cette table P pose, dans son utilisation, un certain nombre de problèmes liés à la redondance des données, ou liés à la nécessité d'avoir des attributs pour lesquels on accepte de ne pas avoir de valeurs (valeurs nulles).

Données redondantes

La table P fait apparaître une personne autant de fois que le nombre de voitures qu'elle possède. Aussi toutes les données caractérisant cette personne se retrouvent alors dans plusieurs tuples de la même table. Ce qui conduit à des risques d'incohérences ; par exemple si M. Pagnol change de numéro de téléphone, il faudra s'assurer que ce changement sera effectué sur chacun des tuples concernant Pagnol, soit pour les tuples 3 et 4.

Une autre redondance est liée à la correspondance type-marque, par exemple Renault-R25S. En effet, par définition un type donné de véhicule est associé à une seule marque, cette correspondance type-marque apparaît dans autant de tuples qu'il y a de propriétaires possédant ce type de véhicule, cas des tuples 1 et 6. Une autre redondance de même nature concerne la correspondance type-cv.

Les valeurs nulles

La table P concerne à la fois des personnes (propriétaires) et des véhicules. On pourrait, dans cette table, vouloir garder des tuples concernant des voitures sans propriétaire ou des propriétaires sans voiture. Pour les propriétaires sans voiture, les attributs date, n° immat, marque, type, cv, coul n'auront pas de valeur ; ils auront tous et simultanément la valeur nulle.

Conception de schémas relationnels

Dans les systèmes relationnels, redondances et valeurs nulles (principalement sur clés) sont à éviter, car elles introduisent des incohérences potentielles et compliquent l'exploitation, la manipulation des tables. La présence de redondances et de valeurs nulles dans une table est principalement liée au fait que la table ne concerne pas de vraies entités ou de vraies associations entre

entités représentant le réel.

Une première façon de constituer un ensemble de " bonnes " tables (canoniques), c'est-à-dire limitant la redondance et l'usage de valeurs nulles, est de partir d'une table " universelle " dont le schéma se composerait de la totalité des attributs, sur laquelle on applique un processus de normalisation proposé par Codd [Codd 71], mettant en œuvre des opérations de l'algèbre relationnelle (projection). On peut aussi partir d'un ensemble initial de tables quelconques.

Une autre façon de constituer un ensemble de bonnes tables est de les dériver d'un modèle conceptuel/organisationnel de données (MCD/MOD) exprimé en formalisme entité-relation, c'est l'approche privilégiée par la méthode Merise.

Avant d'aborder ces différentes démarches de conception de schémas relationnels, présentons quelques éléments d'algèbre relationnelle.

Éléments d'algèbre relationnelle

L'algèbre relationnelle proposée par Codd se présente comme un ensemble d'opérations formelles s'appliquant à une ou plusieurs tables pour donner une nouvelle table. Ces opérations permettent d'exprimer des manipulations ensemblistes sur les données de la base sans faire appel à un cheminement explicite. On distingue sept opérations : la sélection, la projection, l'union, l'intersection, la différence, la jointure et la division.

Notons que le langage normalisé SQL, proposé dans la plupart des SGBD relationnels, implémente ces opérations de l'algèbre relationnelle. Dans le cadre de cet ouvrage, nous ne traiterons pas du langage SQL. Le lecteur aura par exemple une présentation complète de ce langage dans [Date 89].

La sélection ou restriction

L'opération de sélection est une opération unaire qui consiste à sélectionner un ensemble de tuples d'une table, selon un critère de sélection pouvant porter sur un ou plusieurs attributs. Cette opération génère une autre table de même schéma que la table de départ.

Soit la table CLIENT (n°, nom, ville) composée des tuples suivants :

n°	nom	ville
121	BERTRAND	PARIS
256	PAGNOL	MARSEILLE
542	LANDRY	QUEBEC
652	DUPOND	PARIS

La sélection des clients où ville = PARIS est :

n°	nom	ville
121	BERTRAND	PARIS
652	DUPOND	PARIS

La projection

Cette opération unaire consiste à :

- ne retenir que certains attributs de la table, c'est-à-dire supprimer certaines colonnes ;
- éliminer les occurrences identiques, c'est-à-dire supprimer les lignes ayant le même ensemble de valeurs (tuples doubles).

On obtient ainsi une nouvelle table dans sa structure. Remarquons que l'opération de projection ne réduit le nombre d'occurrences que si un attribut de la clé primaire a été éliminée de la table projetée.

Soit la table : LIGNE_DE_COMMANDES (n°commande, n°article, date, quantité) composée des tuples suivants :

n°commande	n°article	date	quantité
10	121 A	5/5	10
10	253 Z	5/5	1
10	712 H	5/5	3
12	253 Z	5/5	5
13	712 H	6/5	2

La projection de LIGNE_DE_COMMANDES sur (n°article, date) est :

n° article	date
121 A	5/5
253 Z	5/5
712 H	5/5
712 H	6/5

La jointure

La jointure permet d'obtenir une nouvelle table par la composition de deux tables. Elle consiste à :

- faire le produit cartésien des deux tables, c'est-à-dire concaténer chacune des lignes de la première table avec chacune des lignes de la seconde ;
- effectuer une opération de sélection ou qualification, généralement égalité entre un attribut de la première table et un attribut de la seconde (appelés attributs de jointure) ;
- effectuer ou non une opération de projection pour réduire le schéma de la table résultante.

En d'autres termes, l'opération de jointure réalise une concaténation de tables

limitée à des occurrences de tables présentant des valeurs communes sur des attributs de jointure. Cette opération matérialise le lien entre plusieurs tables ou la fusion de plusieurs tables.

Si l'opérateur de jointure est généralement l'égalité, la jointure peut être étendue à des opérateurs logiques quelconques.

Les opérations de jointure peuvent s'effectuer sur tout attribut, sans préjuger de la pertinence sémantique du résultat obtenu. Toutefois, on démontrerait que seules les jointures en égalité construites sur les attributs clés primaires traduisent des relations (conceptuelles).

Notons que la jointure naturelle de deux tables S et R est une jointure telle que les attributs de jointure sont les attributs de R et de S qui ont mêmes noms, et qu'elle est suivie par une projection permettant de conserver un seul de ces attributs égaux de même nom. R et S peuvent être la même table, on a alors une jointure d'une table sur elle-même.

Soient les tables CLIENT et COMMANDE composées des tuples suivants :

CLIENT

n° client	nom
121	DUVAL
253	LEROY
260	LANDRY
293	SANCHEZ

COMMANDE

n° commande	n° client	date
10	121	5/5
11	260	5/5
12	121	5/5
15	253	6/5

La jointure CLIENT et COMMANDE est telle que n°client de la table CLIENT = n° client de la table COMMANDE :

n° client	nom	n° commande	date
121	DUVAL	10	5/5
253	LEROY	15	6/5
121	DUVAL	12	5/5
260	LANDRY	11	5/5

Les opérations ensemblistes : union, intersection et différence

En considérant les tables comme des ensembles de tuples, ces opérations binaires — union, intersection et différence — correspondent aux opérations habituelles de la théorie des ensembles. Ces opérations ne peuvent être appliquées que sur des tables de même schéma et donnent une nouvelle table de même schéma (voir la figure 13.2).

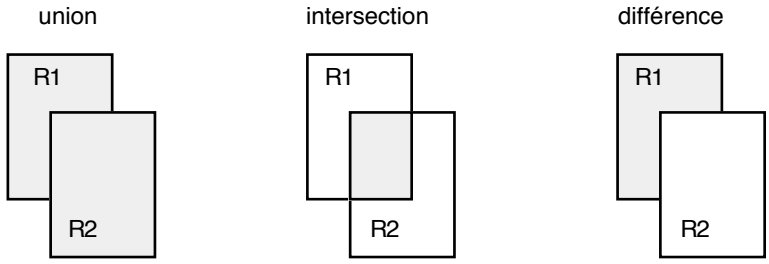


Figure 13.2 : Les opérations d'union, intersection et différence.

La division

L'opération de division est une opération binaire. Le quotient de la division d'une table R de schéma R (A1, A2… An) par la sous-table S de schéma S (Ap… An) est la table Q définie ainsi :

- le schéma de Q est constitué de tous les attributs de R n'appartenant pas à S, soit (A1, A2… Ap-1).
- les tuples de Q sont tels que, concaténés à chacun des tuples de S, ils donnent toujours un tuple de R.

Cette opération de l'algèbre relationnelle présente en fait un intérêt limité.

Par exemple, la division de la table PIECE par la table NATURE donne la table PIECE_NATURE.

PIECE	nom	couleur	poids	ville
tuple 1	clou	rouge	14	Londres
tuple 2	boulon	vert	17	Paris
tuple 3	vis	bleu	12	Rome
tuple 4	vis	rouge	14	Londres
tuple 5	rivet	bleu	12	Paris
tuple 6	clou	bleu	12	Rome

NATURE	nom
tuple 1	vis
tuple 2	clou

PIECE_NATURE	couleur	poids	ville
tuple 1	bleu	12	Rome
tuple 2	rouge	14	Londres

Les tuples de la table Pièce_Nature concaténés à chacun des tuples de la table

Nature donnent un tuple de la table Pièce.

Processus de normalisation

Comme nous l'avons évoqué précédemment, une première façon de constituer un ensemble de bonnes tables limitant le risque d'incohérences potentielles (éviter les redondances et les valeurs nulles) est de partir d'une table universelle dont le schéma se compose de la totalité des attributs, sur laquelle on applique un algorithme de normalisation, ou théorie de la normalisation.

La normalisation nécessite de disposer de plus de sémantique sur les données. Cette sémantique complémentaire s'exprimera au travers des dépendances fonctionnelles (DF) entre attributs. La normalisation se présente alors comme un processus de décomposition de cette table de départ en plusieurs tables par des projections définies judicieusement en fonction de ces dépendances fonctionnelles entre attributs. Ce processus de normalisation, ou théorie de la normalisation, peut être illustré de la façon suivante (cf. figure 13.3) :.

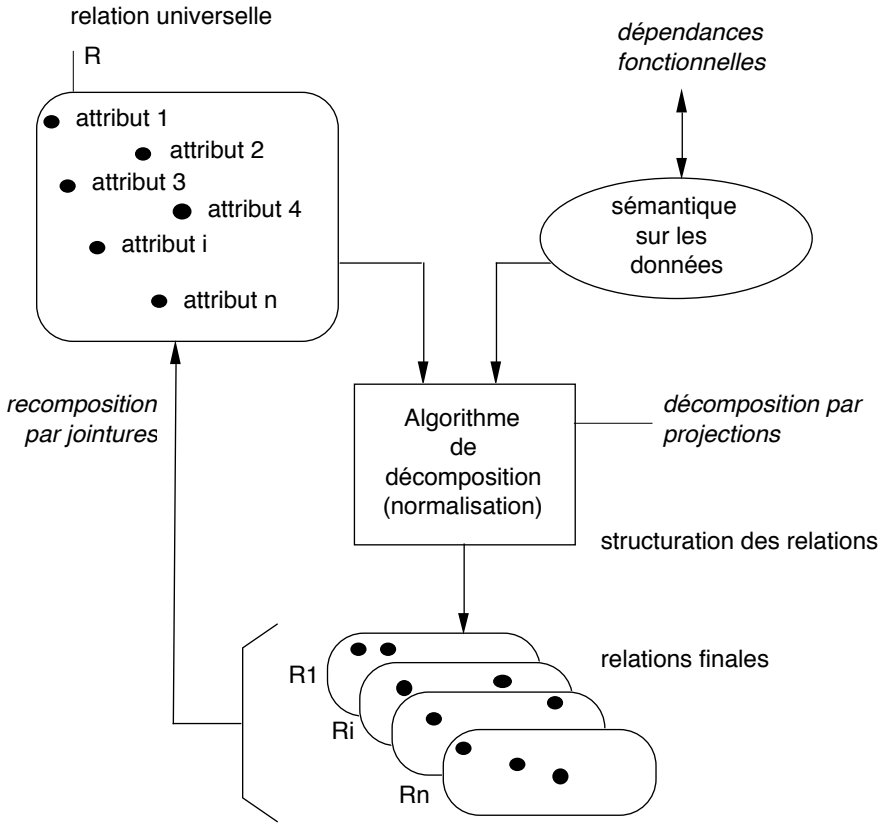


Figure 13.3 : Le processus de normalisation.

Notons que ce processus est absolument réversible et permet, à partir des tables normalisées, de retrouver les tables de départ, au moyen de jointures.

Codd a proposé trois formes normales, auxquelles ont ultérieurement été ajoutées d'autres formes normales comme les 4e, 5e formes normales ou la forme de Boyce-Codd. Ces autres formes normales peuvent être considérées comme des raffinements du modèle relationnel face à des problèmes très particuliers (optimisation).

Dans l'ensemble des tables pouvant être générées par décomposition, un sous-ensemble de tables est en 1re forme normale ; dans ce sous-ensemble, certaines tables sont en 2e forme normale, enfin, parmi ces dernières tables, certaines sont en 3e forme normale (voir figure 13.4).

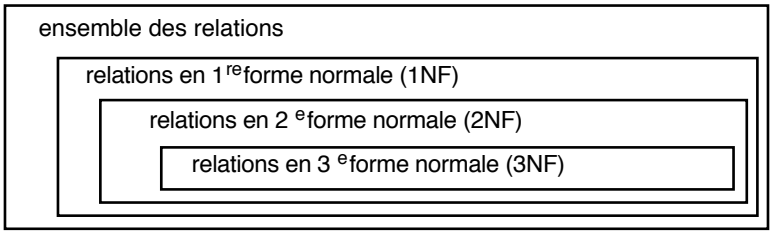


Figure 13.4 : Les 3 formes normales.

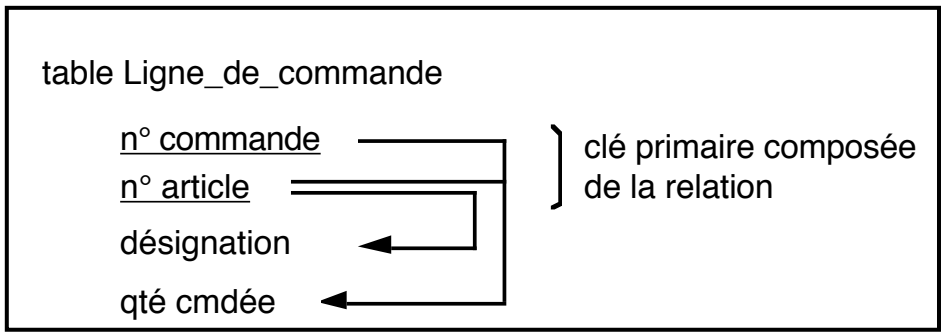
Notion de dépendance fonctionnelle

Un attribut (ou groupe d'attributs) B d'une table R est fonctionnellement dépendant d'un autre attribut (ou groupe d'attributs) A de R, *si, à tout instant, chaque valeur de A n'a qu'une valeur associée de B : on note A* **->** *B.*

Soit la table Ligne_de_commande (n° commande, n° article, désignation, qté cmdée) dans laquelle il s'agit de la désignation de l'article et de la quantité commandée d'articles dans la commande. On a les dépendances fonctionnelles suivantes :

- n° article -> désignation (à une valeur de n° article ne correspond qu'une valeur de désignation) ;
- (n° commande, n° article) -> qté cmdée (à un couple de valeurs n° commande et n° article, ne correspond qu'une valeur de qté cmdée).

On peut représenter ces dépendances fonctionnelles ainsi :



Propriétés des dépendances fonctionnelles

Les dépendances fonctionnelles sont :

- réflexivité (A **->** A) ;
- transitivité (A **->** B et B **->** C alors A **->** C).

Clé primaire d'une table

La clé primaire d'une table est un attribut (ou un ensemble d'attributs) tel que tous les autres attributs (non-clés) de la table sont en dépendance fonctionnelle avec la clé primaire.

Exemples :

Soit la Table Client (n° client, nom, adresse) ; on a les dépendances fonctionnelles :

- n° client -> nom ;
- n° client -> adresse.

n° client est clé primaire de la table Client.

Soit la Table Ligne (n° commande, n° article, désignation, qté cmdée) ; on a les dépendances fonctionnelles suivantes :

- n° commande, n° article -> désignation ;
- n° commande, n° article -> qté cmdée.

(n° commande, n° article) est clé primaire composée de la table Ligne.

Il faut d'autre part vérifier qu'aucun sous-ensemble des attributs composant la clé primaire ne pourrait également être une clé primaire de la table.

Première forme normale (1NF)

Cette normalisation s'applique sur des tables quelconques. La 1re forme normale a pour objet d'éliminer les groupes répétitifs dans une table. La démarche est la suivante :

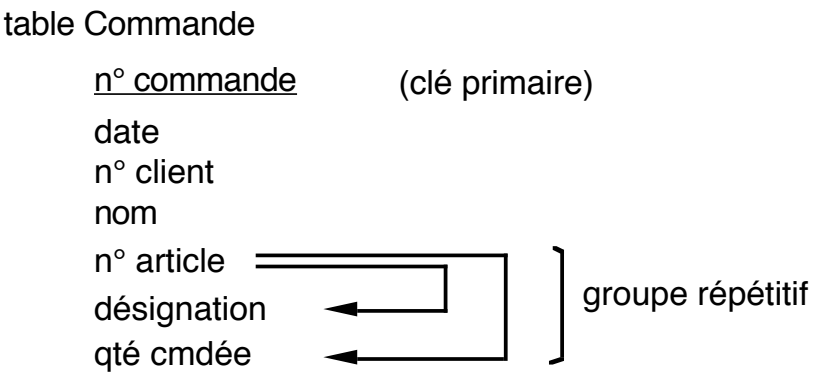
- Sortir le groupe répétitif de la table initiale.

- Transformer le groupe répétitif en table, rajouter dans la clé de cette nouvelle table la clé primaire de la table initiale.

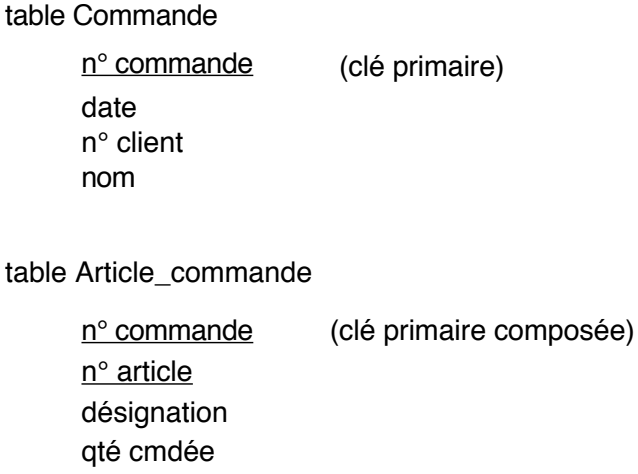
Ce processus de mise en première forme normale est récursif dans le cas où la table initiale comprend plusieurs niveaux de répétitivité.

On remarque que cette 1re forme normale s'apparente à la règle de vérification (ou non-répétitivité) utilisée dans le formalisme entité-relation.

Soit la table Commande (n° commande, date, n° client, nom, n° article, désignation, qté cmdée). L'attribut nom est le nom du client et désignation est la désignation de l'article. Cette table présente également un groupe répétitif :



Tables en première forme normale obtenues après élimination des groupements répétitifs :



Deuxième forme normale (2NF)

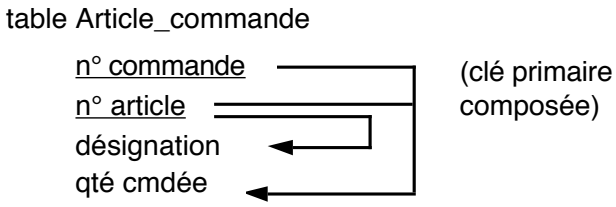
Cette normalisation exige que la table soit déjà en 1re forme normale. Elle ne concerne que les tables à clé primaire composée (composée de plusieurs attributs).

La règle impose que les attributs non-clé primaire dépendent de la totalité de la clé primaire. Tout attribut qui ne dépendrait que d'une partie de la clé primaire doit être exclu de la table. Le processus est le suivant :

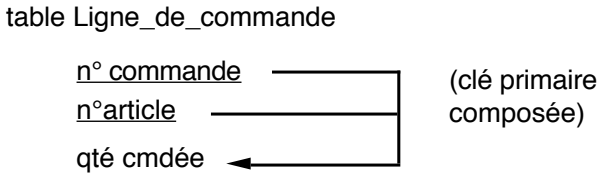
- Regrouper dans une table les attributs dépendant de la totalité de la clé, et conserver cette clé pour cette table ;
- Regrouper dans une autre table les attributs dépendant d'une partie de la clé, et faire de cette partie de clé la clé primaire de la nouvelle table.

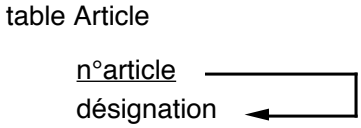
On remarque que cette deuxième forme normale s'apparente à la règle de normalisation d'une relation utilisée dans le formalisme entité-relation.

Dans notre exemple, la table Article_commande n'est pas en deuxième forme normale car l'attribut non-clé désignation ne dépend pas totalement de la clé primaire composée :



Le passage en deuxième forme normale nous conduira à remplacer cette table Article par les tables en deuxième forme normale suivantes :





Troisième forme normale (3NF)

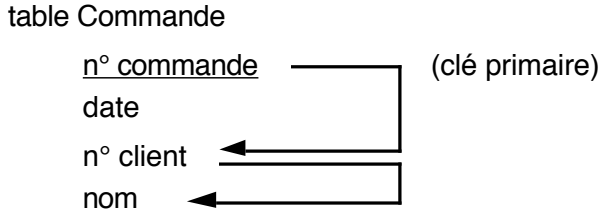
La mise en troisième forme normale ne s'applique que sur des tables déjà en deuxième forme normale. La règle a pour objet l'élimination des dépendances transitives au sein d'une table. La démarche est la suivante :

- Conserver dans la table initiale les attributs dépendant directement de la clé.
- Regrouper dans une table les attributs dépendant transitivement ;

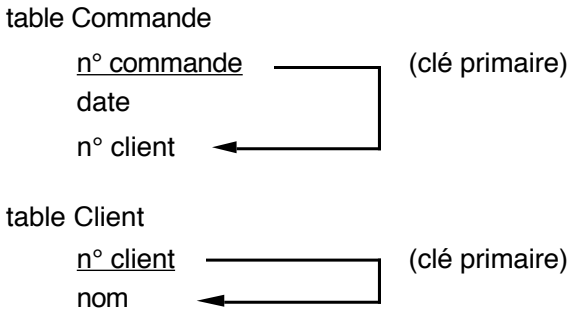
l'attribut de transition reste dupliqué dans la table initiale, et devient la clé primaire de la nouvelle table.

Notons que Codd et de nombreux spécialistes ont démontré rigoureusement qu'un modèle de données en troisième forme normale était une forme " canonique " sur un ensemble de données, et qu'il minimisait ainsi la redondance de la future base de données.

Dans notre exemple, la table Commande précédente n'est pas en troisième forme normale car l'attribut non-clé nom dépend de la clé par transitivité :



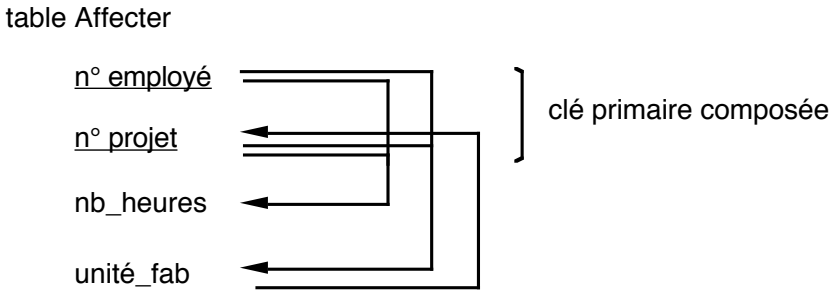
Le passage en troisième forme normale nous conduira à remplacer cette table Commande par les tables suivantes :



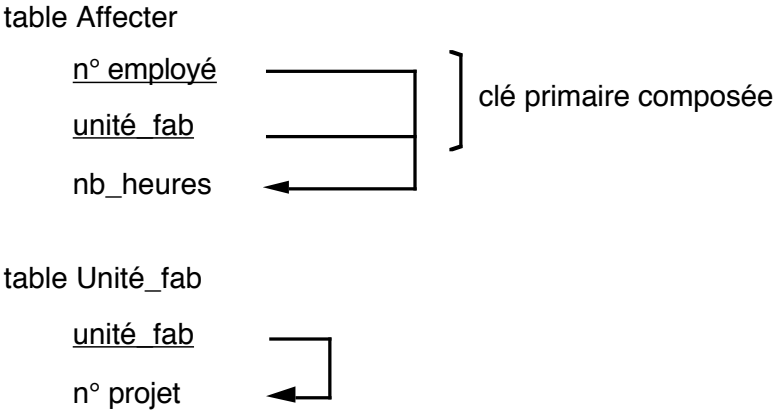
Forme normale de Boyce-Codd (BCNF)

La mise en forme normale de Boyce-Codd permet d'éviter des redondances dues à l'existence de dépendances fonctionnelles autres que celles de la clé vers des attributs non clés.

Soit un exemple dans lequel un employé est affecté (un certain nombre d'heures) à un certain nombre de projets effectués dans un certain nombre d'unités de fabrication (une unité de fabrication ne traite qu'un projet donné). Voici la table Affecter associée permettant de prendre en compte cette affectation :



Le passage en forme de Boyce-Codd nous conduira à remplacer cette table par les tables suivantes :



La décomposition en forme de Boyce-Codd est sans perte. On notera que la dépendance fonctionnelle n° employé -> unité_fab a disparu mais qu'elle peut être recomposée par jointure des deux nouvelles tables sur l'attribut unité_fab.

Notion de vue relationnelle

La plupart des SGBD relationnels permettent la définition de vues. Une vue est une table virtuelle pouvant être composée à partir d'une ou plusieurs tables. Le contenu et le schéma d'une vue sont déduits des tables comme étant le résultat d'une requête d'interrogation mettant en œuvre des opérations d'algèbre relationnelle. Le langage normalisé SQL permet de définir de telles vues.

Formalisation graphique du modèle relationnel

Les représentations des tables en extension par des tableaux de tuples, ou en intention par les schémas, sont intéressantes, mais limitées pour expliciter les liens sémantiques entre les tables. Une représentation graphique aidera le concepteur à spécifier ces liens sémantiques et les contraintes d'intégrité référentielle associées. Plusieurs représentations graphiques du modèle relationnel ont été proposées [Senko 76, Everest 77, Ridjanovic 83...]. Dans le cadre de cet ouvrage, nous en avons retenu une que nous allons présenter.

La table, ses attributs et sa clé primaire

Soit une table Employé de schéma : Employé (matricule, nom , âge, adresse), matricule étant défini ici comme clé primaire. La figure 13.5 montre la représentation graphique associée à cette table.

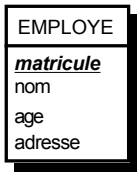


Figure 13.5 : Une table avec sa clé primaire.

Notons que, dans cette représentation graphique, le ou les attributs composant la clé primaire de la table sont soulignés.

Contraintes d'intégrité référentielle

Rappelons qu'une telle contrainte reliera deux tables A et B de telle sorte qu'à une valeur de la clé primaire de B puissent correspondre plusieurs valeurs de clés primaires de A, et qu'à une valeur de clé primaire de A ne corresponde au plus qu'une valeur de clé primaire de B. On parlera de lien relationnel un vers plusieurs.

Cette liaison conduit à une duplication dans la table A de la clé primaire de la table B, qui devient ainsi dans la table A une *clé étrangère*. La figure 13.6 montre la représentation graphique associée.

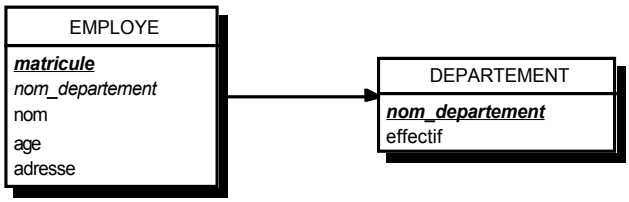


Figure 13.6 : Une contrainte d'intégrité référentielle.

On notera que, dans cette représentation, l'attribut *nom_departement* de la table EMPLOYE est clé étrangère et pointe vers l'attribut *nom_departement* , clé primaire de la table DEPARTEMENT. La lecture de ce modèle graphique conduit ainsi aux schémas des tables suivantes :

- Table DEPARTEMENT (nom_departement , effectif, ...);
- Table EMPLOYE (matricule., nom_departement (clé étrangère), nom, âge, adresse).

Clé primaire composée référentielle

Soit une table TACHE, reliée à une table PROJET par une contrainte d'intégrité référentielle. La table TACHE a une clé primaire composée faisant référence à la table PROJET: ainsi l'attribut *n°projet* de la table TACHE est clé étrangère et participe aussi à la clé primaire (voir figure 13.7).

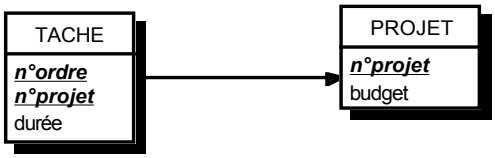


Figure 13.7 : Une clé primaire composée référentielle.

Ce qui donne par simple lecture les schémas suivants :

- Table TACHE (n° projet, n° ordre (clé étrangère), durée);
- Table PROJET (n° projet, budget).

Règles de transformation du formalisme entité-relation en formalisme relationnel

Une seconde façon de constituer une collection de bonnes tables est de la dériver d'un MOD (modèle organisationnel de données) ou d'un MCD (modèle conceptuel de données) exprimé en formalisme entité-relation (voir figure 13.8).

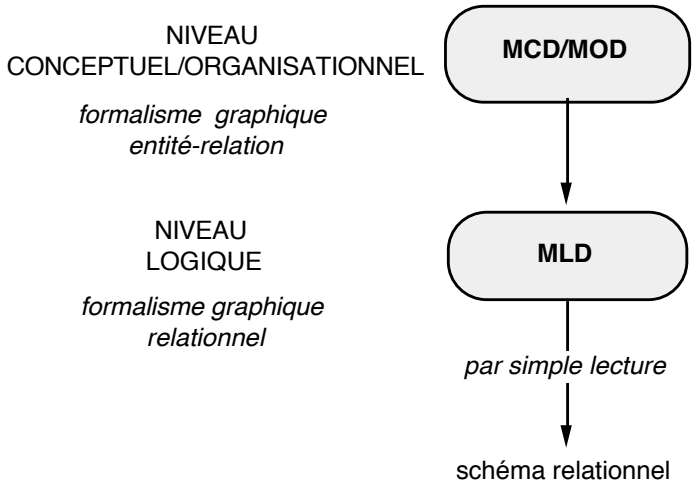


Figure 13.8 : Dérivation d'un schéma relationnel à partir d'un MCD/MOD.

Cette transformation est entièrement algorithmique mais n'est pas totalement réversible. Le modèle ainsi obtenu est obligatoirement en deuxième forme normale. Il n'est pas nécessairement en troisième forme, le choix des entités n'étant pas guidé sur la redondance minimale.

En toute rigueur, il conviendrait de vérifier si les tables issues des entités sont

en troisième forme normale ; ce qui est fréquemment le cas.

Notons que le développement de la retro-conception (reverse engineering) a rendu nécessaire l'élaboration de règles permettant de construire un MCD/MOD exprimé en entité-relation à partir d'un MLD relationnel. Les règles utilisées reconstituent les modélisations en entité-relation les plus probables mais n'offrent pas une certitude de retro-conception.

Entité

Tout entité type est transformée en table. Ses propriétés deviennent des attributs de la table. L'identifiant devient la clé primaire de la table (voir figure 13.9).

Remarquons que l'entité ne comportant que l'identifiant comme propriété présente un cas particulier. Il devient provisoirement une table avec sa clé primaire comme seul attribut. Il est fort probable que les optimisations ultérieures fassent disparaître cette table.

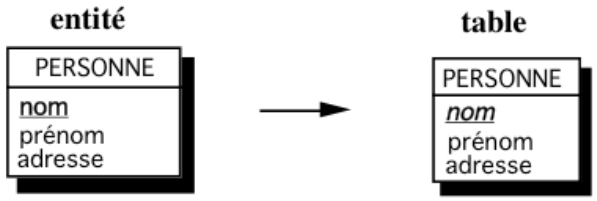
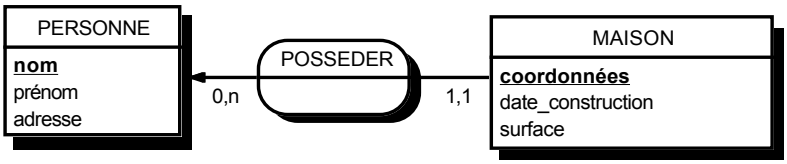


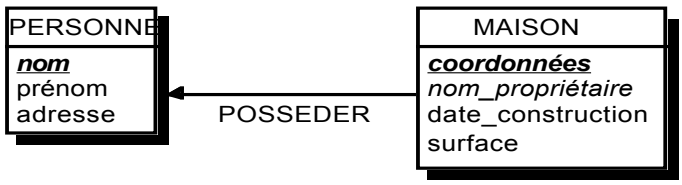
Figure 13.9 : Transformée d'une entité

Relation binaire (0,n)-(1,1) ou (1,n)-(1,1)

On duplique la clé de la table issue de l'entité à cardinalité (0,n) ou (1,n) dans la table issue de l'entité à cardinalité (1,1) où elle devient une clé étrangère. On procède éventuellement à un changement d'appellation de l'attribut dupliqué qui conserve cependant son domaine de valeurs, comme l'illustre la clé étrangère *nom_propriétaire* de la table MAISON de la figure 13.10.



Entité-Relation



Relationnel dérivé

Figure 13.10 : Transformé d'une relation binaire (,n)-(1,1).*

Schémas relationnels associés à la figure 13.10 :

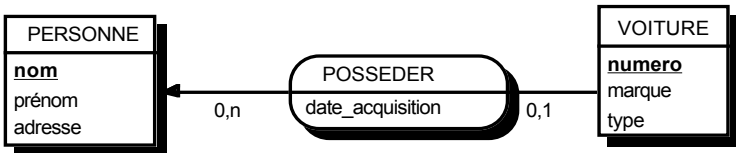
• Table PERSONNE (nom, prénom, adresse),

• Table MAISON (coordonnées, *nom_propriétaire*, date_construction, surface).

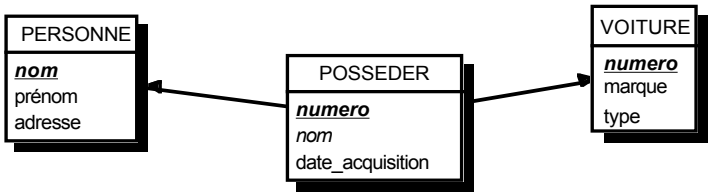
Relation binaire (0,n)-(0,1) ou (1,n)-(0,1)

Deux solutions sont possibles.

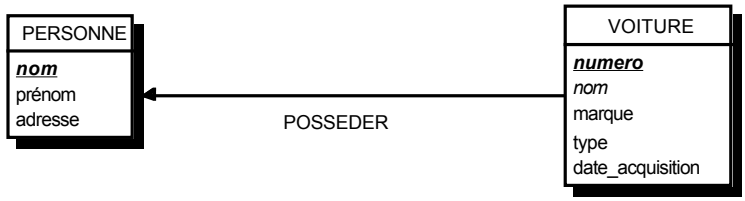
La première solution (solution 1) consiste à créer une table avec comme clé primaire l'identifiant de l'entité à cardinalité (0,1) ; l'identifiant de l'autre entité devenant clé étrangère de cette table. Les éventuelles propriétés de la relation deviennent aussi attributs de la table issue de la relation.



Entité-Relation



Relationnel dérivé : solution 1



Relationnel dérivé : solution 2

Figure 13.11 : Transformé d'une relation binaire (,n)-(0,1).*

Schémas relationnels associés à la figure 13.11 :

Solution 1 :

• Table PERSONNE (nom, prénom, adresse) ;

• Table POSSEDER (*numéro, nom*, date_acquisition) ;

• Table VOITURE (numéro, marque, type).

Solution 2 :

• Table PERSONNE (nom, prénom, adresse) ;

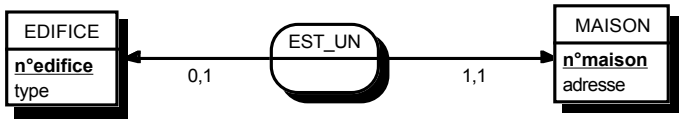
• Table VOITURE (numéro, *nom*, marque, type, date_acquisition).

Dans la seconde solution (solution 2), on duplique comme clé étrangère la clé de la table issue de l'entité à cardinalité (0,n) ou (1,n) dans la table issue de l'entité à cardinalité (0,1). On procède éventuellement à un changement d'appellation de l'attribut dupliqué qui conserve cependant son domaine de valeurs. Les éventuelles propriétés de la relation (conceptuelle) deviennent des attributs de la table issue de l'entité à cardinalité (0,1) (voir figure 13.11).

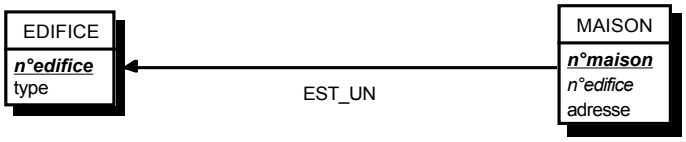
Remarque : Dans la seconde solution, la cardinalité (0,1) posera le problème, pouvant être difficile à gérer selon le SGBD adopté, d'accepter des valeurs nulles sur la clé étrangère.

Relation binaire (0,1)-(1,1)

C'est en fait une particularisation des cas précédemment traités, correspondant souvent à exprimer des sous-types. On duplique la clé de la table issue de l'entité à cardinalité (0,1) dans la table issue de l'entité à cardinalité (1,1) (voir figure 13.12).



Entité-Relation



Relationnel dérivé

Figure 13.12 : Transformé d'une relation binaire (0,1)-(1,1).

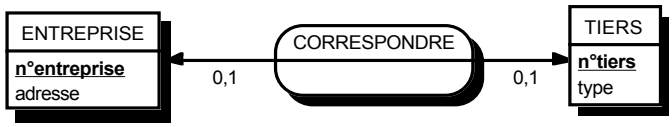
Schémas relationnels associés à la figure 13.12 :

• Table EDIFICE (n°édifice, type) ;

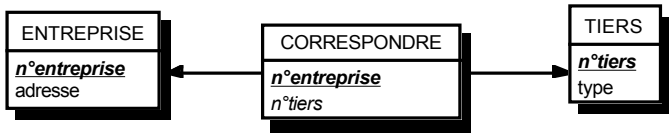
• Table MAISON (n°maison, n°*édifice*, adresse).

Relation binaire (0,1)-(0,1)

C'est en fait également une particularisation des cas précédemment traités. Deux types de solutions sont possibles. Le premier consiste à créer une table avec comme clé les identifiants des entités concernées par la relation considérée. Les éventuelles propriétés de cette relation deviennent des attributs de cette table issue de la relation. La clé de cette table de lien peut être la clé primaire de l'une des deux tables issues des entités (voir figure 13.13).



Entité-Relation



Relationnel dérivé : solution 1

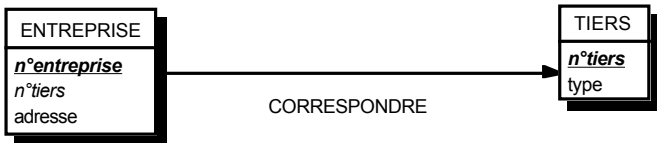
Figure 13.13 : Transformé d'une relation binaire (0,1)-(0,1).

L'autre type de solutions consiste à dupliquer de la clé d'une table issue d'une entité dans l'autre table issue de l'autre entité. Les éventuelles propriétés de cette relation deviennent des attributs de la table dans laquelle a migré l'identifiant externe (voir figure 13.14).

Dans ces deux dernières solutions, la cardinalité (0,1) posera le problème, pouvant être difficile à gérer selon le SGBD adopté, d'accepter des valeurs nulles sur l'attribut migrant. Ce problème des valeurs nulles pourra, dans

certains cas, fixer le sens de migration (par exemple, la taille des clés). On peut considérer cette seconde solution comme associée à un choix d'optimisation, comme nous le verrons dans le chapitre qui y est consacré.

Relationnel dérivé : solution 3



Relationnel dérivé : solution 4

Figure 13.14 : Autres transformés d'une relation binaire (0,1)-(0,1).

Schémas relationnels associés :

Solution 1 :

• Table ENTREPRISE (n°entreprise, adresse) ;

• Table TIERS (n°tiers, type) ;

• Table CORRESPOND (n°entreprise, n° tiers).

Solution 2 :

• Table ENTREPRISE (n°entreprise, adresse) ;

• Table TIERS (n° tiers, type) ;

• Table CORRESPOND (*n°entreprise, n° tiers*) .

Solution 3 :

• Table ENTREPRISE (n°entreprise, adresse) ;

• Table TIERS (n° tiers, *n°entreprise*, adresse).

Solution 4 :

• Table ENTREPRISE (n° entreprise, n° tiers, adresse) ;

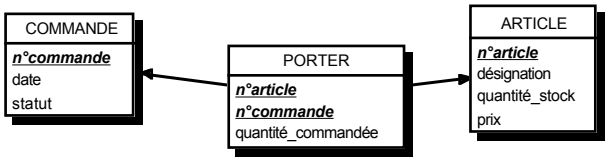
• Table TIERS (n° tiers, type) ;

Relation binaire (0,n)-(0,n) ou (1,n)-(1,n) ou (1,n)-(0,n)

La solution consiste à créer une table ayant comme clé, une clé composée des identifiants des deux entités. Les éventuelles propriétés de cette relation

deviennent des attributs de la table issue de la relation (voir figure 13.15).

Entité-Relation



Relationnel dérivé

Figure 13.15 : Transformé d'une relation binaire (,n)-(*,n).*

Schémas relationnels associés à la figure 13.15 :

• Table COMMANDE (n°commande, date, statut) ;

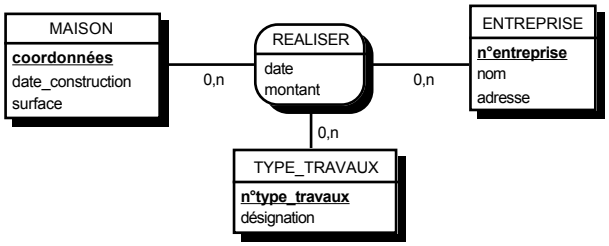
• Table PORTER (*n°article, n°commande*, quantité_commandée) ;

• Table ARTICLE (n°article, désignation, quantité_stock, prix).

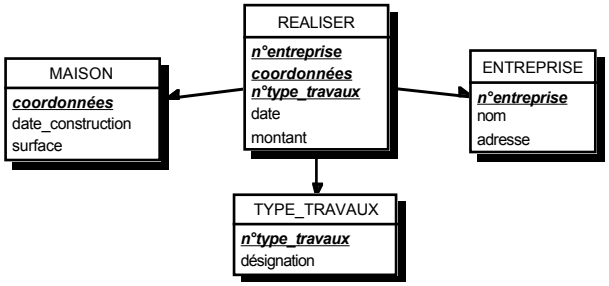
Relation ternaire ou supérieure

La transformée d'une relation ternaire ou supérieure quelles que soient les cardinalités, consiste à créer une table ayant comme clé une clé composée des identifiants des diverses entités reliées par la relation considérée. Les éventuelles propriétés de cette relation deviennent des attributs de la table issue de la relation. Il s'agit en fait de la généralisation de la règle précédente (voir figure 13.16).

Remarque : Rappelons qu'une relation n-aire munie de cardinalité (1,1) aura été au préalable décomposée, comme nous l'avons déjà indiqué dans la partie précédente.



Entité-Relation



Relationnel dérivé

Figure 13.16 : Transformé d'une relation ternaire ou supérieure.

Schémas relationnels associés à la figure 13.16 :

• Table MAISON (coordonnées, date_construction, surface) ;

• Table TYPE_TRAVAUX (n°type_travaux, désignation) ;

• Table RÉALISER (n°entreprise, coordonnées, date, montant) ;

• Table ENTREPRISE (n°entreprise, nom, adresse).

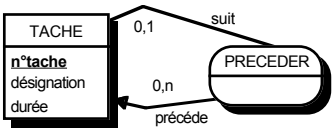
Relation réflexive (sur la même entité type)

Relation binaire réflexive (0,n)-(0,1)

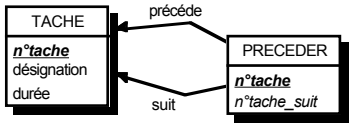
C'est en fait une particularisation des cas précédemment traités. Dans une première solution, la relation conduit à la création d'une table dans laquelle la clé primaire de la table issue de l'entité se retrouve à la fois comme clé et comme simple attribut. On procède à un changement d'appellation de ces attributs dupliqués qui conservent cependant leur domaine de valeurs. Les éventuelles propriétés de cette relation deviennent des attributs de cette table associée.

Dans une seconde solution, on duplique la clé de la table issue de l'entité dans la table issue de l'entité à cardinalité (0,1). On procède à un changement d'appellation de l'attribut dupliqué qui conserve cependant son domaine de valeurs (n° tâche_précédente). Les éventuelles propriétés de cette relation

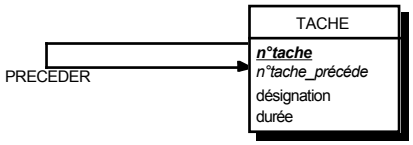
deviennent des attributs de la table issue de l'entité (voir figure 13.17).



Entité-Relation



Relationnel dérivé : solution 1



Relationnel dérivé : solution 1

Figure 13.17 : Transformé d'une relation réflexive (,n)-(0,1).*

Schémas relationnels associés à la figure 13.17 :

Solution 1

• Table TACHE (n°tâche, désignation, durée) ;

• Table PRÉCÉDER (n°tâche, n°tâche_suit).

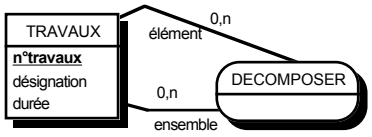
Solution 2

• Table TACHE (n°tâche, n°tâche_précéde, désignation, durée).

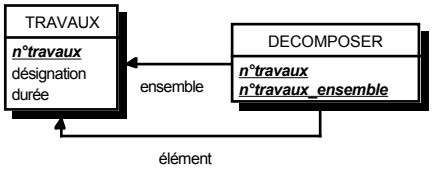
Dans cette seconde solution, la cardinalité (0,1) posera le problème d'accepter des valeurs nulles sur l'attribut migrant. On peut considérer cette seconde solution comme associée à un choix d'optimisation, comme nous le verrons dans le chapitre qui y est consacré.

Relation binaire réflexive (*,n)-(*,n)

La solution consiste à créer une table de lien ayant comme clé une clé composée de deux fois l'identifiant de l'entité. Les clés étrangères seront qualifiées par rapport aux rôles des pattes de la relation. Les éventuelles propriétés de cette relation deviennent des attributs de cette table issue de la relation (voir figure 13.18).



Entité-Relation



Relationnel dérivé

Figure 13.18 : Transformé d'une relation réflexive (,n)-(*,n).*

Schémas relationnels associés à la figure 13.18 :

- Table TRAVAUX (n°travaux, désignation, durée) ;
- Table DÉCOMPOSER (*n°travaux, n°travaux_ensemble*).

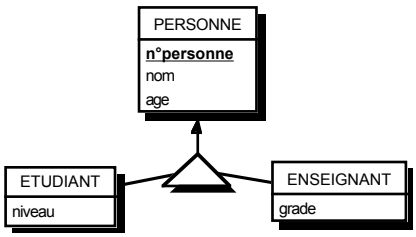
Prise en compte des sous-types du MCD/MOD

Spécialisation

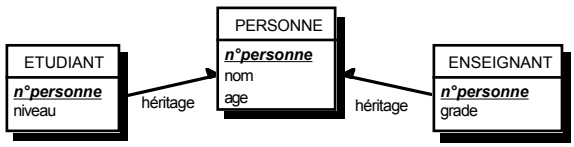
Plusieurs solutions possibles.

Solution 1 :

On exprime les sous-types par des tables spécifiques, correspondant en fait à des relations (0,1)-(1,1). Il y a ainsi migration de l'identifiant du sur-type dans les sous-types (figure 13.19).



Entité-Relation



Relationnel dérivé

Figure 13.19a : Transformé d'une spécialisation (sol.1)

Schémas relationnels associés à la solution 1 (figure 13.19) :

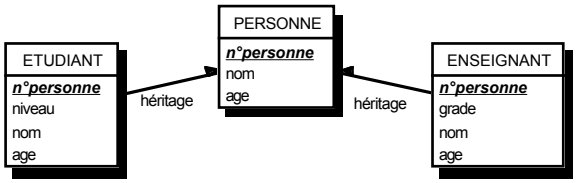
• Table PERSONNE (n°personne, nom, age) ;

• Table ETUDIANT (n°personne, niveau) ;

• Table ENSEIGNANT (n°personne, grade).

Solution 2 :

Comme dans la solution précédente, on exprime les sous-types par des tables spécifiques. Il y a duplication des attributs du sur-type dans les sous-types associés, dont la mise à jour simultanée peut être réalisée à travers un mécanisme automatique implémentant l'héritage, par exemple par triggers (cf. figure 13.19.b).



Relationnel dérivé

Figure 13.19b : Transformé d'une spécialisation (sol.2)

Solution 3 :

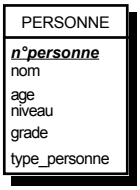
Dans cette solution, on duplique la totalité du contenu du sur-type dans les sous-types associés et on supprime le sur-type. Cette solution n'est pas conseillée dans le cas où il existe, dans le modèle conceptuel (entité-relation) des relations portant sur le sur-type (cf. figure 13.19.c).

Relationnel dérivé

Figure 13.19c : Transformé d'une spécialisation (sol.3)

Solution 4 :

On transfère la totalité des propriétés des sous-types dans la table correspondant au sur-type. On exprime ensuite les sous-types par des vues relationnelles d'une table PERSONNE globale. Dans l'exemple, les sous-types ETUDIANT et ENSEIGNANT sont exprimés par des vues relationnelles de la table PERSONNE rassemblant l'ensemble des propriétés spécifiques aux sous-types (cf. figure 13.19.d).



Relationnel dérivé

Figure 13.19d : Transformé d'une spécialisation (sol.4)

Pour faciliter l'expression de ces vues, on peut introduire un nouvel attribut "type_personne" dont le domaine est {étudiant, enseignant} :

• Table PERSONNE (n°personne, nom, age, **type_personne**, niveau, grade...),

avec comme définition des vues :

• Vue PERSONNE_ETUDIANT : vue de PERSONNE (sélection + projection),

• Vue PERSONNE_ENSEIGNANT : vue de PERSONNE (sélection + projection) ;

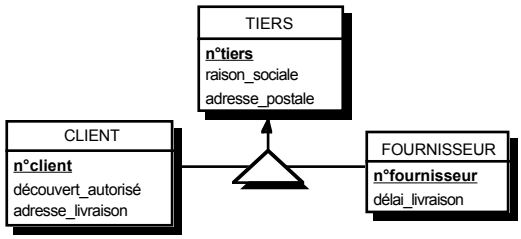
soit en SQL :

- **CREATE VIEW** PERSONNE_ETUDIANT **AS** SELECT n° personne, nom, age, niveau... FROM PERSONNE WHERE PERSONNE.type_personne = 'étudiant';
- **CREATE VIEW** PERSONNE_ENSEIGNANT **AS** SELECT n° personne, nom, age, grade... FROM PERSONNE WHERE

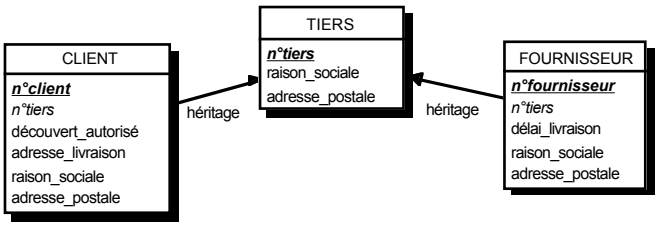
PERSONNE.type_personne = ‘enseignant’;

Généralisation

Dans le cas d’une généralisation, les sous-types ont leurs propres identifiants. Ainsi seules les transformations des solutions précédentes 1 et 2 sont possibles, comme l'illustre la figure 13.20.



Entité-Relation



Relationnel dérivé

Figure 13.20 : Transformé d’une généralisation.

Schémas relationnels associés à la figure 13.20 :

• Table TIERS (n°tiers, raison_sociale, adresse_postale) ;

• Table CLIENT (n°client, *n°tiers* , découvert_autorisé, adresse_livraison, raison_sociale, adresse_postale) ;

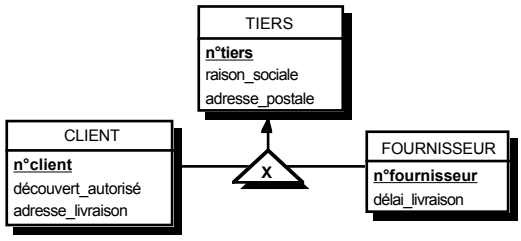
• Table FOURNISSEUR (n°fournisseur, *n°tiers* , délai_livraison, raison_sociale, adresse_postale).

Remarque : Nous sommes strictement dans le cas d’une relation (0,1)-(1,1).

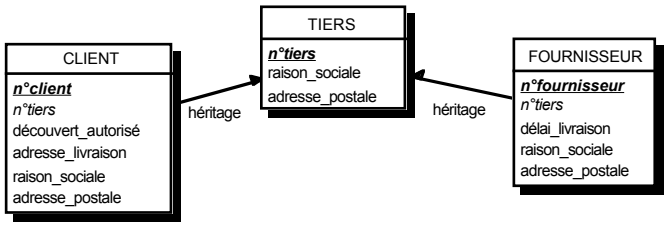
Dans la table CLIENTS et FOURNISSEURS l’attribut n°tiers sera déclaré comme une clé candidate ou clé alternative (cf. “ Valeurs uniques ” du paragraphe sur la prise en compte des contraintes de ce chapitre) et sera déclaré dans le schéma avec une clause UNIQUE.

Exclusion sur spécialisation et généralisation

Une exclusion sur spécialisation ou généralisation s'exprime dans un MCD/MOD par un "X" indiquant une contrainte d'exclusion dans le triangle du sous-typage. Cette contrainte sera prise en compte au niveau logique par l'écriture de triggers spécifiques comme l'illustre la figure 13.21 relative à une généralisation.



Entité-Relation



Relationnel dérivé

Figure 13.21 : Exclusion sur généralisation.

Schémas relationnels associés à la figure 13.21 :

• Table TIERS (n° tiers, n° client, n° fournisseur, raison sociale, adresse,...) ;
• Table CLIENT (n° client, n° tiers, découvert autorisé, adresse de livraison) ;
• Table FOURNISSEUR (n° fournisseur, n° tiers, délai de livraison).

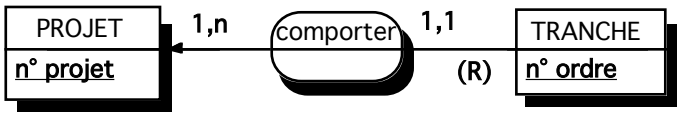
La contrainte d'exclusion implémentée par triggers spécifiques.

Identifiant relatif

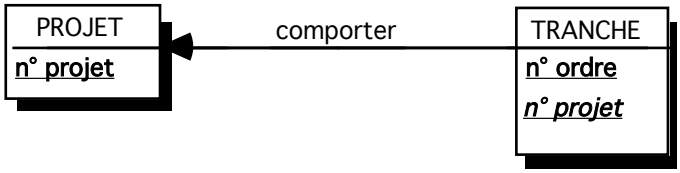
Nous avons vu au chapitre 7 que toute entité type devait être dotée d'un identifiant et nous avons évoqué la difficulté d'inventer de telles propriétés. Certaines entités types ont par ailleurs une existence totalement dépendante d'autres entités types. On a alors recours à un identifiant relatif. L'identification relative est notée par (R) sur la patte servant de relativité. Dans l'exemple de la figure 13.22, l'identifiant de la tranche est n° d'ordre relatif à projet.

La transformée d'une identification relative revient à définir pour la table fille issue de l'entité identifiée relativement, une clé primaire composée constituée de la clé primaire transformée de l'identifiant de cette entité et de la clé

étrangère implémentant la relation servant l'identification relative, comme l'illustre la figure 13.22.



Entité-Relation



Relationnel dérivé

Figure 13.22 : Transformé d'un identifiant relatif

Schémas relationnels associés à la figure 13.22 :

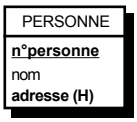
• Table PROJET (n°projet, ...) ;

• Table TRANCHE (n°ordre, *n°projet,* ...).

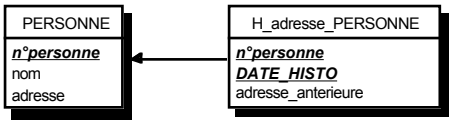
Prise en compte de l'historisation

Historisation de propriété

La conservation des valeurs antérieures (historisation) ne s'applique qu'à certaines propriétés d'une entité ou d'une relation. Graphiquement, on indique alors le caractère historisable par un (H) au niveau de la propriété, comme nous l'avons déjà vu dans le chapitre 7. La transformation au niveau logique relationnel s'effectue de la façon suivante (voir figure 13.22) :



Entité-Relation



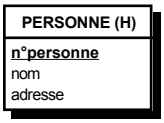
Relationnel dérivé

Figure 13.22 : Cas de l'historisation d'une propriété

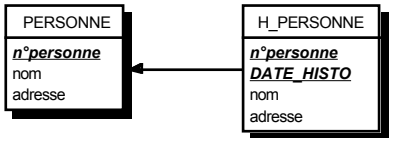
Notons que pour toutes les historisations, l'attribut de datation de l'historisation *(DATE_HISTO)* rentrant dans la clé primaire de la table d'historisation (ici H_adresse_PERSONNE) est déduit de la valeur de datation (jour, semaine, mois, ...).

Historisation d'entité

Pour toute modification de valeur de l'une des propriétés d'une entité, on historise l'ensemble des valeurs des propriétés de l'entité. Graphiquement, on indique alors le caractère historisable par un (H) au niveau de l'entité, comme déjà vu dans le chapitre 7. La transformation au niveau logique relationnel s'effectue ainsi (voir figure 13.23) :



Entité-Relation

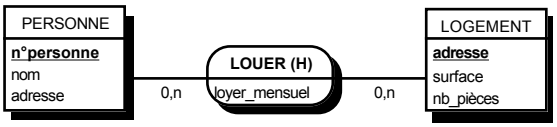


Relationnel dérivé

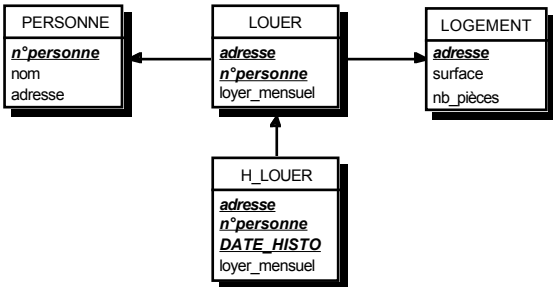
Figure 13.23 : Cas de l'historisation d'une entité

Historisation de relation

Pour toute modification de valeur de l'une des propriétés d'une relation, on historise l'ensemble des valeurs des propriétés de la relation ainsi que son identification. Graphiquement, on indique alors le caractère historisable par un (H) au niveau de la relation (voir chapitre 7). La transformation au niveau logique relationnel s'effectue de la façon suivante (voir figure 13.24) :



Entité-Relation



Relationnel dérivé

Figure 13.24 : Cas de l'historisation d'une relation

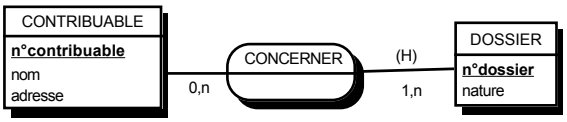
Historisation de patte de relation

Comme nous l'avons déjà indiqué dans le chapitre 7, l'historisation d'une patte d'une relation se présente lorsque, lors de la modification d'une patte de relation, on souhaite historiser la patte précédente, c'est à dire conserver la valeur de l'identifiant de l'occurrence précédente de l'entité dans la patte. Graphiquement, on indique alors le caractère historisable par un (H) au niveau de la patte concernée de la relation.

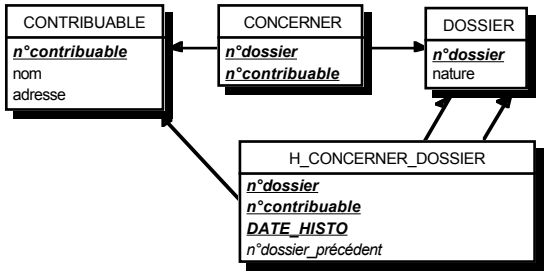
Cas d'une relation binaire (*,n)-(*n), ternaire ou supérieure

La figure 13.25, déjà présenté au chapitre 7 illustre un tel cas. Il s'agit ici d'historiser les changements de dossiers d'un contribuable.

La relation concernée par l'historisation est transformée en table (ici la table CONCERNER). On crée une nouvelle table d'historisation (la table H_CONCERNER_DOSSIER) avec une clé primaire composée des clés primaires des tables DOSSIER et CONTRIBUABLE et d'un attribut Date _Histo. Cette table d'historisation possède aussi un attribut clé étrangère (ici *n°dossier_précédent*) qui réfère au dossier antérieur concerné de la table DOSSIER assurant ainsi l'historisation.



Entité-Relation



Relationnel dérivé

Figure 13.25 : Cas de l'historisation d'une patte de relation (,n)-(0,n)*

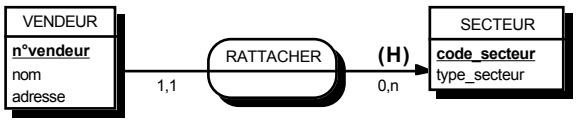
Cas d'une relation binaire (*,n)-(1,1)

Deux cas se présentent selon que l'on souhaite, pour une telle relation, historiser la patte avec la cardinalité (*,n) ou la patte avec la cardinalité (1,1).

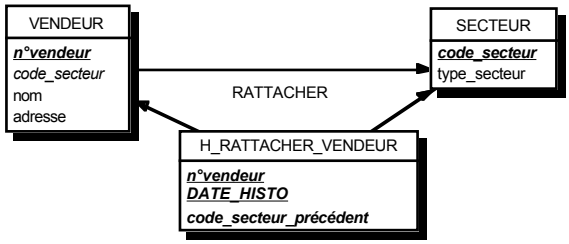
Historisation de la patte (*, n)

Un tel cas est illustré par la figure 13.26, concernant l'historisation des changements de secteurs d'un vendeur.

La relation est transformée en clé étrangère dans la table issue de l'entité coté (1,1). Pour historiser la patte (*,n) de la relation (ici la relation RATTACHER), on crée une nouvelle table d'historisation (la table H_RATTACHER_VENDEUR dans l'exemple) avec une clé primaire composée de la clé primaire de la table issue de l'entité coté cardinalité (1,1) et d'un attribut *Date_Histo* . Cette table d'historisation possède aussi un attribut clé étrangère qui réfère à la table issue de l'entité qui intervenait dans la patte à historiser (ici la table SECTEUR). Cet attribut accueille la valeur antérieure réalisant l'historisation.



Entité-Relation



Relationnel dérivé

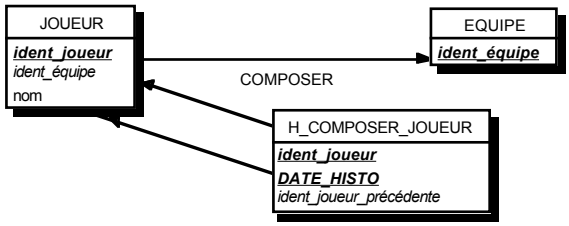
Figure 13.26 : Historisation de la patte (,n) d'une relation binaire (*,n)-(1,1)*

Historisation de la patte (1,1)

Un tel cas est illustré par la figure 13.27, concernant l'historisation de la composition d'une équipe.

La relation (ici la relation COMPOSER) est transformée en clé étrangère dans la table issue de l'entité coté (1,1) (table JOUEUR). Pour historiser la patte (1,1) de cette relation, on crée une nouvelle table d'historisation (ici la table H_COMPOSER_JOUEUR) avec une clé primaire composée de la clé primaire de la table issue de l'entité coté cardinalité (1,1) et d'un attribut *Date_Histo* . Cette table d'historisation possède aussi un attribut clé étrangère (ici *ident_joueur_précédente*) qui réfère encore à cette table issue de l'entité coté (1,1). Cet attribut accueille la valeur antérieure permettant ainsi l'historisation.

Entité-Relation



Relationnel dérivé

Figure 13.27 : Historisation de la patte (1,1) d'une relation binaire (,n)-(1,1)*

Triggers d'historisation automatique

Dans la transformation Entité-Relation (MCD/MOD) vers un modèle logique de données relationnel (MLD), il est possible de générer des triggers qui, selon les cas d'historisation, permettent, lors de la modification des propriétés (devenues des attributs non clés) ou de la collection d'une relation (devenu des clés étrangères), de faire automatiquement la création de l'occurrence de la table d'historisation correspondante avec éventuellement le contrôle par rapport à la datation (simple modification ou historisation) et à la profondeur (suppression des valeurs les plus anciennes).

Intégrité dans les bases de données relationnelles

Intégrité et contraintes d'intégrité

L'intégrité d'une base de données consiste à assurer une constante concordance des données avec le monde réel que la base de données modélise. Le maintien de l'intégrité de la base de données est une fonction essentielle des SGBD relationnels. Les aspects liés à l'intégrité d'une base de données sont multiples :

- l'intégrité sémantique,
- le contrôle de concurrence,
- la protection des données,
- la sécurité des données.

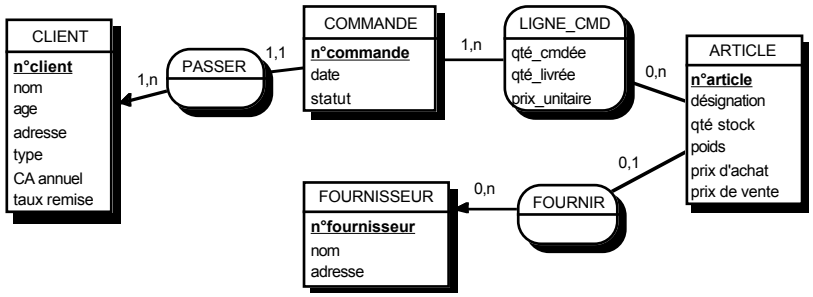
Dans le cadre de cet ouvrage, nous ne nous intéresserons qu'à l'intégrité sémantique. Nous renvoyons le lecteur à des ouvrages spécialisés sur les bases de données relationnelles pour les autres aspects.

L'*intégrité sémantique* concerne la qualité de l'information stockée dans la base de données. Elle consiste à s'assurer que les données stockées sont cohérentes par rapport à leur signification. L'intégrité sémantique sera assurée lorsqu'il y aura respect de règles de cohérence ou contraintes d'intégrité sémantique.

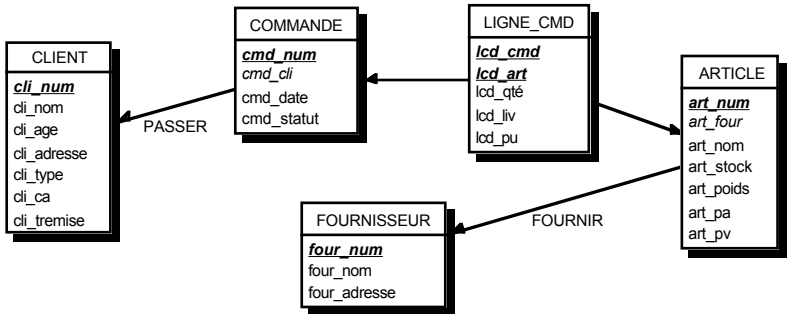
Une *contrainte d'intégrité* sera définie comme une assertion qui doit être vérifiée par des données à des instants déterminés. Une base de données sera dite sémantiquement intègre si l'ensemble des contraintes d'intégrité (implicites et explicites) est respecté par les données de la base

Plusieurs typologie de contraintes d'intégrité sont possibles, selon que l'on s'intéresse à leur signification, leur complexité ou aux types d'opérations qui les déclenchent [Bouzeghoub et al. 91]. Nous distinguerons pour notre part quatre grands types de contraintes d'intégrités prises en compte dans les bases de données relationnelles : les contraintes syntaxiques, les contraintes structurelles, les contraintes temporelles et enfin les contraintes générales.

Pour illustrer ces types de contraintes ainsi que leur prise en compte dans le relationnel, nous prendrons un exemple, celui présenté par le modèle Entité-Relation (MCD/MOD) et le MLD relationnel dérivé de la figure 13.28. Notons qu'un changement de dénomination et préfixage des attributs a été effectué afin de pouvoir faciliter certaines expressions du langage SQL.



Entité-Relation



Relationnel dérivé

Figure 13.29 : Modèle entité-relation et relationnel de l'exemple

Schémas relationnels associés :

- Table CLIENT (**cli_num**, cli_nom, cli_age, cli_adresse, cli_type, cli_ca, cli_tremise);
- Table ARTICLE (**art_num**, art_nom, art_four, art_stock, art_poids, art_pa, art_pv);
- Table COMMANDE (**cmd_num**, *cmd_cli*, cmd_date, cmd_statut);
- Table LIGNE_CMD (***lcd_cmd, lcd_art***, lcd_qte, lcd_liv, lcd_pu);
- Table FOURNISSEUR (**four_num**, four_nom, four_adresse);

Les attributs clés primaires sont représentés ici en gras et les attributs clés étrangères en italique.

Nous pouvons associer à cette base de données relationnelle, les contraintes d'intégrité suivantes :

Des contraintes syntaxiques concernant le format des données (domaine de définition) :

- Définition d'un domaine de typage (exemple, dans la table CLIENT, l'attribut *cli_num* est de type CHAR(8)).
- Définition d'un domaine en plage de valeurs (exemple : dans la table CLIENT, l'attribut *cli_age* ne peut prendre que des valeurs comprises entre 0 et 120).
- Définition d'un domaine de variation défini en extension (exemple, dans la table CLIENT, *cli_type* peut prendre les valeurs Particulier, Administration, Entreprise ; dans la table Commande, *cmd_statut* peut prendre les valeurs Prévisionnelle, Validée, Urgente).
- Définition d'un domaine par défaut (exemple, dans la table COMMANDE, *cmd_statut* prend par défaut la valeur 'Prévisionnelle').
- Acceptation de la valeur nulle (exemple : dans la table CLIENT, *cli_age* accepte la valeur nulle — si le client est une personne morale, c'est à dire si l'attribut *cli_type* prend la valeur 'Administration' ou la valeur 'Entreprise').

Des contraintes structurelles concernant les règles de construction de :

- Clé de table, unicité de valeur pour une donnée (exemple, dans la table CLIENT, *cli_num* est clé primaire).
- Dépendances fonctionnelles (intrarelation, intratuple) (exemple, dans la table CLIENT, on a les dépendances *cli_num* -> *cli_nom*...).
- Contraintes référentielles (interrelation, intrarelation) (exemple, l'attribut *cmd_cli* de la table COMMANDE doit prendre comme valeur des valeurs de l'attribut *cli_num* de la table CLIENT...).

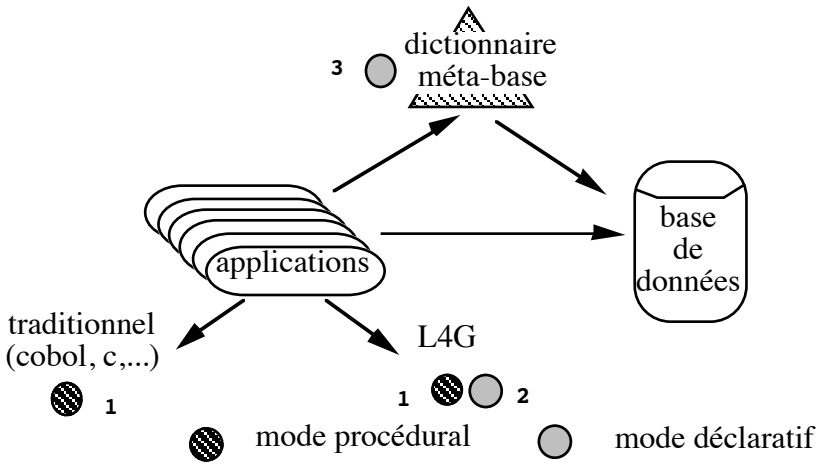
Des contraintes temporelles précisant des règles d'évolution des valeurs de certains attributs en fonction du temps ; par exemple, seulement vraies en fin de mois, seule une évolution croissante de la valeur d'une donnée est autorisé.

Dans notre exemple, considérons la table LIGNE_CMD. On doit avoir à tout instant pour tout tuple de cette table : *lcd_qte* >= *lcd_liv* signifiant que la quantité commandée est toujours supérieure ou égale à la quantité livrée.

Des contraintes d'intégrité générales spécifiant toujours la cohérence des données entre elles mais n'entrant pas dans les catégories précédemment citées.

Dans notre exemple, on supposera que le prix unitaire de l'article *cmd_pu* de la table LIGNE_CMD ne peut être remisé à plus de 25% du prix de vente de l'article (prix catalogue) *art_pv* défini dans la table ARTICLE.

Dans le développement d'applications autour d'une base de données relationnelle, les contraintes d'intégrité peuvent être prises en compte de différentes façons soit de façon procédurale, soit de façon déclarative et ceci à différents niveaux soit au niveau du dictionnaire (ou schéma) ou au niveau de l'application (développé en langages de troisième (L3G) ou de quatrième génération (L4G)) comme l'illustre la figure 13.29.



1 - procédural au niveau des applications (L3G, L4G)

2 - déclaratif au niveau des applications (L4G)

3 - déclaratif au niveau du dictionnaire

Figure 13.29 : Prise en compte des contraintes d'intégrité.

Traitement procédural

On peut ainsi prendre en compte une contrainte d'intégrité de façon procédurale, c'est-à-dire en écrivant une procédure qui veillera au respect de celle-ci. Cette procédure sera définie dans les différents modules applications utilisant la base de données, que ces modules soient écrits en langages de troisième (L3G) ou de quatrième génération (L4G).

Traitement déclaratif au niveau des applications

Une autre façon de traiter ces contraintes est de les déclarer par des assertions. Ces assertions peuvent être définies, stockées, tout d'abord au niveau des applications, principalement au niveau des écrans, formes programmées en langages de troisième ou quatrième génération. Un intérêt par rapport à la solution précédente est qu'un écran, une forme, peut être utilisé par plusieurs modules applications ; la contrainte est alors partagée, ce qui limite les problèmes d'incohérences.

Traitement déclaratif au niveau du dictionnaire

Une dernière solution est de déclarer ces contraintes au niveau du dictionnaire,

du schéma de la base de données relationnelle. Cette dernière solution est la plus puissante et ceci pour plusieurs raison :

- Tout d'abord, la spécification des contraintes d'intégrité au niveau du dictionnaire permet d'alléger le travail de programmation. En effet, les contraintes ne sont définies qu'une seule fois et de façon centralisée. Le SGBD vérifie automatiquement chaque contrainte à chaque manipulation de données. Il envoie un éventuel message d'erreur à l'application que celle-ci peut alors gérer par un traitement spécifique. Plusieurs applications partageant les mêmes données, partageront aussi les mêmes contraintes d'intégrité sur ces données.
- Ensuite, dans une architecture client-serveur, les contraintes spécifiées au niveau du dictionnaire de données conduiront à réduire le niveau du trafic sur le réseau, la prise en compte des contraintes étant réalisée au niveau du serveur et non au niveau de l'application s'exécutant sur le poste client.

Prendre en compte les contraintes d'intégrité au niveau du dictionnaire nécessite de disposer d'une part d'un langage déclaratif de spécification de ces contraintes et d'autre part, que les SGBD soient effectivement capables de prendre en compte, de façon opérationnelle (performance), ces contraintes ainsi déclarées. Jusqu'à présent, très peu de contraintes d'intégrité pouvaient être " assimilées " de façon déclarative par les SGBD relationnels. La norme SQL-92 (ou SQL-2) [Marée & Ledant 94] comme nous le verrons plus loin, constitue une avancée significative dans la spécification déclarative des contraintes.

Vérification des contraintes d'intégrité

La plupart des contraintes d'intégrités déclarées sont vérifiées immédiatement et automatiquement. Ainsi, dès qu'une requête de mise à jour (exprimée en langage SQL) est exécutée par une application, les contraintes d'intégrité sont vérifiées par le système et si une contrainte n'est pas vérifiée, la requête est défaite.

La vérification de certaines contraintes peut être différée, par exemple à la fin d'une transaction. Rappelons qu'une transaction est un ensemble de requête SQL qui fait passer la base de données d'un état cohérent à un autre état cohérent. Entre ces états cohérents, il existe des états transitoires de la base de données qui sont incohérents. La vérification des contraintes est faite à la fin de la transaction en cours, pour vérifier que l'état dans laquelle celle-ci à laissé la base de donnée est effectivement cohérent.

Dans SQL-92 il est possible de spécifier pour chaque contrainte déclarée son mode de prise en compte : immédiat ou différé à la fin de la transaction en cours (NOT DEFERABLE ou DEFERRABLE). Ce mode peut être fixé initialement et changer ensuite (INITIALLY DEFERRED\IMMEDIATE).

Certaines contraintes d'intégrité, assez complexes, traitée de façon procédurale ne sont pas déclenchées automatiquement par le SGBD mais par le programmeur. Celui-ci peut programmer dans une application le déclenchement de la contrainte dès la survenance d'un événement donné. Le programmeur peut alors avoir recours à un mécanisme puissant disponible dans de nombreux SGBD, le "trigger". Ainsi, pour chaque contrainte complexe à vérifier sur apparition d'un événement, on créera un trigger, consistant en une procédure compilée, cataloguée dans le dictionnaire, qui s'exécutera automatiquement chaque fois que l'événement déclenchant associé se produira. La norme SQL-92 ne normalise pas l'expression de ces triggers, SQL-3, la prochaine norme dont la sortie est prévue pour 96, devrait la normaliser.

Enfin, notons qu'avant de prendre en compte une nouvelle contrainte d'intégrité, il faut s'assurer de sa conformité avec :

- *le schéma de la base* ; c'est-à-dire que tout attribut ou table cité dans la contrainte d'intégrité appartient bien au schéma ;
- *les données de la base* ; il faut s'assurer que toutes les données déjà existantes dans la base respectent la nouvelle contrainte d'intégrité ;
- *les autres contraintes d'intégrité* ; il faut s'assurer qu'il n'y a pas de contradiction, de redondances...

Représentation et expression des contraintes d'intégrité

Il est possible d'exprimer les contraintes d'intégrité en *logique des prédicats du premier ordre*. Très tôt des langages plus adaptés ont aussi été proposés comme le langage LDC [Simon 84 - INRIA-Sabrina], le langage RAISIN [Stonebraker 85]. Comme nous l'avons déjà évoqué, l'expression des contraintes d'intégrité commence à être normalisée dans le langage SQL-2. Nous allons voir dans ce qui suit comment des contraintes, précisées de façon implicite ou explicite dans un modèle conceptuel de données en formalisme Entité-Relation, peuvent être traitées dans le modèle relationnel.

Nous illustrerons ceci au travers de l'exemple défini plus haut, auquel peut être associé le schéma relationnel suivant en SQL-92 (figure 12-30 - afin de mettre en valeur les termes du langage SQL, ils sont écris en majuscules, alors que les désignations des tables et des attributs sont en minuscules) :

```
CREATE TABLE Client
    (cli_num      CHAR(8) PRIMARY KEY
                  CONSTRAINT PK_CLIENT,
    cli_nom       CHAR(25) UNIQUE,
                  CONSTRAINT UNIQUE_CLIENT,
    cli_age       INTEGER CHECK (cli_age > 0 AND
       cli_age < 120),
                  CONSTRAINT AGE_CLIENT,
    cli_adresse   VARCHAR(80),
    cli_type      VARCHAR(16) DEFAULT 'Particulier',
    cli_ca        INTEGER DEFAULT 0
    cli_tremise   INTEGER NOT NULL);
    CHECK (cli_type IN ('Particulier', 'Administration,
    'Entreprise'))
                  CONSTRAINT TYPE_CLIENT);

CREATE TABLE Article
    (art_num      CHAR(8) DEFAULT 0 PRIMARY KEY
                  CONSTRAINT PK_ARTICLE,
    art_nom       VARCHAR(25) NOT NULL,
    art_four      CHAR(8) REFERENCES Fournisseur
                  ON UPDATE CASCADE
                  ON DELETE SET NULL
                  CONSTRAINT FK_FOURNISSEUR
    art_stock     INTEGER DEFAULT 0 CHECK
    (art_stock > 0)
                  CONSTRAINT STOCK_ARTICLE,
    art_poids     NUMERIC (8,1),
    art_pa        INTEGER NOT NULL,
    art_pv        INTEGER NOT NULL,
    CHECK (art_pv > art_pa / 0.8)
                  CONSTRAINT PVPA_ARTICLE);

CREATE TABLE Commande
    (cmd_num      CHAR(8) PRIMARY KEY
                  CONSTRAINT PK_COMMANDE,
    cmd_cli       CHAR(8) NOT NULL REFERENCES
    Client
                  ON UPDATE CASCADE
                  ON DELETE NO ACTION
                  CONSTRAINT FK_CLIENT
    cmd_date      DATE NOT NULL,
    cmd_statut    VARCHAR(16)          DEFAULT
    'Prévisionelle'
    CHECK (cmd_statut IN ('Prévisionnelle', 'Validée',
    'Urgente')
                  CONSTRAINT
    STATUT_COMMANDE);
```

```
CREATE TABLE Ligne_cmd
    (lcd_art      CHAR(8) NOT NULL,
    lcd_cmd       INTEGER NOT NULL,
    lcd_qte       INTEGER NOT NULL,
    lcd_liv       INTEGER DEFAULT 0,
    lcd_pu        INTEGER NOT NULL,
    PRIMARY KEY (lcd_cmd, lcd_art)
      CONSTRAINT PK_LIGNCDE
    FOREIGN KEY (lcd_cmd) REFERENCES Commande
                  ON UPDATE CASCADE
                  ON DELETE CASCADE
                  CONSTRAINT FK_COMMANDE
    FOREIGN KEY (lcd_art) REFERENCES Article
                  ON UPDATE CASCADE
                  ON DELETE NO ACTION
                  CONSTRAINT FK_ARTICLE
    CHECK (lcd_pu >=0.75 *
                  (SELECT art_pv FROM Article
                        WHERE art_num = lcd_art))
                  CONSTRAINT PU_LIGNCDE);

CREATE TABLE Fournisseur
    (four_num     CHAR(8) PRIMARY KEY
                  CONSTRAINT PK_FOURNISSEUR,
    four_nom      CHAR(25) UNIQUE,
                  CONSTRAINT
    UNIQUE_FOURNISSEUR,
    four_adresseVARCHAR(80);
```

Figure 13.30 : Schéma relationnel en SQL-92 de l'exemple

Prise en compte des contraintes d'intégrité

Comme nous l'avons déjà vu au chapitre 7, dans un modèle Entité-Relation (MCD/MOD) peuvent être spécifiées (explicitement ou implicitement) plusieurs types de contraintes : contraintes intra-entité, contraintes intra-relation, contrainte d'intégrité générales, de stabilité, de transition,

d'identifiant relatif et enfin contraintes portant sur la participation d'une entité à plusieurs relations. Il s'agit maintenant de prendre en compte ces contraintes au niveau logique, dans un MLD relationnel, c'est à dire à les spécifier dans le langage SQL.

Contraintes intra-entité

Dans SQL-92 on dispose d'un certain nombre de clauses permettant de déclarer ce type de contraintes.

Contraintes d'intégrité d'entité : clause PRIMARY KEY

Une entité du MCD se transforme en une table du MLD relationnel. L'identifiant de l'entité devient la clé primaire de la table logique associée. Ainsi, dans le MLD relationnel de l'exemple ci-dessus, cli_num (n° client), art_num (n° article) et cmd_num (n° commande) sont les clés primaires des tables Client, Article et Commande.

La contrainte d'intégrité d'entité traduit qu'une table possède une clé primaire. Lorsque la clé primaire est constituée d'un seul attribut, la clause PRIMARY KEY dans la définition de cet attribut le déclare comme clé primaire :

```
CREATE TABLE Article
(art_num CHAR(8) DEFAULT 0 PRIMARY KEY
         CONSTRAINT PK_ARTICLE,
...
```

Lorsque la clé primaire est constituée de plusieurs attributs (clé primaire composée), la clause est spécifiée après la définition des attributs de la table :

```
CREATE TABLE Ligne_cmd
(lcd_cmd INTEGER NOT NULL,
lcd_art    CHAR(8) NOT NULL,
...
PRIMARY KEY (lcd_cmd, lcd_art)
  CONSTRAINT PK_LIGNCDE
...
```

Valeurs "obligatoires" : clause NOT NULL

En spécifiant la clause NOT NULL dans la définition d'un attribut, on impose que cet attribut possède une valeur autre que la valeur "nulle" (rappelons qu'un attribut à la valeur "nulle" s'il n'a pas de valeur). Par exemple dans la commande CREATE TABLE de la table article, la clause NOT NULL associée à l'attribut *art_nom* précise que cet attribut ne peut rester indéterminé :

CREATE TABLE Article ....

art_nom VARCHAR(25) **NOT NULL**, ...

Valeurs par défaut : clause DEFAULT

La clause DEFAULT permet de spécifier, lors de la création d'une table, une valeur par défaut pour l'un des attributs de cette table. Par exemple dans la commande CREATE TABLE de la table article, la clause DEFAULT associée à l'attribut *cli_type* précise que cet attribut aura comme valeur par défaut la valeur "Particulier" :

```
CREATE TABLE Client
...
cli_type   VARCHAR(16) DEFAULT 'Particulier',
...
```

Valeurs uniques : clause UNIQUE

Lors de la création d'une table, la clause UNIQUE permet de spécifier que pour un attribut donné de cette table, deux tuples de la table ne peuvent prendre la même valeur. Ceci revient à spécifier les clés dites "candidates" ou encore clé "alternative" de la table, clés qui n'ont pas été choisies comme clés primaires, mais qui auraient pu l'être. Par exemple dans la commande CREATE CLIENT de la table client, la clause UNIQUE associée à l'attribut *cli_nom* précise que deux clients ne peuvent avoir un même nom (hypothèse arbitraire) :

```
CREATE TABLE Client
...
cli_nom   CHAR(25) UNIQUE,
          CONSTRAINT UNIQUE_CLIENT
...
```

Domaine de valeurs défini en extension : Clause CHECK

La clause CHECK permet de spécifier une contrainte qui doit être vérifiée à tout moment par les tuples de la table. Un nombre quelconque de telles clauses peuvent être placées dans la définition d'une table. L'exemple suivant indique que l'attribut cli_type de la table Client ne peut prendre qu'une des valeurs suivantes :

```
CREATE TABLE Client
...
CHECK (cli_type IN ('Particulier', 'Administration, 'Entreprise'))
          CONSTRAINT TYPE_CLIENT);
...
```

Contraintes intrarelation

Contraintes d'intégrité référentielle : FOREIGN KEY

Nous avons vu qu'une relation conceptuelle conduisait dans le modèle relationnel à l'introduction de clés étrangères. Rappelons que, dans le modèle relationnel, une telle clé est un attribut ou une combinaison d'attributs d'une table dont les valeurs doivent correspondre à celles prises par une clé primaire d'une autre table. On appelle contrainte d'intégrité référentielle la contrainte garantissant que toute valeur d'une clé étrangère est bien égale à une valeur de clé primaire.

Lorsqu'une clé étrangère est constituée d'un seul attribut, elle est précisée directement lors de la définition de cet attribut. Ainsi, dans l'exemple précédent, on a à exprimer les contraintes d'intégrité référentielles suivantes : l'attribut *cmd_cli*, clé étrangère dans la table Commande (table qui référence), réfère à cli_num, clé primaire de la table Client (table référencée) :

```
CREATE TABLE Commande
...
cmd_cli   CHAR(8) NOT NULL REFERENCES Client
          CONSTRAINT FK_CLIENT
...
```

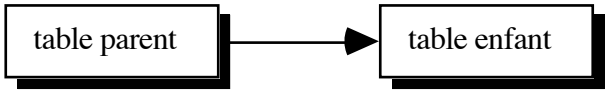
Lorsqu'une clé étrangère est constituée de plusieurs attributs, elle doit être précisée par la clause FOREIGN KEY :

```
CREATE TABLE Ligne_cmd
lcd_art   CHAR(8) NOT NULL,
(lcd_cmd  INTEGER NOT NULL,
...
FOREIGN KEY (lcd_cmd) REFERENCES Commande
          CONSTRAINT FK_COMMANDE
FOREIGN KEY (lcd_art) REFERENCES Article
          CONSTRAINT FK_ARTICLE
```

...

Contraintes de modification ou de suppression d'une clé étrangère

SQL-92 a recensé quatre actions possibles pouvant être précisées au niveau de la définition des clés étrangères au niveau des tables "enfant", en cas de modification (**ON UPDATE**) ou de supression (**ON DELETE**) de la valeur de la clé primaire correspondante dans la table "parent".



Ces options sont les suivantes : **CASCADE, NO ACTION, SET DEFAULT** et **SET NULL**. Voyons en détail chacune de ces options :

• CASCADE :

- en cas de mise à jour - ON UPDATE -, les tuples correspondant de la table "enfant" sont aussi mis à jour;
- en cas de suppression - ON DELETE -, il y a aussi suppression des tuples correspondants de la table "enfant";

- **NO ACTION** : Les changements effectués sur la clé primaire de la table "parent" sont refusés et le SGBD retourne une erreur. Cette clause correspond à l'ancienne clause RESTRICT. NO ACTION est une clause souvent utilisée, par laquelle on refuse la suppression ou la mise à jour d'une clé primaire d'une table "parent" possédant des "enfants";
- **SET DEFAULT** : la modification demandée est permise et la valeur de la clé étrangère de la table "enfant" est mise à la valeur par défaut spécifiée (cf. clause DEFAULT) dans tous les tuples concernés;
- **SET NULL** : la modification demandée est permise et la valeur de la clé étrangère de la table "enfant" est mise à NULL dans tous les tuples concernés. Notons que pour utiliser cette option, NOT NULL ne peut avoir été spécifié pour chacun des composant de la clé étrangère;

Dans notre exemple, les actions précisées dans la table Commande relativement à la clé étrangère *cmd_cli* sont les suivantes :

```
CREATE TABLE Commande
...
cmd_cli   CHAR(8) NOT NULL REFERENCES Client
          ON UPDATE CASCADE
          ON DELETE NO ACTION
          CONSTRAINT FK_CLIENT
...
```

Cette déclaration signifie que :

- **ON UPDATE CASCADE:** toute modification d'une valeur de la clé primaire de la table Client, c'est à dire l'attribut *cli_num*, conduira à une mise à jour automatique de la clé étrangère *cmd_clt* dans tous les tuples concernés de la table Commande;
- **ON DELETE NO ACTION:** toute suppression d'une valeur de la clé primaire de la table Client, c'est à dire de l'attribut *cli_num* , sera refusé.

Toujours dans notre exemple, les actions précisées dans la table Article relativement à la clé étrangère *art_four* sont les suivantes :

```
CREATE TABLE Article
...
art_four  CHAR(8) REFERENCES Fournisseur
          ON UPDATE CASCADE
          ON DELETE SET NULL
          CONSTRAINT FK_FOURNISSEUR
```

...

Cette déclaration signifie que :

- **ON UPDATE CASCADE :** toute modification d'une valeur de la clé primaire *four_num* de la table FOURNISSEUR, conduira à une mise à jour automatique de la clé étrangère *art_four* dans tous les tuples concernés de la table ARTICLE;
- **ON DELETE SET NULL :** toute suppression d'une valeur de la clé primaire de la table FOURNISSEUR, et donc d'un fournisseur donné, impliquera la mise à NULL de la clé étrangère *art_four* dans tous les tuples de la table ARTICLE qui étaient fournis par ce fournisseur. Il n'y a ainsi plus de fournisseur pour ces articles.

Nous aurions pu définir une valeur par défaut de cette clé étrangère *art_four* , par exemple la valeur "inconnu", et nous aurions alors pu utiliser l'option **SET DEFAULT:**

```
CREATE TABLE Article
...
art_four   CHAR(8) DEFAULT 'inconnu' REFERENCES Fournisseur
           ON UPDATE CASCADE
           ON DELETE SET DEFAULT
           CONSTRAINT FK_FOURNISSEUR
...
```

Toute suppression d'une valeur de la clé primaire de la table FOURNISSEUR, et donc d'un fournisseur donné, impliquera la mise à la valeur "inconnu" de la clé étrangère *art_four* dans tous les tuples de la table ARTICLE qui étaient fournis par ce fournisseur.

Contraintes référentielles dans la transformée de relation conceptuelle

Transformée d'une relation binaire fonctionnelle (cardinalité (1,1))

La transformée d'une relation binaire fonctionnelle présentant une cardinalité (1,1) conduira à spécifier dans la définition de la table associée à l'entité du coté de la cardinalité (1,1) et au niveau de la déclaration de la clé étrangère les options suivantes :

- NOT NULL
- **ON UPDATE CASCADE** ou **NO ACTION** (CASCADE étant le plus le plus probable)
- **ON DELETE CASCADE** ou **NO ACTION** (NO ACTION étant le plus le plus probable)

Ainsi, dans notre exemple (cf. figure 13.29), la relation PASSER entre les entités CLIENT et COMMANDE est du type considéré. Sa transformée en

relationnel conduit à définir dans la table COMMANDE la clé étrangère *cmd_cli* de la façon suivante :

```
CREATE TABLE Commande
...
cmd_cli   CHAR(8) NOT NULL REFERENCES Client
          CONSTRAINT COMMANDE_CLIENT
...
```

Transformée d'une relation binaire fonctionnelle (cardinalité (0,1))

La transformée d'une relation binaire fonctionnelle présentant une cardinalité (0,1) conduira à spécifier dans la définition de la table associée à l'entité du coté de la cardinalité (0,1) et au niveau de la déclaration de la clé étrangère des options suivantes :

- pas d'option **NOT NULL** ou une option **DEFAULT**
- **ON UPDATE CASCADE** ou **NO ACTION** (CASCADE étant le plus le plus probable)
- **ON DELETE SET NULL** ou **SET DEFAULT** (SET NULL étant le plus le plus probable)

Dans notre exemple (cf. figure 13.29), la relation FOURNIR entre les entités FOURNISSEUR et ARTICLE est du type considéré. Sa transformée en relationnel conduit à définir dans la table ARTICLE la clé étrangère *art_four* de la façon suivante :

```
CREATE TABLE Article
...
art_four  CHAR(8) REFERENCES Fournisseur
          ON UPDATE CASCADE
          ON DELETE SET NULL
          CONSTRAINT FK_FOURNISSEUR
...
```

Transformée d'une relation binaire (*,n)-(*,n), ternaire ou supérieure

La transformée d'une relation binaire (*,n)-(*,n), ou de toute relation ternaire ou supérieure conduira à la création d'une table spécifique à la relation dont la clé primaire composée sera constituée des clés étrangères référant aux tables associées aux entités reliées par la relation. Dans la définition de cette table associée à la relation et au niveau de la déclaration de la clé primaire composée des clés étrangères, on aura les options suivantes :

- NOT NULL
- **ON UPDATE CASCADE** ou **NO ACTION** (CASCADE étant le plus le plus probable)
- **ON DELETE CASCADE** ou **NO ACTION** (CASCADE étant le plus le plus

probable)

Dans notre exemple (cf. figure 13.29), la relation LIGNE_CMD (ligne de commande) entre les entités COMMANDE et ARTICLE est du type considéré. Sa transformée en relationnel est la table LIGNE_CMD dans laquelle les clés primaires et étrangères sont définies ainsi :

```
CREATE TABLE Ligne_cmd
(lcd_art   CHAR(8) NOT NULL,
lcd_cmd  INTEGER NOT NULL,
...
PRIMARY KEY (lcd_cmd, lcd_art) CONSTRAINT PK_LIGNCDE
FOREIGN KEY (lcd_cmd) REFERENCES Commande
        ON UPDATE CASCADE
        ON DELETE CASCADE
        CONSTRAINT FK_COMMANDE
FOREIGN KEY (lcd_art) REFERENCES Article
        ON UPDATE CASCADE
        ON DELETE NO ACTION
        CONSTRAINT FK_ARTICLE
...
```

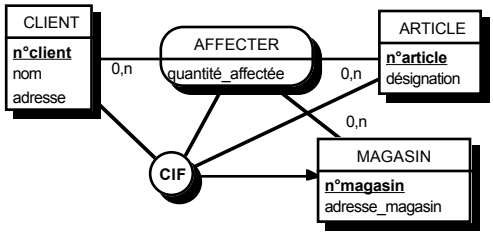
Dépendances fonctionnelles dans une relation n-aire

Dans le cas de dépendances fonctionnelles n'englobant pas la totalité de la collection, il est conseillé de décomposer la relation comme nous l'avons vu précédemment.

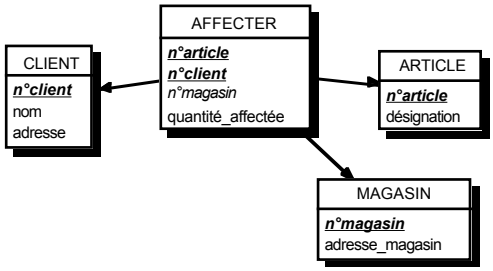
Dans le cas de dépendances fonctionnelles englobant la totalité de la collection, la contrainte conceptuelle se traduit par la spécification de clés primaires et étrangères dans la table associée à la relation conceptuelle. Soit à exprimer la contrainte suivante :

Un client ne s'approvisionne pour un article donné que dans un magasin donné.

Soit : Client X Article -> Magasin. On a alors les schémas de la figure 13.26.



Entité-Relation



Relationnel dérivé

Figure 13.31 : Transformé d'une dépendance fonctionnelle sur une relation n-aire.

Schémas relationnels associés à la figure 13.31 :

• Table CLIENT (n°client, nom, adresse) ;

• Table ARTICLE (n°article, désignation) ;

• Table MAGASIN (n°magasin, adresse_magasin) ;

• Table AFFECTER (*n°client, n°article, n°magasin*, quantité_affectée)

Comme le précise le modèle relationnel, la dépendance fonctionnelle sera traduite ici par la définition, dans la table Affecter, des trois clés étrangères, cli_num, art_num, mag_num, et de la clé primaire composée (cli_num, art_num).

Lors de la création de la table Affecter, il s'agira, par les options FOREIGN KEY et NOT NULL, de préciser les clés primaires et étrangères de la table :

```
CREATE TABLE Affecter
(cli_num  CHAR(5) NOT NULL,
(art_num  CHAR(5) NOT NULL,
(mag_num          CHAR(5) NOT NULL...)
PRIMARY KEY (cli_num, art_num)
  CONSTRAINT PRIMARY_AFFECTER
FOREIGN KEY (cli_num) REFERENCES Client ;
  CONSTRAINT AFFECTER_CLIENT
FOREIGN KEY (art_num) REFERENCES Article ;
  CONSTRAINT AFFECTER_ARTICLE
FOREIGN KEY (mag_num) REFERENCES Magasin ;
  CONSTRAINT AFFECTER_MAGASIN
```

Contraintes d'intégrité générales

Il s'agit principalement d'exprimer des contraintes sur la prise de valeurs de propriétés en fonction de valeurs déjà prises par d'autres propriétés de la même ou d'autres entités ou relations. Ces contraintes se traduisent par des contraintes d'intégrité générales sur les attributs du MLD relationnel. On peut

distinguer deux grands types de contraintes d'intégrité générale selon que leur activation soit ou non associé à l'apparition d'un événement.

Contraintes non associées à un événement

Ces contraintes sont assez bien prises en compte au travers de la déclaration d'une clause CHECK avec ou sans création d'une ASSERTION. Ainsi dans l'exemple ci-dessus, soit à exprimer la contrainte d'intégrité générale suivante :

CI : le prix unitaire de l'article *lcd_pu* de la table LIGNE_CMD ne peut être remisé à plus de 25% du prix de vente de l'article (prix catalogue) *art_pv* défini dans la table ARTICLE.

Cette contrainte peut être traitée par la clause CHECK suivante déclarée dans la table LIGNE_CMD :

```
CREATE TABLE Ligne_cmd
...
CHECK (lcd_pu >=0.75 *
        (SELECT art_pv FROM Article
        WHERE art_num = cmd_art))
        CONSTRAINT PU_LIGNCDE);
```

On peut aussi prendre en compte cette contrainte en déclarant une assertion spécifique. Une assertion est une contrainte non rattachée à une table en particulier, en conséquence souvent utilisée pour spécifier une contrainte d'intégrité définie sur plusieurs tables. La contrainte précédente est alors déclarée ainsi :

```
...
CREATE ASSERTION PU_PV
CHECK (NOT EXISTS
        (SELECT * FROM Ligne_cmd
        WHERE lcd_pu < 0.75 *
                (SELECT art_pv FROM Article
                WHERE art_num = lcd_art )));
```

Contraintes associées à un événement : triggers

Certaines contraintes d'intégrité, souvent assez complexes, doivent être exécutées dès l'apparition d'un événement donné. Jusqu'à présent, ces contraintes étaient traitées de façon procédurale par le programmeur. Dans de nombreux SGBD relationnels commercialisés, le programmeur peut dans son application déclencher la contrainte dès la survenance d'un événement par la création d'un **trigger**.

Pour chaque contrainte complexe à vérifier sur l'apparition d'un événement, on créera ainsi un trigger, consistant en une procédure compilée, cataloguée dans le dictionnaire du SGBD, et qui s'exécutera automatiquement chaque fois que

l'événement déclenchant associé se produira. Notons que certains SGBD du marché permettent de déclarer aussi des triggers au niveau de l'application et non du dictionnaire. Nous ne nous intéresserons ici qu'aux triggers déclarés au niveau du dictionnaire.

La norme SQL-2 ne normalise pas l'expression de ces triggers, SQL-3 devrait le faire. Néanmoins, une syntaxe générale pourrait être :

```
CREATE TRIGGER nom_trigger
EVENEMENT ON nom_table
WHEN (condition à vérifier)
- INSTRUCTIONS -
FOR EACH ROW / STATEMENT
```

La clause EVENEMENT

Elle permet de préciser le contexte de déclenchement du trigger. Ce contexte précise tout d'abord le type d'événement concerné, soit une insertion (INSERT), une mise à jour (UPDATE) ou une suppression (DELETE). Il doit ensuite indiquer si le trigger doit être déclenché avant (BEFORE) cet événement, ou après (AFTER). La clause EVENEMENT peut en conséquence prendre une des six valeurs suivantes : BEFORE INSERT, AFTER INSERT, BEFORE UPDATE, AFTER UPDATE, BEFORE DELETE, AFTER DELETE.

La clause WHEN

Elle permet de restreindre encore les circonstances d'exécution du trigger en précisant une condition portant par exemple sur les valeurs de certains attributs de la table.

La clause FOR EACH ROW

Elle précise que le trigger devra être exécuté pour chaque insertion, mise à jour ou suppression d'un tuple de la table concernée. La clause FOR EACH STATEMENT, clause par défaut, spécifie au contraire que le trigger devra être déclenché qu'une seule fois pour tous les tuples concernés.

Le trigger peut en partie action, spécifier un ensemble d'instructions INSTRUCTIONS. Certains SGBD ont défini un mini langage structuré (IF, THEN, ELSE, ...) permettant d'enchaîner des ordres SQL en consultation et en mise à jour sur les tables.

Pour illustrer l'utilisation de trigger, supposons que nous souhaitions, dans notre exemple, vérifier lors d'un ajout d'une nouvelle ligne de commande à une commande donnée, que le montant de cette commande, l'attribut cmd_montant de la table COMMANDE, soit bien égal à la somme des montants associés à chacune des lignes de commande de la commande, montant calculé comme le produit de la quantité commandée d'article par un

prix unitaire (lcd_qte * lcd_pu).

Dans le SGBD Oracle, un tel trigger s'exprimerait ainsi :

```
CREATE TRIGGER Calcul_montant_si_ajout
AFTER INSERT ON ligne_cmd
BEGIN
  UPDATE Commande
  SET cmd_montant = cmd_montant + :new.lcd_qte * :new.lcd_pu
  WHERE cmd.num = :new.lcd_cmd;
END;
```

Les instructions comprises entre le BEGIN et le END sont exprimées en PL/SQL, langage procédural proposé dans Oracle. On notera l'usage des variables *:new* et *:old*, la variable *:new* étant la variable associée à un nouveau tuple courant inséré (INSERT) ou mis à jour (UPDATE) et la variable *:old* étant associée à l'ancien tuple mis à jour (UPDATE) ou supprimé (DELETE).

Contraintes de stabilité

Les contraintes évoquées jusqu'ici sont de type statique. Comme nous l'avons vu précédemment, il est possible de spécifier sur un MCD des contraintes permettant d'exprimer la stabilité des états du modèle de données. Elles concernent principalement :

- les propriétés stables ;
- les pattes définitives et verrouillées.

Il est souvent difficile de traiter ce type de contraintes de façon déclarative.

Pattes définitives et verrouillées

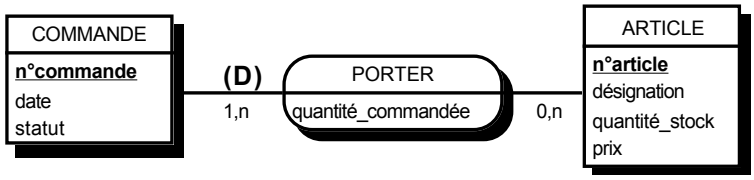
Comme nous l'avons vu au chapitre 7, une patte de relation est *définitive* (D) si une occurrence de la relation ne peut être supprimée que par et seulement par la suppression simultanée de l'occurrence correspondante de l'entité impliquée dans la patte de la relation.

De même, une patte de relation est *verrouillée* (V) si, étant donné une occurrence de l'entité reliée par cette patte, toutes les occurrences de la relation dans lesquelles cette occurrence de l'entité intervient, sont créées en même temps que l'occurrence de l'entité. Ultérieurement, on ne pourra ni ajouter ni supprimer une occurrence de la relation impliquant cette occurrence d'entité.

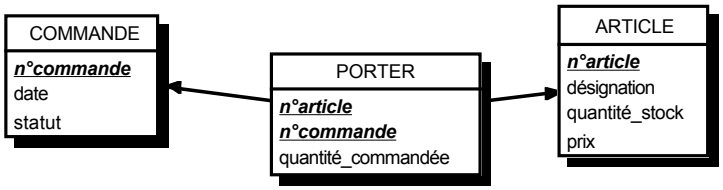
Notons que l'option Verrouillée correspond à l'option Définitive à laquelle pour une même occurrence d'entité, on interdit la possibilité de rajouter une nouvelle occurrence de relation (soit V = D + interdiction (n+1)).

La prise en compte de pattes définitives et verrouillées au niveau relationnel pourra se faire en utilisant les options de déclaration des clés étrangères vues

précédemment. Soit l'exemple suivant :



Entité-Relation



Relationnel dérivé

Figure 13.31b : Transformé de patte définitive.

Schémas relationnels associés à la figure 13.31b :

• Table COMMANDE (n°commande, date, statut) ;

• Table PORTER (*n°article, n°commande*, quantité_commandée) ;

• Table ARTICLE (n°article, désignation, quantité_stock, prix).

La prise en compte au niveau relationnel de pattes définitives (D) et verrouillées (V) pourra être faite en affectant judicieusement les options de supression/modification vues précédemment. Ainsi pour notre exemple avec patte définitive on choisira les options suivantes pour les clés étrangères de la table "enfant" PORTER :

Options relatives à la clé étrangère *n°commande*, référant à la table "parent" COMMANDE (patte définitive - D) :

- ON DELETE CASCADE
- ON UPDATE CASCADE

Options relatives à la clé étrangère *n°article*, référant à la table "parent" ARTICLE :

- ON DELETE NO ACTION
- ON UPDATE NO ACTION

Propriétés stables

Les triggers permettent aussi de traiter la plupart de ces contraintes de stabilité. On pourrait ainsi envisager un trigger assurant la stabilité de l'attribut *cmd_date* de la table COMMANDE (correspondant à la propriété stable date de l'entité COMMANDE) :

Dans le SGBD Oracle, un tel trigger s'exprimerait ainsi :

```
CREATE TRIGGER Stable_cmd_date
AFTER INSERT ON cmd_date OF Commande
  ON EACH ROW
        BEGIN
              IF :new.cmd_date !=  :old.cmd_date
                    raise_application_error (-20001, 'Date non
modifiable ');
              END IF;
        END;
```

Remarques :

- Dans le traitement par trigger de la contrainte de stabilité ci-dessus, "!=" exprime la négation dans le PL-SQL d'Oracle et que *raise_application_error* permet d'affecter un code d'erreur pouvant être ensuite traité ainsi qu'un message à l'utilisateur.
- Dans la plupart des SGBD, il est possible d'interdire une mise à jour d'un attribut à un groupe d'utilisateurs. Un usage des commande SQL "Grant" et "Revoke" permet de gérer de telles autorisations.

Enfin, certaines de ces contraintes ne pourront être prise en compte que de façon procédurale dans le développement des applications, en L3G ou L4G.

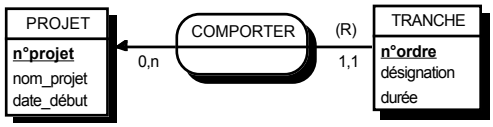
Identifiant relatif

Rappelons qu'au niveau conceptuel l'identifiant relatif ne peut s'utiliser que pour une entité type qui est émetteur d'une *dépendance fonctionnelle obligatoire* vers l'entité type "maître", c'est-à-dire une cardinalité (1,1), et si la relation qui le rattache à l'entité maître est *verrouillée* (contrainte de stabilité (V)).

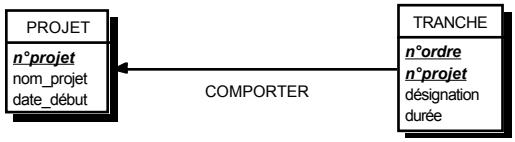
Dans l'exemple déjà rencontré dans le chapitre 7, l'identifiant relatif identifie une tranche par son ordre (ou rang) dans le projet (cf. figure 13.32). Dans le passage au relationnel, lors de la création de la table TRANCHE, il s'agira, par les options FOREIGN KEY et NOT NULL, de préciser les clés primaires et externes de la table :

```
CREATE TABLE Tranche
(n°projet  CHAR(5) NOT NULL,
n°ordre    CHAR(5) NOT NULL...
PRIMARY KEY (n°projet, n°ordre) ...
FOREIGN KEY (n°projet) REFERENCES Projet
```

ON UPDATE CASCADE
ON DELETE CASCADE



Entité-Relation



Relationnel dérivé

Figure 13.32 : Transformé d'un identifiant relatif.

Schémas relationnels associés à la figure 13.32 :

• Table PROJET (n°projet, nom_projet, date_début) ;

• Table TRANCHE (n°projet, n°ordre, désignation, durée).

Propriétés à valeurs codées

Comme nous l'avons vu au chapitre 7, il s'agit de propriétés d'entité dont la valeur s'exprime par un code (numérique ou alphanumérique, mnémonique ou non) associé à un libellé explicatif. Nous avons vu qu'en modélisation conceptuelle il était préférable de modéliser ce type de propriété comme les autres propriétés classiques , c'est à dire de les rattacher à leur entité naturelle (sans tenir compte de l'aspect code + libellé).

Soit par exemple une entité PERSONNE caractérisée par une propriété à valeurs codées "civilité" telle que :

Civilité : 1 = Monsieur

2 = Madame

3 = Mademoiselle

Cette propriété à valeurs codées doit être spécifiquement prise en compte lors du passage au niveau logique. Elle conduit à la création d'une table dite "table de références", comme l'illustre la figure suivante :

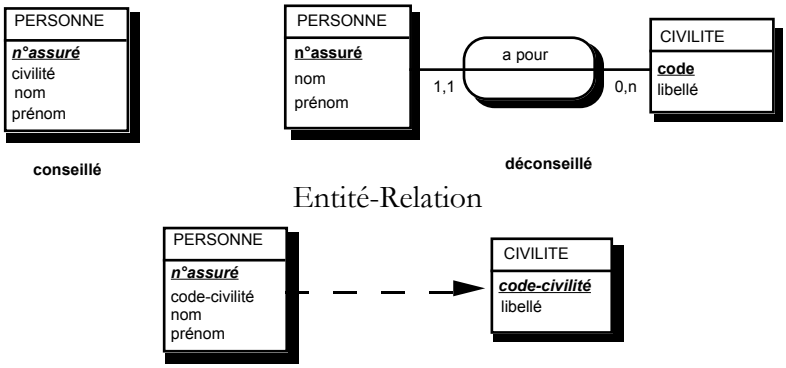


Figure 13.33 : Prise en compte au niveau logique de valeurs codées

Les schémas relationnels associés à cette modélisation logique sont:

• Table PERSONNE (n°assuré, *code-civilité*, nom, prénom) ;

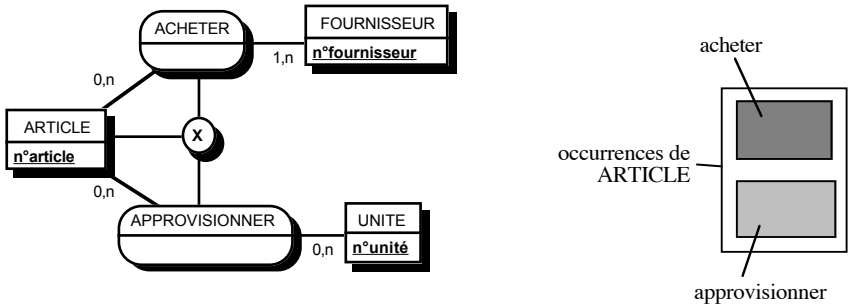
• Table CIVILITE (code-civilité, libellé).

Notons que l'implémentation réelle du lien relationnel n'est pas toujours nécessaire. Si celle-ci est implémentée, l'attribut *code civilité* de la table PERSONNE est déclaré clé étrangère et les options de mise à jour doivent être judicieusement choisies.

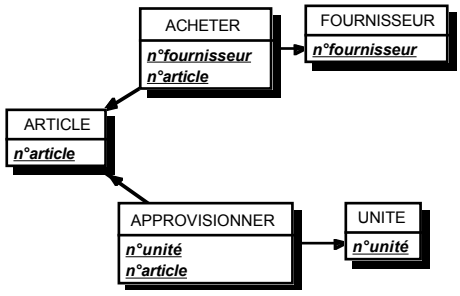
Contraintes sur la participation d'une entité à plusieurs relations

Exclusion de participation d'une entité à plusieurs relations

On l'exprime au niveau conceptuel par une contrainte X (voir " Contraintes intrarelation " du chapitre 7). Dans l'exemple suivant, par rapport à l'entité article, les relations Acheter et Approvisionner sont mutuellement exclusives, c'est à dire qu'un article donné ne peut être à la fois acheté à un fournisseur et approvisionné auprès d'une unité de production (cf. figure 13.34) :



Entité-Relation



Relationnel dérivé

Figure 13.34 : Exemple d'exclusion sur participation.

Schémas relationnels associés à la figure 13.34:

- Table ARTICLE (n°article, ...) ;
- Table ACHETER (n°article, n°fournisseur, ...) ;
- Table FOURNISSEUR (n°fournisseur, ...) ;
- Table APPROVISIONNER (n°article, n°unité) ;
- Table UNITÉ (n°unité...).

Pour exprimer la contrainte d'exclusion X en SQL-2, on déclarera l'assertion CX suivante :

```
CREATE TABLE Article
CREATE TABLE Acheter
CREATE TABLE Fournisseur
CREATE TABLE Approvisionner
CREATE TABLE Unité
...
CREATE ASSERTION CX
CHECK (NOT EXISTS
        (SELECT * FROM Acheter WHERE n°article IN
                (SELECT n°article FROM Approvisionner))
        UNION
        (SELECT * FROM Approvisionner WHERE n°article IN
                (SELECT n°article FROM Acheter ));
```

…

On pourra aussi pour exprimer la contrainte d'exclusion (X) par les deux triggers Oracle suivants :

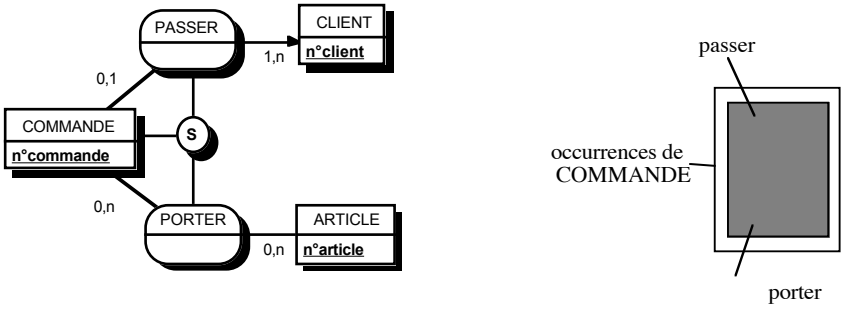
```
CREATE TRIGGER Exclusion_acheter_approvisionner
BEFORE INSERT ON Acheter
  ON EACH ROW
  DECLARE
        nb_article_approvisionner       number;
```

```
    BEGIN
        SELECT COUNT(*) INTO nb_article_approvisionner FROM
Approvisonner
            WHERE n°article = :new.n°article;
        IF nb_article_approvisionner > 0 THEN
            raise_application_error (-20002, 'Article N° ', :new.n°article,
            ' est un article d approvisionnement' ));
        END IF;
    END;
...
    CREATE TRIGGER Exclusion_approvisionner_acheter
    BEFORE INSERT ON Approvisionner
    ON EACH ROW
    DECLARE
        nb_article_acheter      number;
    BEGIN
        SELECT COUNT(*) INTO nb_article_acheter FROM Acheter
            WHERE n°article = :new.n°article;
        IF nb_article_acheter > 0 THEN
            raise_application_error (-20003, 'Article N° ', new.n°article,
            ' est un article à acheter' ));
        END IF;
    END;
...
```

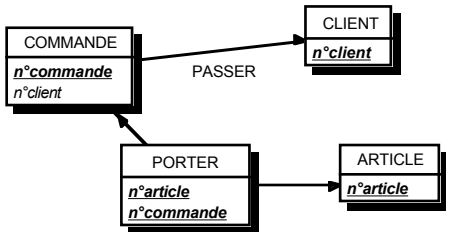
Simultanéité de participations d'une entité à plusieurs relations (ET LOGIQUE)

Toute occurrence de l'entité type participe de façon simultanée à deux (ou plusieurs) relations types. On l'exprime au niveau conceptuel par une contrainte S (voir chapitre 7 " Contraintes sur la participation d'une entité à plusieurs relations ").

Par exemple, une Commande portant sur des Articles est obligatoirement destinée à un Client (cf. figure 13.35).



Entité-Relation



Relationnel

Figure 13.35 : Exemple de simultanéité sur participation.

Avec les schémas relationnels suivants associé à la figure 13.35:

- Table COMMANDE (n°commande, *n°client*, ...) ;
- Table CLIENT (n°client, ...) ;
- Table ARTICLE (n°article, ...) ;
- Table PORTER (n° article, n°commande).

Pour exprimer la contrainte d'exclusion S en SQL-2, on déclarera l'assertion CS suivante :

```
CREATE TABLE Commande
CREATE TABLE Client
CREATE TABLE Article
CREATE TABLE Porter
...
CREATE ASSERTION CS
CHECK (NOT EXISTS
        (SELECT n°commande FROM Commande
        WHERE n°commande NOT IN
                (SELECT n°commande FROM Porter)
        AND n°client IS NOT NULL
        UNION
        (SELECT n°commande FROM Porter
        WHERE n°commande NOT IN
                (SELECT n°commande FROM Commande)
                WHERE n°client IS NOT NULL
```

La résolution d'une contrainte de simultanéité par trigger sera partielle au risque de blocage. Ainsi en Oracle, on pourra retenir l'un de ces deux triggers :

```
CREATE TRIGGER Simultaneité_Passer_Porter
BEFORE INSERT OR UPDATE n°client ON Commande
  ON EACH ROW
```

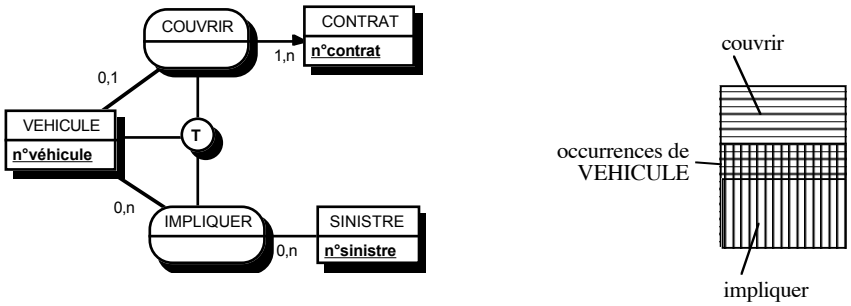
```
WHEN new.n°client IS NOT NULL
DECLARE
        nb_article_passer        number;
BEGIN
        SELECT COUNT(*) INTO nb_article_passer FROM Passer
        WHERE n°commande = :new.n°commande;
        IF nb_article_passer = 0
          raise_application_error (-20004, 'La commande n'est pas
porteuse d'articles' );
        END IF;
END;
```

ou bien :

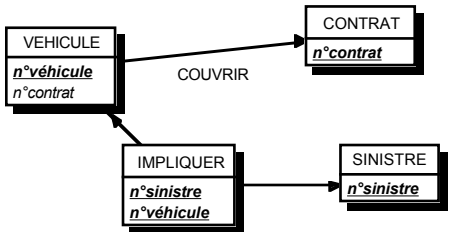
```
CREATE TRIGGER Simultaneité_Porter_Passer
BEFORE INSERT ON Porter
  ON EACH ROW
  WHEN new.n°client IS NOT NULL
  DECLARE
        nb_article_porter        number;
  BEGIN
        SELECT COUNT(*) INTO nb_article_porter FROM Commande
        WHERE n°client IS NOT NULL;
        IF nb_article_porter = 0
          raise_application_error (-20005, 'La commande n'est pas encore
           passée par un client' );
        END IF;
  END;
```

Totalité de participations d'une entité à plusieurs relations (OU INCLUSIF)

On l'exprime au niveau conceptuel par une contrainte T (voir chapitre 7 " Contraintes sur la participation d'une entité à plusieurs relations "). Exemple, tout Véhicule est reliée soit à Contrat par la relation Couvrir, soit à Sinistre par la relation Impliquer, soit les deux (cf. figure 13.36).



Entité-Relation



Relationnel dérivé

Figure 13.36 : Exemple de totalité sur participation.

Schémas relationnels associés à la figure 13.36:

- Table VEHICULE (n°véhicule, n° contrat, ...) ;
- Table CONTRAT (n°contrat, ...) ;
- Table SINISTRE (n°sinistre, ...) ;
- Table IMPLIQUER (n° sinistre, n°véhicule).

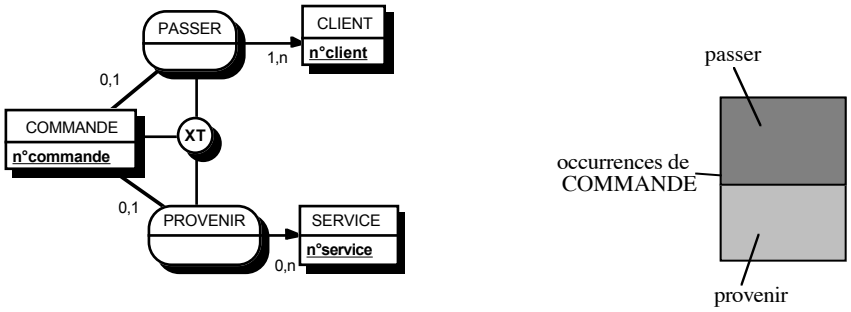
Pour exprimer la contrainte d'exclusion (T) en SQL-2, on déclarera l'assertion CT suivante :

```
CREATE TABLE Véhicule
CREATE TABLE Contrat
CREATE TABLE Sinistre
CREATE TABLE Impliquer
...
CREATE ASSERTION CT
CHECK (NOT EXISTS
        (SELECT n°véhicule FROM Véhicule
        WHERE n°véhicule NOT IN
                (SELECT n°véhicule FROM Véhicule
                WHERE n°contrat IS NOT NULL)
                UNION
                SELECT n°véhicule FROM Impliquer));
```

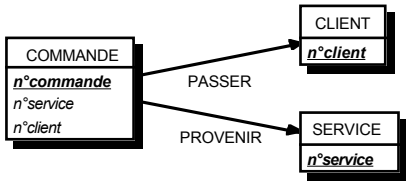
On ne peut traiter cette contrainte de totalité de participation T par triggers, car on doit d'abord insérer (INSERT) un tuple de véhicule avant de l'affecter soit par la relation "couvrir" ou la relation "impliquer" (Primary Key et Foreign Key). Un trigger sur la table Véhicule sur insertion ou mise à jour ne peut ainsi assurer aucun contrôle pertinent. Une telle contrainte sera donc traitée de façon traditionnelle dans l'application.

Partition de participations d'une entité à plusieurs relations (OU EXCLUSIF)

On l'exprime au niveau conceptuel par une contrainte XT. Reprenons l'exemple qui nous avait servi pour présenter au niveau conceptuel cette contrainte : une Commande est, soit destinée à un Client, soit à un Service interne de l'entreprise, mais pas au deux à la fois, on a ici une contrainte d'exclusion de participation de l'entité Commande aux relations Concerne/1 et Concerne/2 (cf. figure 13.37).



Entité-Relation



Relationnel dérivé

Figure 13.37 : Exemple de partition sur participation.

Schémas relationnels associés à la figure 13.37:

- Table COMMANDE (n°commande, *n°service*, *n°client*) ;
- Table CLIENT (n°client,...).
- Table SERVICE (n°service, ...).

Pour exprimer la contrainte de partition XT en SQL-2, on déclarera l'assertion CXT suivante :

```
CREATE TABLE Commande
CREATE TABLE Client
CREATE TABLE Service
...
CREATE ASSERTION CXT
CHECK (NOT EXISTS
```

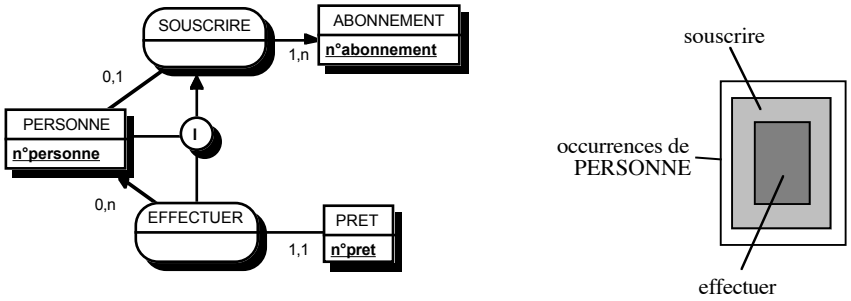
```
(SELECT n°commande FROM Commande
WHERE n°commande NOT IN
        (SELECT n°commande FROM Commande
        WHERE n°client IS NOT NULL)
        UNION
        SELECT n°commande FROM Commande));
        WHERE n°service IS NOT NULL)
```

...

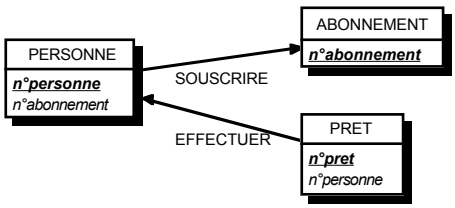
On ne peut totalement traiter cette contrainte de partition de participation XT par triggers. Si la contrainte d'exclusion X peut être traitée par deux triggers sur le modèle déjà présenté, le traitement de la contrainte de totalité T n'est pas possible pour les raisons déjà évoquées précédemment.

Inclusion de participations d'une entité à plusieurs relations (INCLUSION)

On l'exprime au niveau conceptuel par une contrainte I. Soit à exprimer que toute personne qui effectue un prêt doit avoir souscrit un abonnement. Par rapport à l'entité PERSONNE, la relation EFFECTUER est incluse dans la relation SOUSCRIRE (cf. figure 13.38).



Entité-Relation



Relationnel dérivé

Figure 13.38 : Exemple d'inclusion sur participation.

Schémas relationnels associés à la figure 13.38 :

• Table PERSONNE (n°personne, *n°abonnement, ...*) ;

- Table ABONNEMENT (n°abonnement, ...) ;
- Table PRET (n°prêt, *n°personne*, ...).

Pour exprimer la contrainte d'inclusion I en SQL-2, on déclarera l'assertion CI suivante :

```
CREATE ASSERTION I
   CHECK (NOT EXISTS
     (SELECT DISTINCT n°personne FROM Personne A
     WHERE NOT EXISTS
            (SELECT DISTINCT n°personne FROM Prêt B
            WHERE (A.n°personne= B.n°personne))
     AND n°abonnement IS NOT NULL));
```

On pourrait traiter aussi cette contrainte d'inclusion I par un trigger, comme celui-ci déclaré dans Oracle :

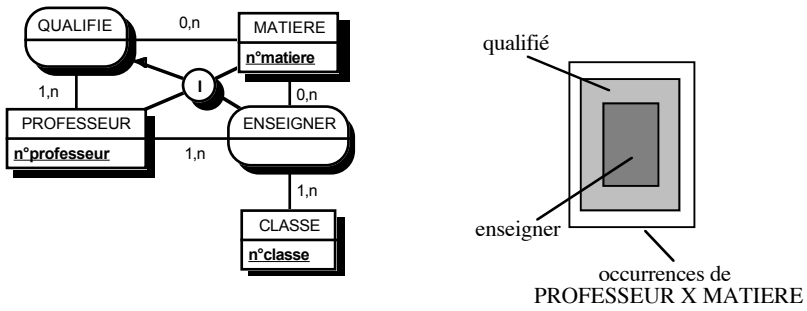
```
CREATE TRIGGER Inclusion_Effectuer_Souscrire
BEFORE INSERT ON Pret
  ON EACH ROW
            WHEN new.n°personne IS NOT NULL
  DECLARE
            nb_abonnement          number;
  BEGIN
            SELECT COUNT(*) INTO nb_abonnement FROM Personne
                     WHERE n°personne = :new.n°personne;
            IF nb_abonnement = 0 THEN
                     raise_application_error (-20006, 'Un abonnement n'a pas
été souscrit' ));
            END IF;
END;
```

Contraintes sur la participation de plusieurs entités à plusieurs relations

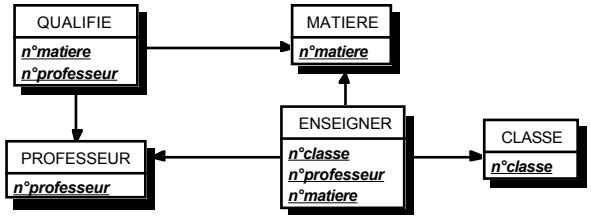
Rappelons que ces contraintes permettent d'exprimer des conditions d'existence d'occurrences de relations types selon la présence ou l'absence de participations à d'autres relations types. Nous allons étudier différentes situations pouvant se présenter (voir chapitre 7 " Contraintes sur la participation d'une entité à plusieurs relations ").

Contraintes d'inclusion de relations sur d'autres relations

Par exemple, pour qu'un professeur enseigne une matière à une classe donnée, il faut qu'il sache enseigner cette matière. On a ici une contrainte d'inclusion de la relation Enseigner dans la relation Qualifier, que l'on modélisera de la façon suivante (cf. figure 13.39) :



Entité-Relation



Relationnel dérivé

Figure 13.39 : Exemple d'inclusion sur participation de plusieurs entités à plusieurs relations.

Avec les schémas relationnels suivants (figure 13.39) :

- Table PROFESSEUR (n°professeur,...) ;
- Table MATIÈRE (n°matière,...) ;
- Table CLASSE (n°classe,...) ;
- Table QUALIFIE (n° matière, n°professeur) ;
- Table ENSEIGNER (n°classe, n°professeur, n°matière)

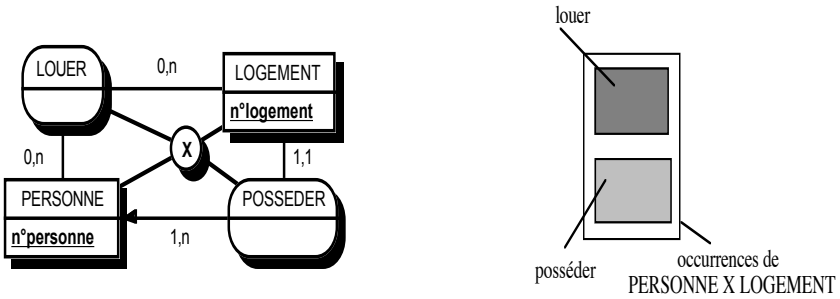
Dans Oracle, on pourrait traiter cette contrainte d'exclusion X par le trigger suivant :

```
CREATE TRIGGER Inclusion_Enseigner_Qualifier
BEFORE INSERT ON Enseigner
  ON EACH ROW
  DECLARE
        est_qualifié       number;
  BEGIN
        SELECT COUNT(*) INTO est_qualifié FROM Qualifier
        WHERE n°professeur = :new.n°professeur
              AND n°matière = :new.n°matière;
        IF est_qualifié = 0 THEN
```

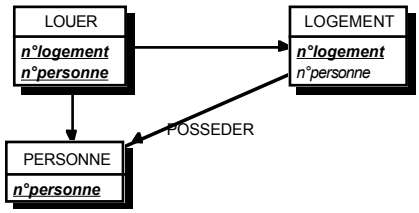
```
                raise_application_error (-20007, 'Ce professeur n'est pas
                    qualifié pour cette matière' ));
            END IF;
END;
```

Contraintes d'exclusion de relations sur d'autres relations

Par exemple, une personne ne peut pas être locataire et propriétaire d'un même logement , comme l'illustre la figure suivante (cf. figure 13.40) :



Entité-Relation



Relationnel dérivé

Figure 13.40 : Exemple d'exclusion sur participation de plusieurs entités à plusieurs relations.

Schémas relationnels associés (figure 13.39) :

- Table LOUER (n° logement, n° personne) ;
- Table LOGEMENT (n°logement, *n°personne,...*) ;
- Table PERSONNE (n°personne, ...)

Dans Oracle, on pourrait exprimer cette contrainte d'exclusion X par les deux triggers suivants :

```
CREATE TRIGGER Exclusion_Louer_Logement
BEFORE INSERT ON Louer
  ON EACH ROW
  DECLARE
        possède_déjà    number;
```

```
BEGIN
        SELECT COUNT(*) INTO possède_déjà FROM Louer
        WHERE n°personne = :new.n°personne
                AND n°logement = :new.n°logement ;
        IF possède_déjà > 0 THEN
                raise_application_error (-20008, 'Cette personne est déjà
                    propriétaire de ce logement' ));
        END IF;
END;
```

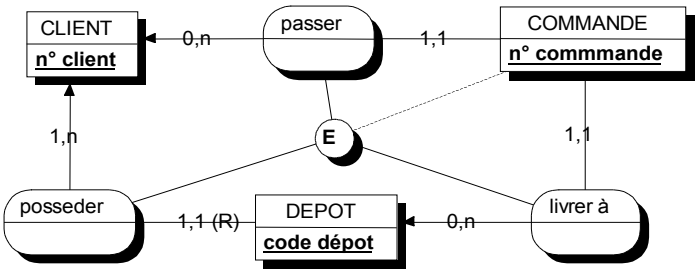
et :

```
CREATE TRIGGER Exclusion_Logement_Louer
BEFORE INSERT ON Logement
  ON EACH ROW
        WHEN new.n°personne IS NOT NULL
  DECLARE
        nb_location       number;
  BEGIN
        SELECT COUNT(*) INTO nb_location FROM Louer
        WHERE n°personne =: new.n°personne
        IF nb_location > 0 THEN
                raise_application_error (-20009, 'La personne n° ',
new.n°personne,
                    ' est déjà locataire' ));
        END IF;
  END;
```

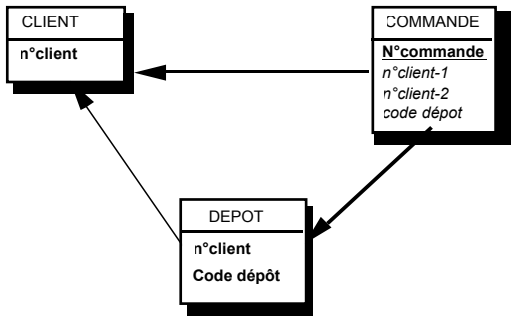
Contraintes sur les occurrences de relations ayant des entités communes - Contrainte d'égalité

Soit une entité I1 reliée indirectement à une entité I2 par des relations binaires fonctionnelles formant des cheminements distincts. Pour toute occurrence de I1, l'occurrence de I2 obtenue par l'un des cheminements est identique à celle obtenue par l'autre cheminement. On l'exprime par une contrainte E dite "contrainte d'égalité".

Reprenons l'exemple que nous avons déjà vu au chapitre 7. Pour toute occurrence de commande, l'occurrence de client donneur d'ordre de la commande (relation passer) est toujours identique à l'occurrence de client propriétaire (relation posséder) du dépôt auquel doit être livrée (relation livrer à) la commande. La figure suivante illustre la modélisation conceptuelle associée spécifiant une identification relative de l'entité DEPOT par rapport à l'entité CLIENT et une contrainte d'égalité E. La dérivation au niveau logique est la suivante :



Entité-Relation



Relationnel dérivé

Figure 13.41 : Prise en compte au niveau logique de contrainte d'égalité

Les schémas relationnels associés sont:

• Table CLIENT (n°client).

• Table COMMANDE (n°commande, n°client-1, n°client-2, code dépôt);

• Table DEPOT (n°client, code depot).

L'identifiant relatif conduit à la clé primaire composée n°client, code depot dans la table DEPOT. Dans la table COMMANDE, *n°client-1* est la clé étrangère réalisant le lien avec la table CLIENT et n°client-2, code dépôt constitue la clé étrangère (composée) vers la table DEPOT. Compte tenu de la contrainte d'égalité et de l'identification relative, à déclarer pour la table COMMANDE deux clés étrangères identiques (n°client-1 et n°client-2) aussi peut-on les fusionner, ce qui donne alors le nouveau schéma relationnel suivant :

• Table CLIENT (n°client).

• Table COMMANDE (n°commande, *n°client, code dépôt);*

• Table DEPOT (n°client, code depot).

Quantification des modèles logiques de données

Une première quantification en volume du MOD (modèle organisationnel de données) a déjà été effectuée, elle a permis notamment de définir les cardinalités moyennes et le taux de participation des relations (voir figure 13.42).

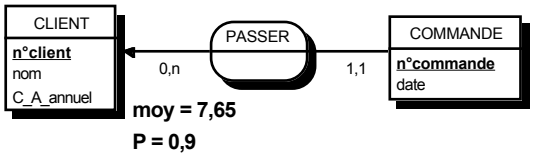


Figure 13.42 : Représentation d'une cardinalité moyenne sur un MCD.

Au niveau logique, il s'agira simplement d'affiner cette première estimation en tenant compte de la structure logique des données correspondant à la modélisation conceptuelle. Ainsi pour un MLD relationnel :

la cardinalité moyenne d'une patte de relation du MCD devient la cardinalité moyenne du lien relationnel associé. Elle est alors définie par le rapport :

nb d'occurrences table enfant

nb d'occurrences table parent

le taux de participation d'une table enfant à une table parente est défini par le rapport :

nb d'occurrences de la table parente référées

nb total d'occurrences table parent

Ce taux est de 100 % si la cardinalité mini est de 1. Dans notre exemple, on aura le schéma de la figure 13.43.

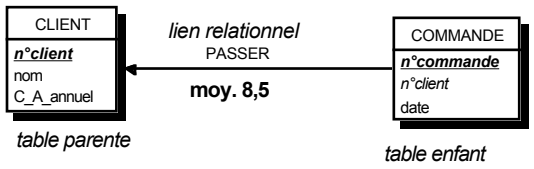


Figure 13.43 : Cardinalité moyenne d'un lien relationnel.

Le concepteur exprime ainsi l'ensemble des résultats du chiffrage :

- Pour chaque table/record,
  - la taille totale, en caractères = somme de la taille des attributs composant la table ;
  - la durée de vie des occurrences de la table;
  - le nombre d'occurrences maximum mémorisables dans la mémoire

immédiate.

- Pour chaque lien relationnel,
  - la cardinalité moyenne ;
  - le taux de participation ;
  - optionnellement la cardinalité maxi ou mini à 95 %.

Évaluation du volume global du modèle logique de données

A partir des éléments quantifiés précédemment déterminés, le concepteur peut procéder à une évaluation grossière du volume total des données à mémoriser, pour la mémoire immédiate. Cette évaluation sera affinée ultérieurement en tenant compte de l'optimisation (abordée ultérieurement) et des caractéristiques techniques du système de gestion de base de données utilisé.

Dans le cas d'un MLD relationnel, ce volume se décompose en deux parties :

- Le volume consacré aux données proprement dites contenues dans les tables " primaires ", ou volume " utile ". Il peut être estimé par

$\sum_{\text{toutes les tables}}$ taille des tables x nombre d'occurrences maxi.

- Le volume dû aux index secondaires qui seront installés sur ces tables (ces index sont, dans la plupart des SGBD relationnels commercialisés, des tables à part entière — " tables index " —, leur estimation en volume suit alors le mode précédent).

Ce volume total est une estimation de la taille nécessaire pour la mémorisation de la future base de données. En pratique, une bonne exploitation de la base de données nécessitera un volume utile bien plus important, on adoptera un coefficient de 2 à 3 ou plus selon les requêtes effectuées sur la base et les performances du SGBD adopté.

Modèles logiques de données répartis et client-serveur

Problématique générale

Nous avons vu dans la deuxième partie, relative à l'étude du SIO (système d'information organisationnel), un premier type de répartition des données exprimée dans les MOD (voir chapitre 10). Cette répartition organisationnelle des données consistait tout d'abord à choisir, à partir du MCD, les données à mémoriser informatiquement et celles qui ne le seraient pas (MOD Global), ensuite à répartir les données informatisées selon les unités organisationnelles (MOD locaux), en définissant pour chaque unité et pour chaque donnée, les droits d'accès autorisés ainsi que les modalités de partage des données entre ces unités organisationnelles.

Pour les données comme pour les traitements, les répartitions

organisationnelles et logiques sont très différentes. La *répartition organisationnelle des données* est perceptible par l'utilisateur et concerne l'organisation de ses tâches à travers le besoin et la disponibilité des données que l'on peut spécifier au travers des MOD répartis (cf. chapitre 10). La *répartition logique de données* est elle liée à l'implantation informatique des données sur des machines logiques différentes. Notons que la répartition logique est transparente à l'utilisateur, et n'a aucun impact sur son organisation mais concerne exclusivement l'informatique. La figure suivante illustre cette problématique :

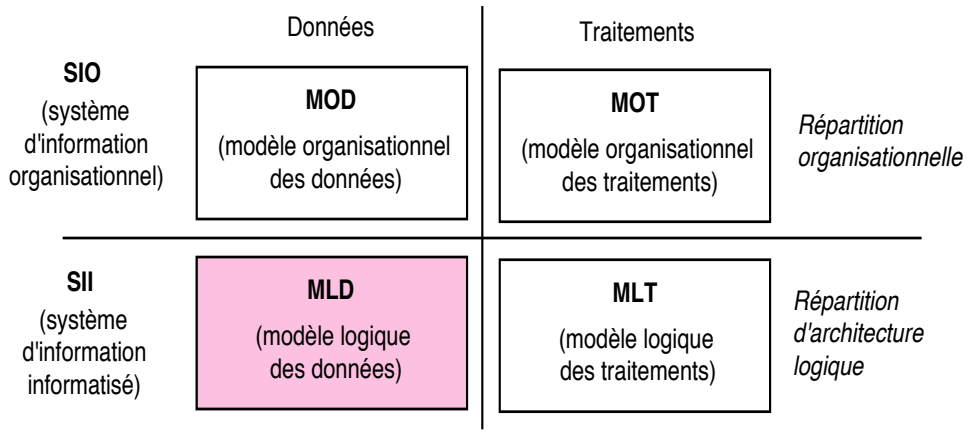


Figure 13.44 : Répartition logique des données.

Critères de répartition

Les critères de répartition reposent souvent sur des considérations techniques (volume des échanges, capacité des systèmes). Aussi la quantification des modèles logique de données que nous venons de traiter a ici tout son intérêt. De même on peut tirer parti de la répartition organisationnelle, donc des MOD, pour tenter de rapprocher le plus possible les données de leur lieu d'utilisation, en distinguant entre autre la mise à jour et la lecture comme nous le verrons par la suite.

La répartition des données au niveau logique doit être perçue comme un arbitrage entre des considérations techniques (caractéristiques des matériels) et l'utilisation des données mémorisées (localisation, fréquence et nature des accès) qui provient du niveau organisationnel. Nous l'illustrons dans la solution du cas X que nous proposons.

Dans la répartition logique des données un point nous semble important à évoquer, c'est la mise en cohérence de données dupliquées. Il y a duplication de données lorsque des occurrences identiques de tables sont mémorisées sur des sites différents. Il est alors nécessaire de préciser les mécanismes qui permettront d'assurer la cohérence des valeurs; leur choix dépend des utilisations de ces données (fréquence, action).

Répartition et architectures client - serveur

La répartition informatique (logique et physique) des données est particulièrement concernée dans la mise en œuvre des architectures client-serveur .

Bien que la méthode Merise se positionne essentiellement sur le terrain de l'ingénierie des systèmes d'information, avec un accent particulier sur le système d'information organisationnel (SIO), pour lequel les considérations technologiques ne sont pas la préoccupation centrale. Toutefois, l'évolution technologique apportée par le client-serveur est suffisamment importante pour s'interroger sur son impact sur les raisonnements à mettre en oeuvre pour la conception d'un système d'information dans un tel environnement. Ainsi la mise en œuvre d'architecture client-serveur est grandement facilité dans Merise deuxième génération du fait que :

- les modélisations des données et des traitements proposées sont tout à fait adaptées aux nouvelles possibilités apportées par le client-serveur;
- la répartition organisationnelle des données et des traitements (MOD - MOT) s'avère un élément décisif dans le choix de la répartition informatique dans une solution client-serveur.

Aussi développerons-nous en détail ces deux points dans les paragraphes suivants.

Modélisations conceptuelle et organisationnelle des données pour le client-serveur

Le client-serveur a incontestablement consacré la place centrale des données dans la conception d'un système d'information. Par sa puissance d'expression sémantique et sa facilité de transformation en formalisme relationnel, la modélisation conceptuelle et organisationnelle des données en Entité-Relation proposée par Merise confirme incidemment son importance.

Modélisation des données et serveur de données relationnel

La répartition des données et des traitements sur des systèmes distincts a rendu encore plus nécessaire une analyse des données indépendante des traitements. Cette situation privilégie une démarche de conception et de structuration des données à partir de leur sémantique. Rappelons qu'il s'agit d'un des principes fondamentaux de la méthode Merise.

Les raisonnements proposés par la méthode Merise pour la conception d'une base de données relationnelle restent donc toujours pertinents pour la définition de serveurs de données relationnels, comme l'illustre la figure suivante :

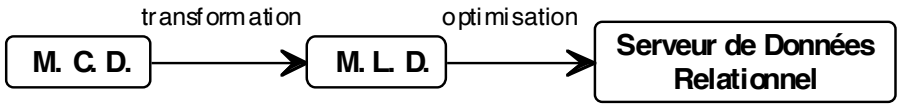


Figure 13. 45: Raisonnements proposés par Merise pour la définition de serveurs de données.

La nécessité d'une modélisation conceptuelle/organisationnelle des données pour concevoir un serveur de données relationnel se confirme d'ailleurs par la présence de tels outils dans la grande majorité des AGL. Notons que les formalismes utilisés pour cette modélisation dans ces AGL peuvent être différents (surtout pour les outils d'inspiration anglo-saxone) et leur puissance d'expression plus ou moins importante, comme nous l'avons déjà été évoquée dans la présentation du formalisme Entité-Relation de Merise.

Modélisation des données et systèmes ouverts

La possibilité de développer des systèmes ouverts conduit progressivement à concevoir un serveur de données sur un domaine d'activité avant de connaître et de spécifier les différents traitements qui pourront ultérieurement utiliser ce serveur. Un tel contexte rend indispensable une conception de la base de données centrée sur la sémantique. L'approche préconisée depuis plusieurs années par Merise trouve ici toute sa justification. Dans le contexte actuel des serveurs relationnels, même de deuxième génération, nous restons persuadés qu'une modélisation conceptuelle/organisationnelle de données avec le formalisme Entité-Relation de Merise reste encore la démarche la plus efficace pour concevoir un serveur de données relationnel dans un système ouvert.

Modélisation des données et systèmes d'information d'aide à la décision

Dans un système d'information d'aide à la décision, l'utilisateur recherche une vision globale de l'entreprise, puisant ses informations dans divers systèmes d'information de production déjà existants, répartis sur différents serveurs de données. Les traitements consistent fréquemment en des analyses de données (tableaux, graphiques, analyses statistiques) et des simulations; les logiciels "classiques" de la micro offrent de multiples possibilités à l'ergonomie adaptée et appréciée.

Dans le développement de tels systèmes d'information, le problème principal est la connaissance du contenu des différents systèmes d'information de production. Les modélisations conceptuelle/organisationnelle des données prennent ainsi une importance capitale. Les concepteurs doivent très tôt vérifier si la perception globale de l'utilisateur du futur SI d'aide à la décision correspond aux perceptions modélisées dans les MCD/MOD des SI de production. En particulier, ils doivent s'assurer de la présence d'entités partagées (elles permettent de faire le lien entre les systèmes) et à leur concordance de signification (le produit vendu, l'item fabriqué, la fourniture achetée peuvent ils être considérés comme un sous-type d'article ?)

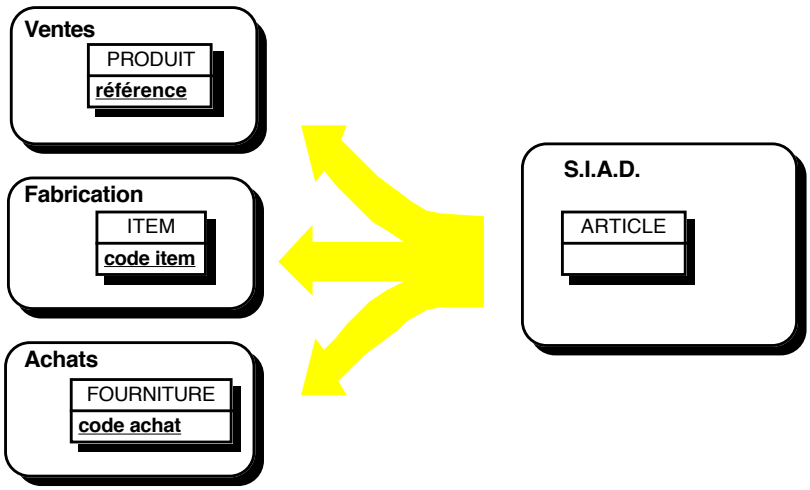


Figure 13. 46 : MCD et système d'information d'aide à la décision.

Là encore, le synthétisme et la puissance d'expression d'un MCD/MOD tel que le préconise Merise est un atout pour représenter et connaître le contenu des différents serveurs de données. Ce rôle ne fait que se confirmer avec le développement du "data warehousing".

Modélisation des données et serveurs deuxième génération

Cette nouvelle génération se caractérise par la prise en charge de traitements par le serveur; la plupart de ces traitements étant déclarés sous forme de "triggers", "règles" ou "procédures" gérées par le même logiciel que les données. En général, ces traitements sont décrits dans le dictionnaire commun de données et de traitements.

Cette migration des traitements vers le serveur a une double justification:

- des préoccupations techniques de performance qui ont été justifiées dans la première partie de ce chapitre;
- la nécessité, dans des systèmes ouverts, d'associer plus intimement les données et les règles de traitement qui préservent sa cohérence sémantique; ces règles et contraintes ont une permanence et un champ d'application indépendant du contexte d'utilisation par tel ou tel traitement.

L'expression de ces règles et contraintes peut s'effectuer sur le modèle conceptuel de données. C'est dans cet objectif que nous avons introduit les contraintes inter relations, les contraintes de stabilité et le concept de règle dans le formalisme Entité - Relation (cf. chapitre 7). Ces différents concepts sont traduits au niveau logique (cf. chapitre 12) et implémentables pour la plupart par des "triggers".

Modélisations logiques des données et traitements répartis

L'une des caractéristiques principales de l'architecture client serveur est la possibilité de répartir les données et les traitements sur des systèmes différents. Comme nous l'avons vu dans la première partie du chapitre, cette répartition est le critère d'une typologie de clients-serveurs.

Fréquemment cette répartition est globale et ne nécessite pas une modélisation spécifique. Ainsi, dans le cas d'une application où l'ensemble de la présentation et des traitements est sur les postes clients, et l'ensemble des données concernées est implanté sur un serveur unique, il est superflu d'utiliser une quelconque modélisation pour représenter une telle répartition.

Par contre , dans le cas d'architectures client-serveur distribuées, les traitements et les données d'une application peuvent être répartis entre les postes clients et plusieurs serveurs. Il est alors utile de représenter la répartition des différents traitements et données entre les différents systèmes.

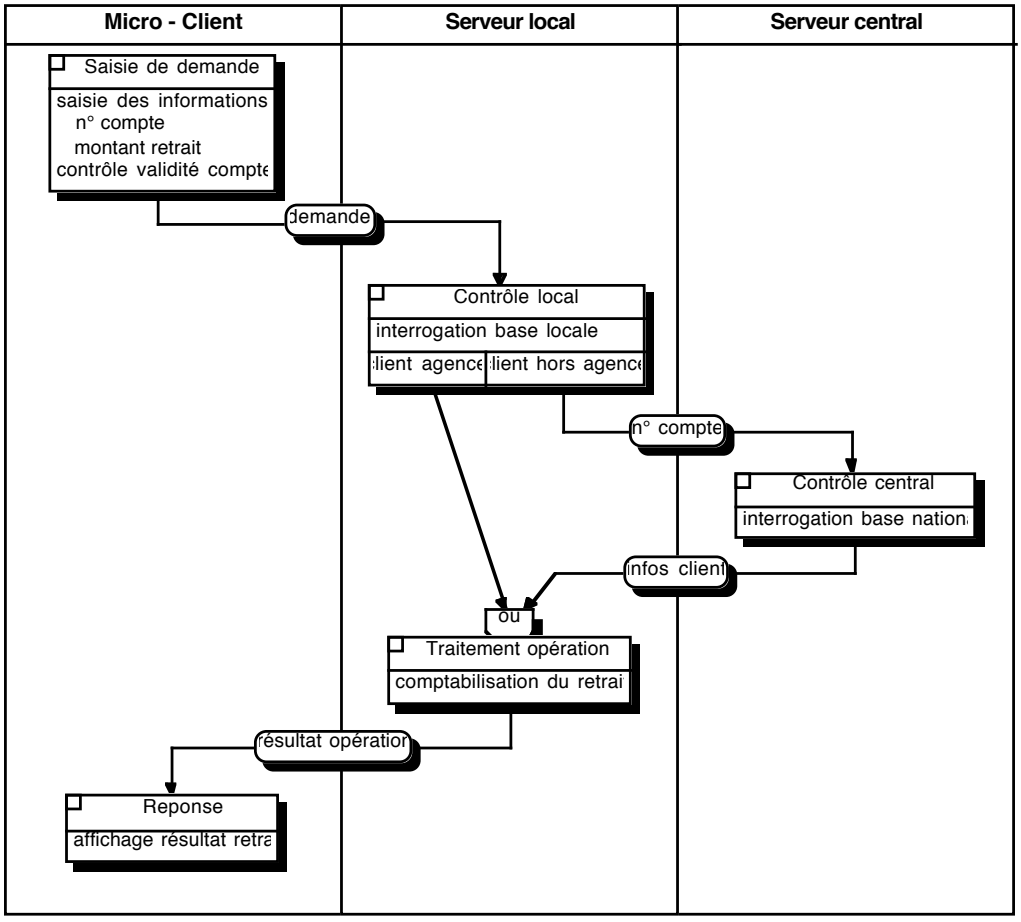


Figure 13.47 ; Exemple de répartition des traitements en client-serveur.

Dans Merise deuxième génération, la modélisation logique des traitements (voir chapitre 12) permet d'exprimer complètement cette répartition au moyen

:

- des machines logiques,
- de la décomposition des unités logiques de traitement (ULT) globales en ULT par nature.

La figure 13.47 présente un MLT spécifiant une répartition logique de traitements client-serveur.

Les modèles organisationnels de données locaux (voir MOD - chapitre 10) ont été spécialement introduits dans Merise pour prendre en compte la répartition des données d'une application sur plusieurs systèmes, en précisant également les notions d'accès aux données et de références. Le cas société X traité au chapitre 10 illustre cette problématique de la répartition entre plusieurs serveurs.

De la répartition organisationnelle à la répartition informatique

B.P.R., répartition organisationnelle et client-serveur

Les tendances actuelles de la conception des systèmes d'informations, tirant profit des principes de réorganisation des activités proposés par le B.P.R., préconisent une organisation des entreprises en petites unités organisationnelles relativement autonomes, chacune disposant des moyens nécessaires pour assurer ses missions avec le maximum de flexibilité et d'efficacité. Dans cette optique, les systèmes clients-serveurs, en facilitant la répartition des données et des traitements, permettent de se libérer progressivement des architectures hiérarchiques (techniques et organisationnelles) induites par les systèmes centralisés, tout en préservant l'accessibilité des données nécessaire à la cohérence globale de l'entreprise. La répartition informatique des données et des traitements est alors au service d'une répartition organisationnelle.

Cependant, il nous semble aujourd'hui que de nombreuses applications développées ou transposées en client-serveur n'ont opéré qu'un simple changement de technologie, en concentrant sur le même serveur l'ensemble des données antérieurement implantées sur un système centralisé. La répartition informatique se réduit alors à celle techniquement induite par la nouvelle technologie. Pourtant, les capacités d'ouverture et d'interopérabilité du client-serveur, le "down-sizing" accompagnant sa diffusion, rendent accessibles techniquement et économiquement une meilleure répartition des systèmes d'information informatisés.

Pour bénéficier pleinement des possibilités offertes par le client-serveur pour concevoir de nouveaux systèmes d'informations plus modulaires, la plupart des spécialistes en la matière préconisent de rapprocher les données et les traitements de leur site d'utilisation. La répartition informatique est alors au service de la répartition organisationnelle.

La méthode Merise dispose, avec les différents modèles dédiés à la description du système d'information organisationnel, d'un ensemble de raisonnements permettant d'exprimer cette répartition des données et des traitements. Elle peut ainsi fournir une aide méthodologique pour la conception d'un système d'information informatisé distribué, dans un environnement client-serveur.

Répartition organisationnelle des traitements

La répartition des activités s'exprime essentiellement autour de la notion d'unité organisationnelle qui recouvre généralement un ensemble de postes représentant par exemple un service ou un site géographique.

La représentation de cette répartition des activités s'effectue fréquemment sous la forme de macro-MOT ou MCT réparti, une modélisation assez proche du degré de détail d'un MCT courant sur lequel on exprime la répartition entre les unités organisationnelles. La figure 13.48 illustre la répartition des opérations entre les unités organisationnelles dans le cas société X (cf. MOT - fin du chapitre 8).

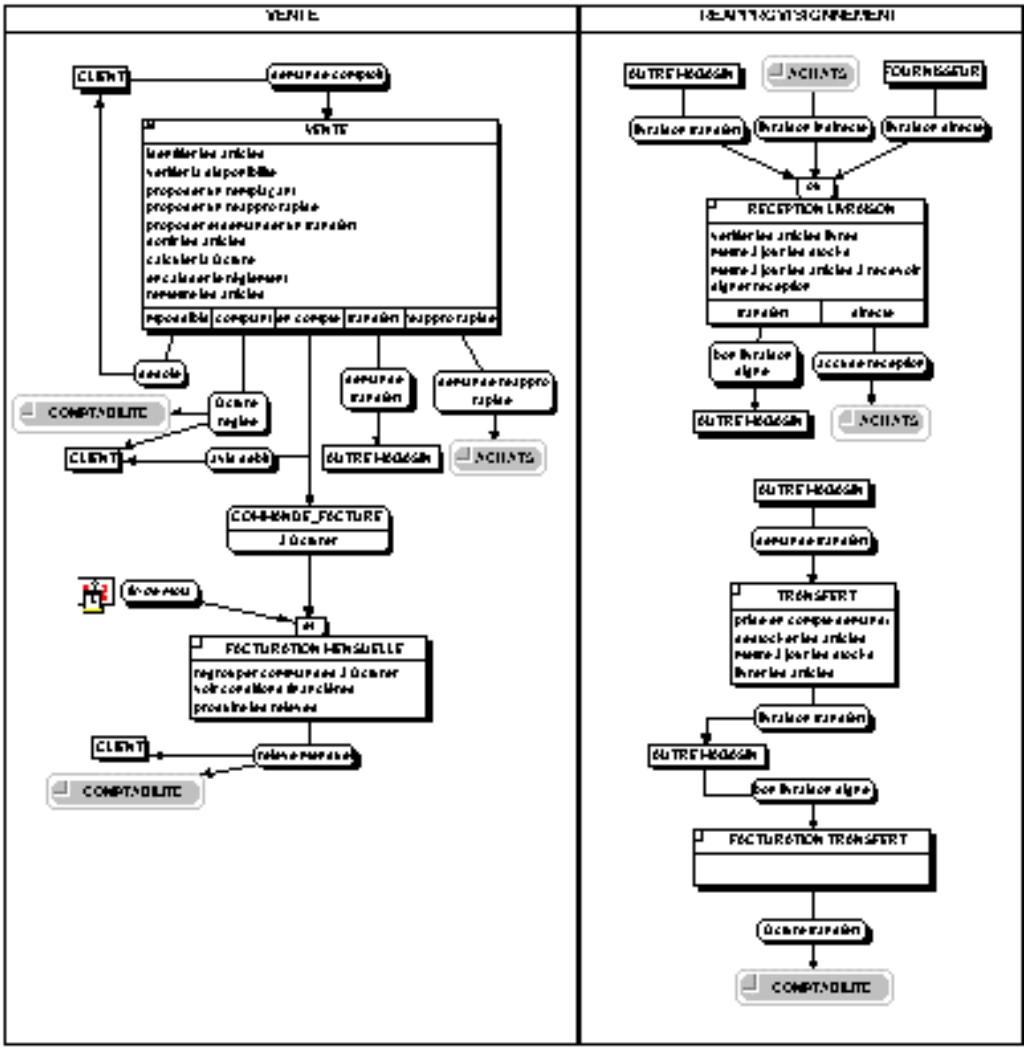


Figure 13.48 : Répartition des activités entre unités organisationnelles

La mise en évidence de cette répartition doit s'effectuer assez tôt dans l'étude, généralement en étude préalable. Cette répartition peut être soit décidée à priori, soit déduite d'une analyse plus fine des MOT et obtenue ensuite par un regroupement de postes.

Répartition organisationnelle des données

Cette répartition se formalise par les MOD locaux qui expriment l'ensemble des données communes et partagées par les utilisateurs d'une même unité organisationnelle (voir chapitre 10). Rappelons qu'un MOD local se représente pour chaque unité organisationnelle par :

- un schéma des entités, des relations et propriétés utilisées,
- un tableau précisant l'accessibilité des données (actions autorisées,

restriction d'occurrences)

On peut adopter deux démarches complémentaires pour l'élaboration des MOD locaux :

- définir a priori, à partir du MOD global et connaissant les activités des différentes unités organisationnelles, la répartition et l'utilisation des entités et des relations,
- déduire les MOD locaux en cumulant les sous-schémas de données associés aux phases ou tâches prises en charge par chaque unité organisationnelle.

Des indicateurs pour la répartition logique des données

Dans le cas d'une implémentation sur un serveur de donnés unique, nous avons vu que le passage du MCD/MOD au MLD puis à la base de données relationnelle était quasiment automatique, en appliquant des règles formelles.

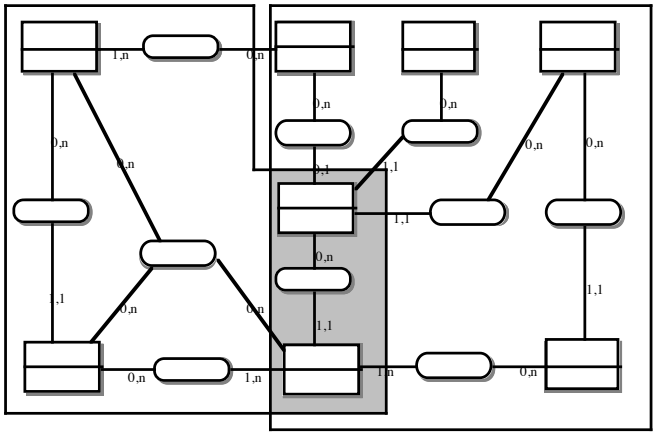
Dans le cas d'une répartition des données sur plusieurs serveurs, les critères influants sur le choix de répartition sont multiples et ne peuvent aujourd'hui être automatisés. S'il est certain que les aspects techniques ont une part indéniable, la répartition de l'utilisation des données exprimée dans les MOD locaux fournit, comme nous allons le présenter, de très utiles indications pour orienter une répartition informatique.

Degré de partage

Il y a partage lorsque des entités ou des relations figurent dans plusieurs MOD locaux. L'intérêt d'une répartition logique dépend d'une part de l'étendue du partage, d'autre part des différents accès envisagés. Une première règle de répartition, relevant du "bon sens", (cf. figure 13.49) peut s'exprimer ainsi :

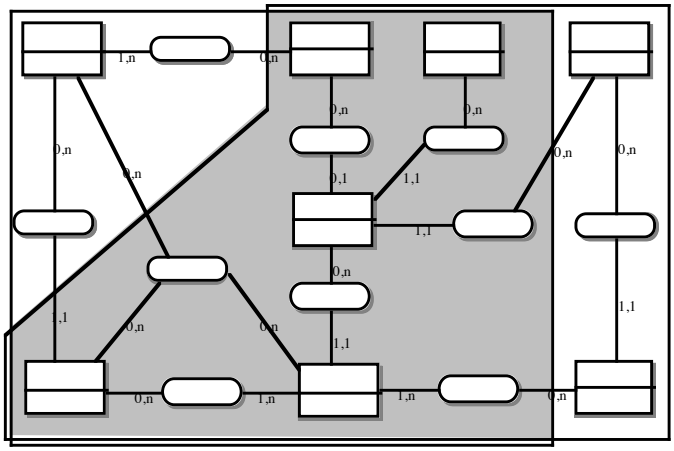
- si peu de données sont partagées, alors une répartition informatique quasi-naturelle s'ébauchera;
- si les données partagées sont importantes, en type et en occurrences, la répartition informatique est assez déconseillée.

Faible intersection



répartition informatique bénéfique

Forte intersection



peu propice à une répartition informatique

Figure 13.49 : Degré de partage et répartition informatique des données

Accessibilité

Pour les données partagées, une seconde règle, basée sur le type d'accès, peut être appliquée :

- des données en lecture seule peuvent être implantées sur un serveur distant sans trop de contraintes;
- des données en mise à jour (création, modification, suppression) doivent être de préférence implantée sur un serveur plus proche.

Le type d'accès doit également pondéré par la fréquence d'accès. Ce qui nous donne le tableau synthétique suivant :

	Fréquences	d'accés
Actions	Faible	Elevée

Mise à jour	acceptable	défavorable
Lecture	favorable	acceptable

Situation pour un accès distant

Le caractère de proximité peut être physique et/ou technique (puissance de débit du réseau, capacité d'accès du serveur).

Duplication

Les règles précédentes cherchaient à privilégier une localisation informatique unique des différentes données, même en cas d'utilisation partagée. Cette unicité de mémorisation est, encore actuellement, une bonne garantie pour le maintien de la cohérence; bien que les systèmes clients - serveurs annoncent des possibilités de réplication de plus en plus fiables et efficaces.

Dans le cas où des données sont partagées avec des actions de mise à jour relativement importantes, il peut être nécessaire de dupliquer ces informations sur des serveurs différents. Se pose alors le problème du maintien de la cohérence entre les serveurs. Deux grandes catégories de solutions peuvent alors être utilisées :

- une mise à jour différée
- la "réplication" des données sur les serveurs avec la technique de "commit en deux phases"

Le choix de la solution dépend certes des possibilités techniques, mais s'apprécie également à travers les conditions d'utilisation des données exprimés au niveau du système d'information organisationnel par les MOT et MOD.

Les solutions du cas de la Sté X proposées (cf. CD-ROM) illustrent différents exemples de répartition, duplication et synchronisation.

En conclusion ...

La technologie client-serveur a apporté directement ou indirectement deux grandes avancées : d'une part la généralisation des interfaces graphiques pour les systèmes d'information informatisés et d'autre part la possibilité de répartir des données et des traitements sur des systèmes distincts et autonomes.

La première est, à notre avis, la plus visible et appréciée des utilisateurs mais a confronté les informaticiens à de nouvelles logiques de construction d'interfaces (ergonomie et dialogue). Reconnaissons que, sur ce point, la méthode Merise n'apporte pas vraiment d'aide méthodologique, pas plus qu'elle ne l'apportait précédemment pour la conception des interfaces transactionnelles. Les modèles logiques de traitement intègrent certes la présentation comme composant de la modélisation logique des traitements mais sans donner de conseil sur sa construction.

La seconde, plus discrète, ouvre des possibilités encore peu exploitées. Nous

avons voulu montrer que la méthode Merise non seulement restait d'actualité pour concevoir des serveurs de données de deuxième génération, mais proposait également, avec son approche du système d'information organisationnel, un cadre méthodologique opérationnel pour maîtriser la mise en place de systèmes distribués au service de l'organisation.

14

Optimisation des Modèles de Données

Introduction

Problématique de l'optimisation

Les modèles logiques de données (MLD) dérivés directement des modèles conceptuels/organisationnels de données (MCD/MOD) ont jusqu'à présent ignoré les problèmes d'accès, de performance, de volume et de coût, pour se concentrer exclusivement, par la prise en compte sur des structures informatisables, de la signification des informations.

La définition des traitements, au travers des modèles organisationnels de traitements (MOT) et des modèles logiques de traitements (MLT), précise la nature et la fréquence de ces accès aux données. Ces accès influencent de façon déterminante les performances des traitements. Une amélioration des performances de ces derniers pourra être obtenue en optimisant l'organisation logique et l'implémentation physique des données en agissant donc sur le modèle logique de données (MLD) et le modèle physique des données (MPD).

L'optimisation des modèles logique et physique de données nécessite ainsi une très bonne connaissance des traitements qui y seront effectués. Il est certain que des traitements n'effectuant que des accès aux données en consultation ne conduiront pas aux mêmes optimisations que celles qui seraient conduites pour des traitements effectuant des accès en mises à jour nombreuses à ces mêmes données. De même, la nature des traitements, conversationnelle ou différée, unitaire ou par lots, conduira à des optimisations différentes.

En conséquence, l'optimisation des modèles de données du système d'information informatisé sera toujours un compromis global, consistant à définir une organisation et une implémentation des données conduisant à des performances globalement satisfaisantes pour l'ensemble des traitements. Ce compromis portera principalement sur :

- le volume global occupé par les données mémorisées ;

- le temps nécessaire pour accéder à ces données mémorisées ;
- la contrainte de transfert entre les données stockées et l'unité centrale ;
- des contraintes diverses et particulières à certains systèmes.

Nous allons dans ce chapitre présenter cette optimisation spécifiquement à l'environnement relationnel déjà étudié dans le cadre de la modélisation.

Optimisation en relationnel

Les bases de données relationnelles présentent beaucoup d'intérêt, résultant principalement de la puissance du modèle relationnel auquel elles se réfèrent. Bien que ce modèle soit apparu pratiquement au même moment que le modèle navigationnel, il a fallu attendre de nombreuses années pour que les systèmes de gestion de bases de données (SGBD) relationnels présentent, pour des développements d'une certaine taille, des performances suffisantes, comparables à celles des SGBD navigationnels, et pour leur être préférés.

Les performances des SGBD relationnels continuent à s'améliorer, d'une part, grâce à l'usage d'un module intégré au cœur du système, appelé " optimiseur de requêtes ", de plus en plus sophistiqué et, d'autre part, grâce à une parallélisation de plus en plus importante des traitements des requêtes, conduisant aux machines bases de données. Cependant, l'optimisation du modèle logique relationnel n'en reste pas moins une étape à la fois incontournable et délicate.

L'optimisation des modèles de données dans les systèmes relationnels se situe à la fois au niveau logique et au niveau physique. Nous précisons ici un ensemble d'optimisations possibles. Il est certain que cette optimisation est étroitement liée au SGBD retenu ; aussi pour un SGBD donné, certaines des optimisations proposées dans cet ouvrage ne seront pas possibles, d'autres optimisations, non évoquées, pourront alors être possibles. Notons qu'à notre connaissance, très peu d'ouvrages, spécialisés en bases de données, traitent cette optimisation de façon approfondie. Nous recommanderons cependant la lecture du travail de recherche réalisé par Daniel Richard [Richard 89].

Dans cet ouvrage, nous distinguerons deux grands types d'optimisations :

- *L'optimisation physique* qui consiste à tirer partie des possibilités du SGBD relationnel utilisé pour implanter au mieux les différentes tables de la base de données, ceci sans remettre en cause les modèles logiques de données déjà définis.
- *L'optimisation logique* ou *dénormalisation* qui consiste à effectuer différentes adaptations du modèle logique de données relationnel brut (dérivé du MCD/MOD) qui est normalisé. Ces adaptations reviennent à dénormaliser ce modèle. Cette dénormalisation devra être conduite avec une extrême prudence.

Optimisation physique

Comme nous vennons de le dire, l'optimisation physique consiste à tirer partie des possibilités du SGBD relationnel utilisé pour implanter au mieux les différentes tables de la base de données, ceci.

Cette optimisation consiste d'une façon générale à exploiter judicieusement les possibilités du SGBD liées au stockage des données sur disque. Elle nécessite en conséquence une très bonne connaissance du SGBD, de ses posssibilités liées à l'implantations des données sur disque mais aussi que du fonctionnement de son optimiseur de requête notamment en ce qui concerne l'exploitation qu'il fait de ces structures de stockage. En ne remettant pas en cause les modèles logiques de données, elle doit être abordée en premier lieu, avant l'optimisation logique associée à une dénormalisation des modèles logiques. L'optimisation physique porte notamment sur les points suivants :

- le choix des structures de stockage physique selon lesquelles seront implantées les tables du modèle logique relationnel;
- la définition des chemins d'accés primaires liés aux organisations indexées (index - clé de placement);
- la mise en place de chemin d'accès secondaires ou index secondaire ;
- le partition horizontales ou verticale des tables ;
- le regroupement (clustering) des tables, etc.

Nous allons aborder en détail ces différents points, en sachant que chaque SGBD présente ses propres spécificités qu'il est nécessaire de maîtriser pour aborder cette optimisation, optimisation relevant souvent de l'administrateur de bases de données.

Choix d'une organisation des tables (chemins d'accès primaires)

Performance et structures de données

Sollicité par une requête un SGBD relationnel peut toujours trouver une donnée, mais un choix judicieux de structures de stockage supportant les tables de la base de données ou choix de l'organisation des tables, permettra de réduire ce temps de recherche.

Un tel choix peut être judicieux pour certaines requêtes alors que pour d'autres il peut être pénalisant. Ce choix peut aussi affecter l'espace disque nécessaire (plus d'espace nécessaire) et avoir des conséquences gênantes, par exemple, sur la gestion de la concurrence (encore plus délicate).

La plupart des SGBD relationnels commercialisés proposent plusieurs organisations ou méthodes d'accès possibles pour stocker les tables de la base de données. Notons que ce choix d'organisations possibles est fortement lié au

système d'exploitation pour lequel le SGBD relationnel est proposé. En effet, le système d'exploitation gère des organisations spécifiques mis à la disposition du SGBD. Parfois, ce dernier peut proposer des organisations nouvelles se rajoutant à celles déjà offertes par le système d'exploitation.

Le système d'exploitation Unix propose ainsi de nombreuses organisations de fichier possibles et en conséquence, les SGBD relationnels sous Unix proposent généralement de plusieurs organisations possibles pour les tables de la base. Par exemple, avec un SGBD sous Unix une table pourra être organisée tout d'abord en :

- HEAP (fichier séquentiel) : c'est généralement l'organisation par défaut.

Sont ensuite proposées diverses organisations indexées (index principal, clé primaire de placement) par exemple les organisations suivantes :

- HASH (clé calculée) : une organisation aléatoire caractérisée par un accès sur une valeur exacte de la clé et une fonction de randomisation ne pouvant en général pas être modifiée par l'utilisateur.
- ISAM : une organisation indexée caractérisée par un tri selon une clé des données stockées, permettant un accès rapide sur valeur exacte ou partie de clé et reposant sur un index statique nécessitant une réorganisation quand la table augmente.
- BTREE : une organisation indexée caractérisée par un trie des données selon une clé, un accès rapide sur valeur exacte ou partie de clé et reposant sur un index dynamique (arbre balancé - Btree).

Il est en général possible par une commande (Modify) de changer l'organisation d'une table :

- `Modify employe To heap` : réalisera une réorganisation de la table Employé en fichier séquentiel;
- `Modify employe To hash On nom` : fera une réorganisation de la table Employé en organisation aléatoire selon l'attribut nom;
- `Modify employe To btree On nom, âge` : conduira à une réorganisation de la table Employé en organisation indexée Btree selon les attributs nom et âge.

Choix de l'organisation d'une table

Disposant de plusieurs organisations possibles pour stocker les différentes tables de la base de données, l'administrateur de base de données pourra choisir pour chaque table l'organisation la plus adaptée. Par exemple :

L'organisation séquentielle (HEAP) sera préférée :

- pour le chargement de données dans la base ;

- lorsque la table est petite (occupation de peu de pages) ;
- lorsque les requêtes manipulent des tables entières ;
- pour récupérer de l'espace dû à des DELETE , la commande MODIFY employé TO heap permettra cette récupération. On pourra ensuite réorganiser la table dans l'organisation préférée par un autre MODIFY.

Cette organisation sera évitée lorsque :

- on a des accès à un ou plusieurs tuples ;
- les tables sont grosses.

L'organisation aléatoire HASH sera préférée pour toute recherche sur valeur exacte de clé (la plus rapide). Elle est à éviter lorsque :

- les requêtes nécessitent des recherches sur partie de clé ;
- on a des traitements de table entière ;
- on a des joints naturels (systématiques sans restriction) ;
- on a des débordements dès le départ (on ne peut ajuster la fonction de randomisation dans Ingres).

• L'organisation indexée ISAM sera recommandée lorsque :

- les requêtes nécessitent des recherches sur une partie de clé ;
- la table grossit lentement (peu de réorganisations nécessaires) ;
- la clé est de taille importante.

Cette organisation n'est pas recommandée si :

- la recherche est sur clé complète (on lui préférera l'organisation HASH) ;
- la table est grosse et à croissance rapide.

• L'organisation indexée BTREE sera préférée lorsque :

- les requêtes nécessitent des recherches sur une partie de clé ;
- la table grossit vite ;
- la table est trop grosse pour être souvent réorganisée (MODIFY) ;
- les requêtes conduisent à des joints de tables entières.

Elle sera évitée lorsque :

- la table est statique ou à croissance faible ;
- la clé est de taille importante ;
- il y a beaucoup d'ajouts de nouveaux tuples seulement en fin de table

entraînent un plus grand risque de verrous mortels (dead-locks) par utilisateurs (problèmes de concurrence).

Le tableau de la figure 14.10 synthétise, pour un SGBD donné, la pertinence de chacune de ces organisations possibles, en fonction de la nature de la table ou d'opérations types majeures qui y sont réalisées par les traitements.

OPERATIONS SUR LA TABLE	HEAP	HASH	ISAM	BTREE
chargement de table	1	-	-	2
effacer tuples doubles	-	1	1	1
recherche sur clé complète	-	1	2	2
intervalles/pattern matching	-	-	1	1
recherches séquentielles	1	3	2	2
recherche sur clé partielle	-	-	1	1
accès à données triées	-	-	-	1
joints sur larges tables	-	-	1	1
index croît comme table	-	-	-	1
très petite table	1	-	-	-
très grande table	-	-	-	1

(1) : préféré ; (2) : acceptable ; (3) : médiocre ; (-) : déconseillé ou impossible

Figure 14.10 : Choix pour l'organisation d'une table.

Optimisation par ajout d'index secondaires (chemins d'accès secondaires)

Index primaire et secondaire, clés plaçante et secondaire

Le critère de recherche le plus fréquemment utilisé sur une table relationnelle concerne la clé primaire. Elle peut être constituée d'un attribut (clé primaire simple) ou de plusieurs attributs (clé primaire composée). Placer les tuples de cette table sur disque selon la valeur de cette clé primaire facilitera grandement l'accès à la table selon ce critère. Cela sera possible en adoptant pour la table une organisation indexée, dans laquelle la clé primaire de la table sera aussi *clé plaçante*. On parle alors d'*index primaire*. Comme nous venons de le voir, le choix judicieux, pour chaque table de la base d'une d'organisation parmi différentes organisations proposées par le SGBD adopté (principalement indexées), est un choix d'optimisation important.

Les applications sur la base de données effectuent aussi des accès aux données selon de multiples critères de recherche. L'installation d'*index secondaires* permet d'éviter un coûteux balayage séquentiel de la table. Un index secondaire constitue un réordonnancement logique des tuples de la table en fonction d'une clé d'accès, pouvant être discriminante ou non, simple ou composée, différente de la clé plaçante sur laquelle est basée l'organisation de la table. Cette clé est dite *clé secondaire* (voir figure 14.12).

Lors d'une recherche sur index secondaire, le balayage de l'index secondaire

donne les identifiants de tous les tuples, appelés aussi " tuples identifiants ", satisfaisant le critère de recherche. Les tuples sont ensuite accessibles directement (à raison d'une entrée/sortie par page) via l'identifiant et selon l'organisation de la table.

Dans beaucoup de SGBD relationnels, un index secondaire installé sur une table A selon un attribut a est tout simplement une table organisée selon une organisation indexée particulière, dont les attributs sont la clé a et l'identifiant de tuple associé ou TIDP = pointeur sur le tuple de la table A. La figure 14.12 présente l'exemple d'index secondaire index_salaire dans un SGBD sur une table Emp d'employés, selon l'attribut salaire.

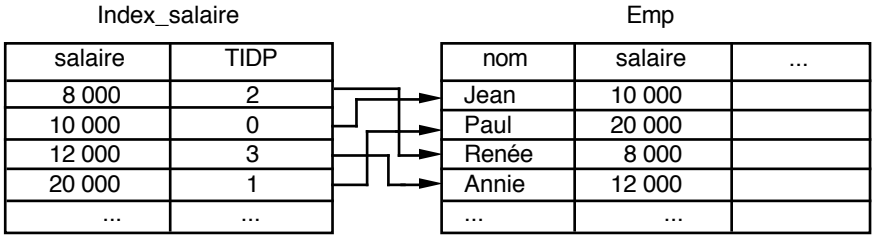
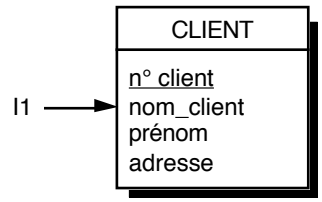


Figure 14.12 : Index secondaires.

Le TIDP est un pointeur sur le tuple (adresse de page et offset sur la page) de la table emp. L'index secondaire index_salaire est une table de la base de données, qui a autant de tuples que la table emp (index dense) et qui est organisée par défaut par exemple en ISAM.

La création d'un index secondaire facilite ainsi grandement certains accès, mais introduit aussi des coûts de stockage et de mise à jour pouvant être importants. Pour le coût de stockage, plus la taille de la clé à indexer sera grande plus l'encombrement de l'index secondaire sera important. Pour le coût de mise à jour, il est clair que toute mise à jour de la table indexée entraînera la mise à jour de tous les index secondaires portant sur cette table. Aussi, la mise en place d'index secondaires ne se fera que si elle est justifiée, restreinte à des critères de recherche les plus fréquents.

Il peut être intéressant de spécifier sur un MLD graphique l'installation d'index secondaires, on peut le faire comme sur la figure 14.13.



I1 est un index secondaire installé sur l'attribut nom_client de la table CLIENT

Figure 14.13 : Index secondaire sur la table Client.

Enfin les index secondaires sont généralement exploités par les *optimiseurs de requêtes* pour faciliter l'exécution de requêtes sur la base, notamment des jointures.

Installation d'index secondaires sur attributs non clés

On peut installer des index secondaires sur des attributs non-clé plaçante sur des tables relationnelles issues d'entités. Il convient d'étudier l'intérêt d'une telle décision d'optimisation en fonction des fréquences prévisibles des accès sur ces attributs.

D'une façon générale, l'installation sur une table d'un index secondaire sur un attribut non clé est justifiée par de nombreux accès à la table selon cet attribut. L'index réduira le coût de ces accès. La contre partie est un coût de stockage lié à l'augmentation du volume à mémoriser de la base et qui dépend de la taille de l'attribut à indexer, de la ou des organisations indexées proposées par le SGBD adopté pour les index secondaires. Signalons aussi un coût supplémentaire associé aux mises à jour de la table (créations, modifications, suppressions) nécessitant aussi à des mises à jour de l'index.

Installation d'index secondaires sur attributs clés étrangères

On peut installer des index secondaires sur des attributs clés étrangères participant ou non à une clé primaire composée :

Index sur la clé primaire composée : la clé primaire étant une clé composée, elle peut avoir une taille importante. Aussi, ce type d'index ne sera utile que lorsqu'il existe des requêtes fréquentes portant sur les tuples de la table désignés par des valeurs complètes de la clé.

Index sur un ou plusieurs attributs composant la clé primaire : ce type d'index permet de réduire de façon considérable (principalement au niveau de l'optimiseur) les accès lors de requêtes dans lesquelles on ne connaît la valeur que d'une partie des attributs de la clé, cas très fréquent.

Index sur un attribut hors clé primaire : il ne sera utile que lorsqu'il existe des requêtes fréquentes portant sur les tuples de la table désignés par des valeurs de cet attribut.

D'une façon générale, l'installation sur une table d'un index secondaire sur un attribut clé étrangère est justifiée par de nombreux accés à la table selon cet attribut. Ainsi que de nombreuses jointures de cette table avec la table vers laquelle la clé étrangère pointe. La contre partie est aussi un coût de stockage qui dépend de la taille de l'attribut à indexer ainsi qu'un coût de mise à jour à évaluer.

L'encodage et la compression des données

La réduction de l'espace nécessaire au stockage des données peut être

déterminante. Une solution est alors l'encodage et la compression des données. Les trois techniques principales sont :

- *la suppression des blancs et des zéros dans les valeurs* : Elle est faite à chaque accès, ce qui entraîne des mises à jour plus coûteuses. Elle peut s'appliquer sur les structures SEQ, HASH, ISAM, BTREE et conduire à un gain en espace disque important. La compression de données est à éviter lorsque les données sont peu compressibles (exemple : l'attribut adresse plus compressible que l'attribut nom) ou qu'il y a de nombreux index secondaires à mettre à jour.
- *la substitution de séquences* : Elle consiste à associer un code aux séquences de caractères les plus fréquentes et à le substituer à chaque occurrence de ces séquences.
- *l'encodage statistique* : Il tient compte de la fréquence d'apparition des caractères (un caractère qui apparaît fréquemment aura une représentation plus courte que celle d'un caractère rare).

Partition de tables relationnelles (segmentation)

Cette optimisation a pour objet de réduire les coûts d'accès aux données en éliminant des informations inutilement transférées de la mémoire secondaire à la mémoire centrale (entrées/sorties). En effet, les attributs et tuples des tables d'une base de données relationnelle sont rarement tous utilisés dans les requêtes ; Knuth [Knuth 73 dans Richard 89] a montré que seulement 20 % des données effectivement stockées dans la base sont concernées par 80% des requêtes lancées sur la base.

Aussi, il peut être judicieux de partitionner des tables afin de réduire le flux des informations en entrée/sortie. Le partitionnement (voir figure 14.14) consiste à éclater soit verticalement en paquets d'attributs, soit horizontalement en paquets de tuples, certaines tables de la base de données. Le paquet qui sera identifié comme le paquet le plus sollicité pourra, par exemple, être stocké sur le support mémoire secondaire le plus performant.

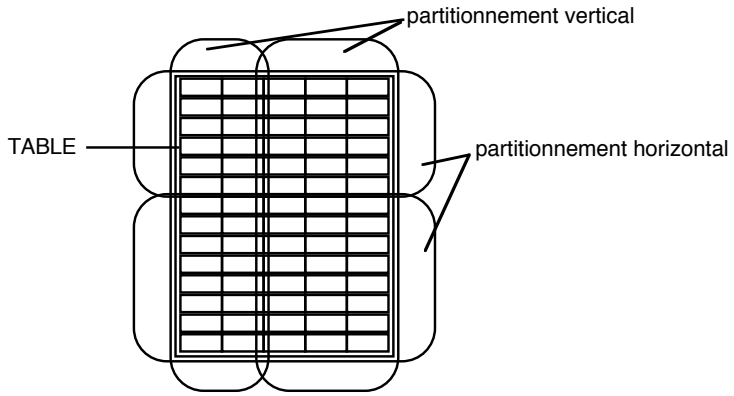


Figure 14.14 : Différents types de partitions.

deTout se passe comme si une table du MLD était découpée en projections ou restrictions arbitraires (non soumises aux règles de la normalisation) et stockée dans des sous-tables implantées physiquement dans des zones mémoires différentes.

Le partitionnement vertical consiste à éclater une table en sous-tables regroupant les attributs les plus souvent invoqués ensemble. Le partitionnement doit être totalement transparent aux utilisateurs qui ne connaissent que les schémas logiques des tables ; ce qui nécessite un prétraitement de leurs requêtes. Les problèmes rencontrés lors de ce partitionnement sont alors [Richard 89] :

- le choix d'une bonne partition (table à n attributs : $n^{n/2}$ partitions possibles);
- les coûts de certaines transactions utilisant des attributs répartis dans des paquets différents peuvent se révéler supérieurs à ce qu'ils seraient si les attributs n'étaient pas répartis.

Le partitionnement horizontal consiste à subdiviser les tuples d'une table en sous-tables de même schéma. On retrouve la table initiale par simple union algébrique des sous-tables. On regroupera ensemble les tuples ayant une forte probabilité d'être accessibles ensemble. Chaque sous-relation pourra avoir une organisation propre, optimisant l'accès. Les intérêts de ce partitionnement sont [Richard 89] :

- il peut apporter des solutions au problème de protection de données sensibles;
- il peut se révéler efficace pour limiter la désorganisation des tables ayant un sous-ensemble de tuples soit de longueurs variables, soit pouvant avoir une valeur nulle, soit dont la valeur est plus souvent modifiée. La sous-table contenant ces tuples sera alors réorganisée aussi souvent que nécessaire, sans que les autres sous-tables le soient.

Le partitionnement horizontal doit cependant être adopté avec prudence car

s'il favorise les requêtes les plus fréquentes ou celles mettant en jeu les plus gros volumes d'informations, il peut au contraire pénaliser fortement d'autres requêtes.

Le regroupement de tables ou "clustérisation"

Certains SGBD relationnels permettent l'usage de " clusters ". Le rôle de ces clusters est de regrouper physiquement, sur une même page (une entrée/sortie), des données provenant d'une ou plusieurs relations et vérifiant des prédicats de sélection précisés. L'utilisation des clusters est totalement transparente à l'utilisateur.

En général, le regroupement concerne deux tables, afin d'optimiser les jointures sur ces tables, c'est le principal intérêt de la clustérisation. Il est aussi possible de définir des clusters sur une seule table, ce qui permet d'implanter les deux variantes de partitionnement précédemment évoquées [Richard 89].

La clustérisation peut être, dans ses finalités, comparée à des optimisation plus structurelles dénormalisantes, mais elle présente le grand avantage de ne pas modifier la structure logique en se plaçant uniquement au niveau de l'implantation physique des données.

Dans le SGBD Oracle, il faut créer tout d'abord les clusters, puis y placer une ou plusieurs tables. Si plusieurs tables sont regroupées dans un cluster, elles doivent avoir au moins un attribut en commun. Le regroupement des tables dans un cluster a pour conséquences que :

- chaque valeur de cet attribut commun est stockée une fois dans la base;
- les tuples d'une table ayant la même valeur pour cet attribut sont stockés dans une même zone du disque (page);
- SQL*plus crée un index, appelé cluster index, sur les attributs regroupés.

Toujours dans Oracle, il est possible soit de créer une table et de placer sa définition dans un cluster, soit de créer la table et la placer dans un cluster existant. La première façon convient pour créer une table, la seconde pour ajouter une table existante dans le cluster [J.P. Perry et J.G. Lateer]. La syntaxe de la création d'un cluster dans Oracle est la suivante :

```
CREATE CLUSTER nom_cluster
(attribut_cluster1 type_données, attribut_cluster2 type_données...)
[SIZE taille_logique]
[SPACE nom_zone]
[COMPRESS \ NOCOMPRESS];
```

Il est aussi possible de supprimer des clusters, il faut alors d'abord en extraire

les tables.

Optimisation logique ou dénormalisation

Rappelons que l'optimisation logique ou dénormalisation consiste à effectuer différentes adaptations du modèle logique de données relationnel dérivé du MCD/MOD, modèle qui est normalisé. Ces adaptations reviennent à dénormaliser ce modèle.

Cette dénormalisation devra être conduite avec une extrême prudence. Nous précisons ici un ensemble de règles générales d'optimisation structurelles du modèle logique relationnel. Ces optimisations structurelles entraînent des dénormalisations dont les conséquences doivent être estimées.

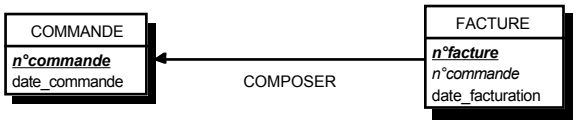
L'utilisation de ces règles garantit le respect des spécifications initiales ; tout autre type de modification, sous couvert d'optimisation, peut entraîner une altération de la sémantique du modèle conceptuel de données. Certes, l'efficacité de ces différentes optimisations dépend du SGBD utilisé ; certaines même ne seront pas toujours possibles sur tel ou tel système.

Pour chacune des règles d'optimisation, nous indiquons les conditions incitant à la pratiquer, les gains espérés et les pertes générées, ainsi que les conséquences secondaires. Le concepteur optimise son modèle par modifications successives de la structure. A chaque pas, il doit évaluer le bien fondé de son choix.

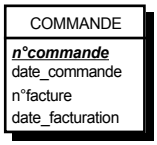
Regroupement de tables relationnelles

Ce type d'optimisation a, en partie, déjà été suggéré lors de la dérivation du MLD relationnel à partir du MCD/MOD. Deux tables relationnelles reliées par un lien relationnel un vers un peuvent être regroupées en une même table, afin de réduire des accés, mises à jour ou jointures sur ces deux tables. Cette opération d'optimisation peut dans certains cas conduire à une dénormalisation dont les conséquences devront être estimées. D'une façon générale, deux tables ayant la même clé primaire peuvent être regroupées. La figure 14.15 présente un exemple de regroupement de tables.

Entité-Relation



Relationnel non optimisé

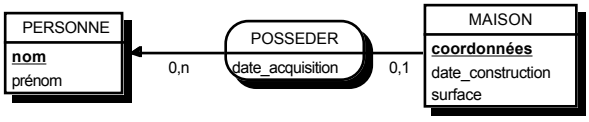


Relationnel optimisé

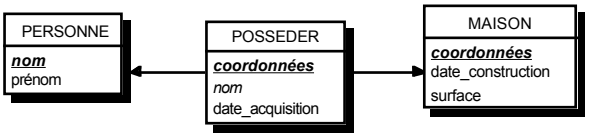
Figure 14.15 : Regroupement de tables.

D'une façon générale, le regroupement de tables est justifié par de nombreux accès à des tables reliées par des liens relationnels un vers un. La contre partie est une augmentation du volume à mémoriser pour la table A = Σ taille des attributs migrés de la table B x Nombre de tuples de la table A accueillant la migration. Une conséquence induite est que si la taille des attributs de B migrant dans A est importante, il peut y avoir augmentation du coût de certaines opérations sur A. Il faut aussi prévoir la gestion des valeurs nulles sur les attributs migrants. La figure 14.16 présente deux exemples de regroupement de tables.

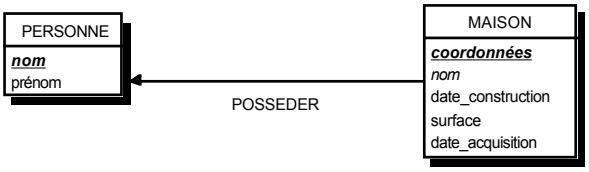
Exemple 1 :



Entité-Relation

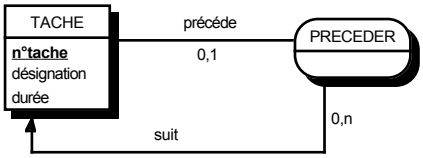


Relationnel non optimisé

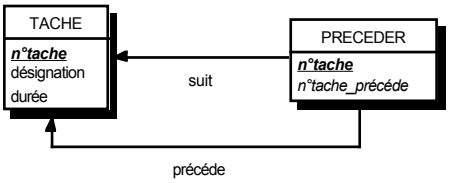


Relationnel optimisé

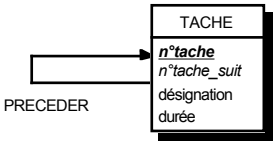
Exemple 2 :



Entité-Relation



Relationnel non optimisé

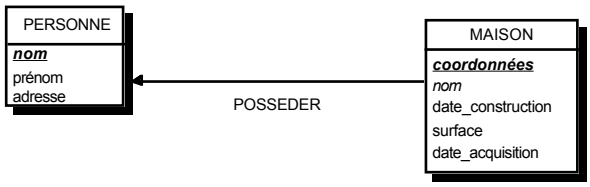


Relationnel optimisé

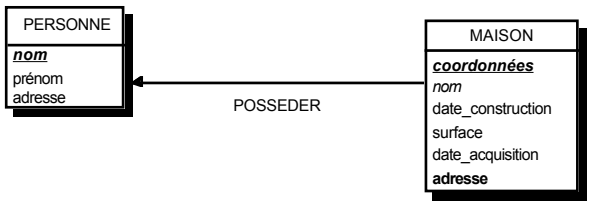
Figure 14.16 : Exemples de regroupement de tables.

Création de redondances d'attributs non-clé

Lorsque la consultation d'un ou plusieurs attributs d'une table relationnelle A à partir d'une autre table B est importante et génère de nombreux accès ou jointures, il peut être intéressant de dupliquer, sous certaines conditions, ces attributs sollicités dans B. Cela revient à une dénormalisation.



Relationnel non optimisé



Relationnel optimisé

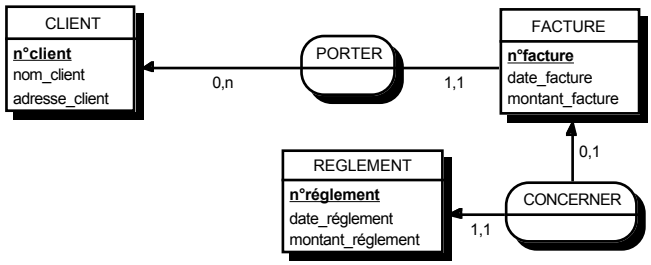
Figure 14.17 : Création de redondance d'attribut non-clé.

Dans l'exemple de la figure 14.17, l'optimisation sera justifiée pour une consultation importante de l'attribut "adresse" à partir de la table Maison. Le

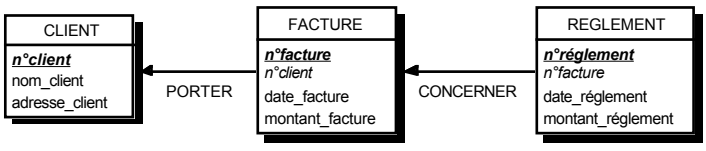
prix de la duplication de cet attribut est une augmentation du volume à mémoriser pour la table accueillant la redondance d'attributs = Σ taille des attributs migrés **x** nombre de tuples de la table accueillant la duplication. De plus si la taille des attributs dupliqués est importante, il peut y avoir augmentation du coût de certaines opérations sur la table accueillant la redondance. Il faut aussi prévoir la gestion des valeurs nulles sur les attributs dupliqués et d'une façon générale les conséquences de cette dénormalisation.

Ajout de clé étrangère redondante

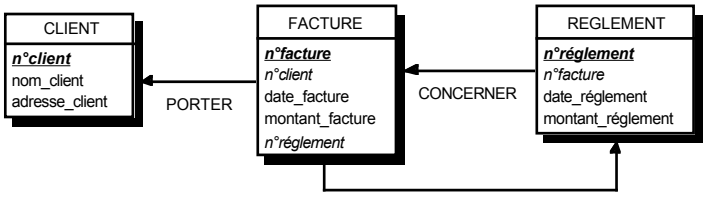
Cette optimisation ne concerne que les liens relationnels un vers un. Lorsque la consultation d'une table relationnelle A à partir d'une autre table B dans laquelle la clé de A est clé étrangère, on peut rajouter dans la table A la clé de B devenant ainsi clé étrangère. Cette opération d'optimisation revient à une dénormalisation dont les conséquences devront être estimées. La contre partie est une augmentation du volume à mémoriser pour la table accueillant la nouvelle clé étrangère = Σ taille des attributs de cette clé **x** nombre de tuples de la table accueillant la migration. Si la taille de la nouvelle clé étrangère est importante, il peut y avoir augmentation du coût de certaines opérations sur la table l'accueillant ; selon les cas, il faudra prévoir la gestion des valeurs nulles sur les attributs dupliqués et d'une façon générale les conséquences de cette dénormalisation. La figure 14.18 présente des exemples d'ajouts de clé étrangère.



Entité-Relation



Relationnel non optimisé

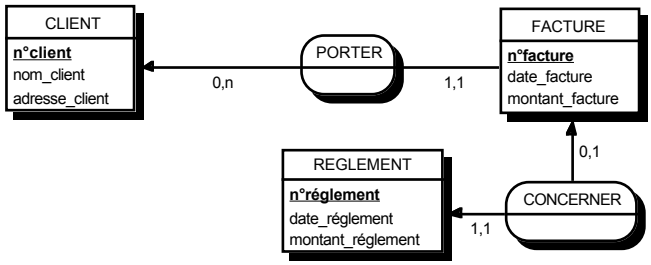


Relationnel optimisé

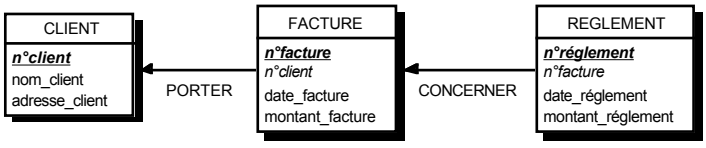
Figure 14.18 : Exemple d'optimisation par ajouts de clés étrangères redondantes.

Création de transitivité

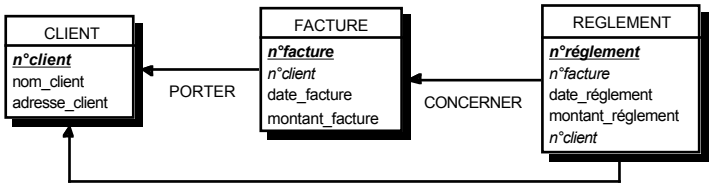
Cette optimisation concerne les liens relationnels un vers un comme les liens un vers plusieurs. Considérons la consultation d'une table C à partir d'une autre table A se fait via une table B, sachant qu'un tuple de C n'est en relation qu'avec un tuple de A. On peut alors rajouter dans la table C la clé de A devenant ainsi clé étrangère. Cette opération d'optimisation revient à créer une nouvelle dépendance transitive, donc à une dénormalisation dont les conséquences devront être estimées. La contre partie est une augmentation du volume à mémoriser pour la table C accueillant la nouvelle clé étrangère = **Σ** taille des attributs de cette clé x nombre de tuples de C. De plus si la taille de la nouvelle clé étrangère est importante, il peut y avoir augmentation du coût de certaines opérations sur la table C ; selon les cas, il faudra prévoir dans C la gestion des valeurs nulles sur les attributs de la clé étrangère et d'une façon générale les conséquences de cette dénormalisation. La figure 14.19 donne un exemple de cette optimisation.



Entité-Relation



Relationnel non optimisé



Relationnel optimisé

Figure 14.19 : Exemple d'optimisation par création de transitivité.

Création de tables de jointures

Cette optimisation a pour objectif d'optimiser le temps de réponse de la base de données par rapport à sa charge de travail. Le principe est simple, il s'agit de réduire le temps d'exécution de requêtes en stockant physiquement les relations résultantes des jointures les plus fréquemment mises en œuvre par ces requêtes.

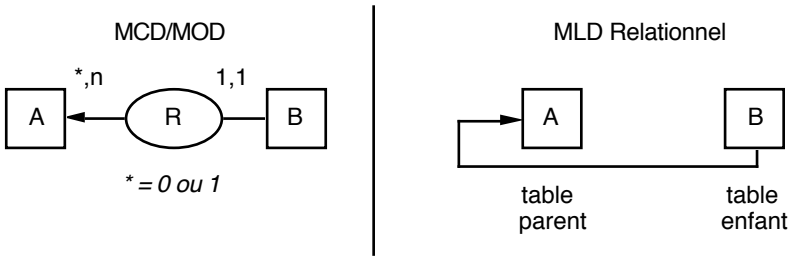


Figure 14.20 : MCD/MOD et MLD relationnel associé.

Soient deux tables A et B, A étant la table parent et B la table enfant. Dans le remplacement de la jointure AXB par une table de jointure, on considérera deux cas selon la cardinalité minimale caractérisant le lien entre A et B (notée * dans la figure 14.20).

• *cardinalité mini * = 1 (ou >1)*

On élimine la table parent A et la table enfant B ; on crée la table de jointure AXB ; les contraintes référentielles entre A et B sont transformées en contraintes de dépendance fonctionnelles sur les attributs de la table de jointure AXB

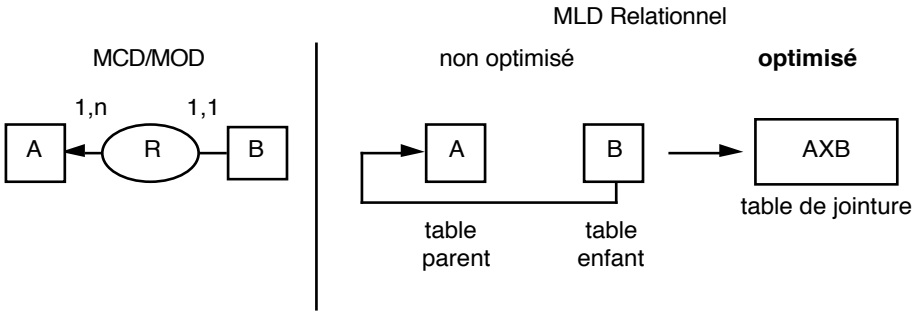
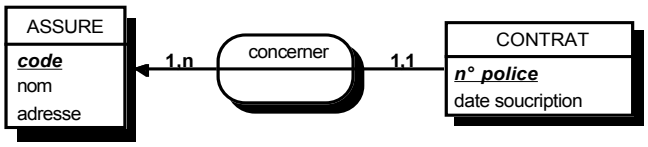
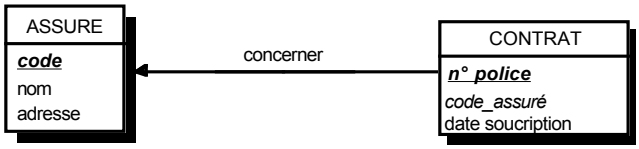


Figure 14.21 : Création de table de jointure (cardinalité mini=1).

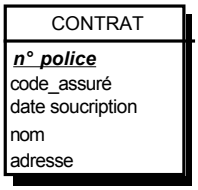
La contre partie de la réduction du coût des jointures AXB est une augmentation du volume à mémoriser pour la table AXB (plus de redondance) ; table plus grosse, plus lourde à exploiter. Si la taille des attributs dupliqués est importante, il peut y avoir augmentation du coût de certaines opérations sur la table de jointure ; selon les cas, il faudra prévoir d'une façon générale les conséquences de cette dénormalisation (mises à jour sur les attributs de A). L'exemple de la figure 14.22 illustre une telle optimisation :



Entité-Relation



Relationnel non optimisé



Relationnel optimisé (CONTRAT = table de jointure)

Figure 14.22 : Exemple d'optimisation par création de table de jointure (cardinalité mini=1).

Ce cas est très fréquent, surtout lorsque la table parent ne contient qu'une clé primaire.

• *cardinalité mini * = 0*

La table parent A doit être conservée et la table enfant B éliminée. On crée la table de jointure AXB en éliminant, éventuellement, les attributs non utilisés

par les requêtes utilisant cette jointure. On transfère les contraintes référentielles entre A et B dans la table de jointure AXB et on introduit les contraintes sur les dépendances fonctionnelles.

Sur l'exemple de la figure 14.23, la réduction du coût des jointures AXB a pour contre partie une augmentation du volume à mémoriser pour la table AXB (plus de redondance). De plus si la taille des attributs dupliqués est importante, il peut y avoir augmentation du coût de certaines opérations sur la table de jointure ; selon les cas, il faudra prévoir d'une façon générale les conséquences de cette dénormalisation (mises à jour sur les attributs de A qui se retrouvent dans A et AXB). On notera qu'il y a disparition de la contrainte référentielle.

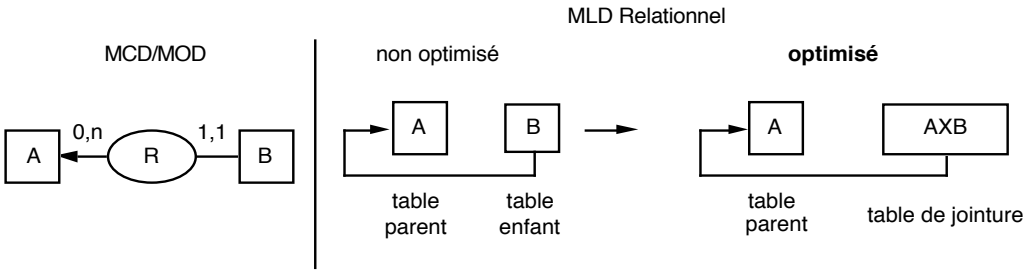
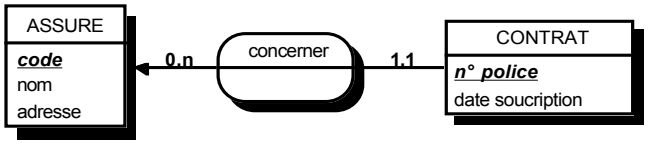
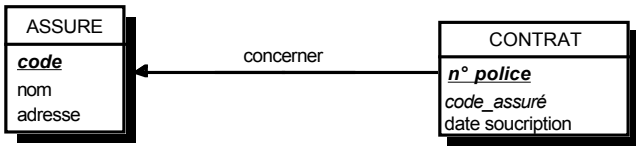


Figure 14.23 : Création de table de jointure (cardinalité mini=0)

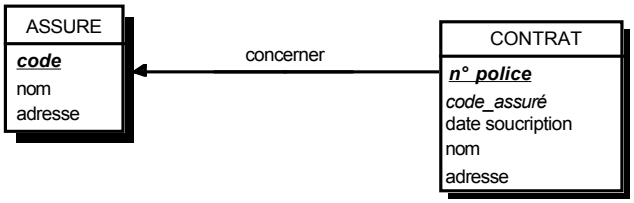
L'exemple de la figure 14.24, semblable à l'exemple précédent mais avec une cardinalité (0,n), illustre une telle optimisation :



Entité-Relation



Relationnel non optimisé



Relationnel optimisé (CONTRAT = table de jointure)

Figure 14.24 : Exemple d'optimisation par création de table de jointure (cardinalité mini=0).

Assistance à l'optimiseur de requêtes

L'optimisation des requêtes dans le relationnel est un point extrêmement sensible. Cela pour deux raisons principales : la forte indépendance logique/physique et la nature assertionnelle des langages de manipulation de données.

La forte indépendance logique/physique fait que l'utilisateur voit et manipule des tuples et des relations (description logique des informations) sans savoir comment se fait le stockage sur mémoire secondaire (physique).

La nature assertionnelle des langages de manipulation de données permet dans l'expression d'une requête de définir le *quoi* pas le *comment*. Cela conduit à un ordre des spécifications indifférent et à de nombreuses formulations possibles pour une même requête.

Il est alors nécessaire à tout SGBD relationnel d'assurer le passage d'une représentation relationnelle à des structures de stockage, de transformer la requête assertionnelle en une séquence d'opérations à effectuer sur les données et enfin de choisir la séquence d'opérations optimale (plan d'exécution optimal). Pour réaliser cela, la plupart des SGBD relationnels possèdent des optimiseurs de requêtes qui ont pour rôle :

- de minimiser les entrées/sorties disques ;
- de minimiser le temps CPU ;
- d'exploiter au maximum les index secondaires (requêtes mono-table) ;
- d'établir une stratégie lors de requêtes multitables : choisir un ordre d'un ordre des joints ; choisir comment faire des jointures ; utiliser de façon judicieuse les index secondaires.

Pour cela les optimiseurs de requêtes utilisent des statistiques sur l'état des tables. Ces informations statistiques sur l'état des tables, les valeurs d'attributs... sont rangées dans une métabase qui sera consultée par l'optimiseur avant optimisation. Le rafraîchissement de ces informations est coûteux et doit parfois être étudié très sérieusement afin d'être pris en compte comme une consigne d'exploitation à part entière. Dans certains SGBD, ce rafraîchissement peut être programmé dans l'applicatif (pouvant être développé en langage de troisième ou de quatrième génération) par le programmeur.

Quatrième Partie

Mise en oeuvre de la méthode Merise

Partie 4
Mise en oeuvre de la méthode Merise

Préambule

La deuxième partie traitait de la conception du Système d'Information Organisationnel. Cette troisième partie est consacrée à l'étude du Système d'Information Informatisé (SII), plus précisément à l'articulation des modélisations et formalismes associés. Cette partie précisera comment élaborer et exprimer les différents modèles, comment passer d'un niveau d'abstraction au suivant et transformer les différents modèles et enfin, comment aborder toute optimisation :

Cette partie est consacrée à la mise en œuvre de la méthode Merise qui passe par le suivi d'une démarche et l'organisation de moyens. Le rôle et la nécessité de ces composantes ont déjà été exposés au chapitre 3. Leur importance nécessiterait des développements détaillés et commentés.

Les précédentes éditions accordaient presque 200 pages à la présentation détaillée de ces thèmes. Les contraintes de pagination de cette nouvelle édition, ainsi que la volonté d'une mise en avant des raisonnements spécifiques à Merise, nous ont conduit :

- à condenser cette partie en n'en présentant que les aspects principaux ;
- à reporter, sous la forme de fichiers (format PDF d'Acrobat Reader) dans le CD joint, le contenu intégral des éditions initiales. Les lecteurs intéressés par des considérations opérationnelles de mise en œuvre retrouveront ainsi matière suffisante.

Des démarches

Une démarche, dans le domaine de l'ingénierie des systèmes d'information, propose une séquence structurée de tâches guidant la mise en œuvre des différents raisonnements applicables pour l'informatisation d'un système d'information organisationnel.

Si une démarche est indispensable, il n'y a pas pour autant de démarche unique, car il n'y a pas de réponse unique à des contextes variés. Les méthodes n'ont certes pas inventé ou découvert ce qui constitue les activités de conception et de développement des projets informatiques, ni la succession des étapes à franchir. Mais tous les « méthodologues » s'accordent sur l'importance de définir précisément, dans une méthode, ces activités et ces étapes ainsi que sur

l'importance de l'unicité de méthode pour un organisme.

Influencées par les environnements techniques, économiques et culturels, les démarches ont évolué et offrent aujourd'hui une diversité pouvant satis- faire une large palette de situations.

Cependant, pour être praticable, une démarche doit également pouvoir s'adapter d'une part à la taille et à la complexité du projet abordé, d'autre part au contexte et aux contraintes de l'entreprise. Cette personnalisation s'avère à chaque fois nécessaire.

Évolution des démarches

Première génération

Lors de la première génération de la méthode Merise, dans les années quatre-vingt, la démarche proposée reposait sur les hypothèses explicites ou implicites suivantes :

- des projets relativement ambitieux, correspondant au développement des grands systèmes d'information « intégrés » ;
- une technologie dominée par les grands systèmes transactionnels avec des bases de données centralisées ;
- une relative aisance budgétaire accordée à l'informatique ; – un horizon perçu comme suffisamment stable pour permettre une planification de projets à moyen terme. La démarche préconisée s'inspire alors de celle mise en oeuvre en génie civil, avec parfois un esprit « grandes manœuvres » : organisation, planification, points de décision, formalisation et structuration des tâches. Cette démarche est présentée au chapitre 15.

Deuxième génération

Dans les années quatre-vingt-dix, l'environnement tant technique qu'économique évolue fortement avec les caractères majeurs suivants :

- une technologie où le micro-ordinateur et son interface graphique est le poste de travail privilégié de l'utilisateur, devenu également plus exigeant ;
- un éclatement et une répartition des systèmes (émergence du client/serveur) ;
- une crise économique restreignant les budgets et exigeant désormais une meilleure productivité des développements informatiques ;
- une plus grande incertitude sur l'avenir favorisant les projets à court terme. Dans ce contexte, la démarche « classique » de la première

génération a été perçue (parfois hâtivement) comme lourde, rigide et n'offrant pas de visibilité suffi- sante pour la conduite de projet (effet tunnel).

Une nouvelle démarche, ***dite rapide***, a été proposée en alternative à la démarche classique. Cette démarche rapide remplace la « cascade » de tâches par une conception itérative faisant majoritairement appel au maquettage/prototypage et à une forte implication des utilisateurs. Cette démarche rapide relève plutôt d'un es- prit « commando ». Cette démarche est présentée au chapitre 16.

Dans cette deuxième génération de la méthode Merise, nous proposons une cohabitation des deux démarches qui, à l'expérience, correspondent à des situations de projets très différentes. Ainsi, suivant la nature et le contexte du projet envisagé, le concepteur optera-t-il pour l'une ou l'autre démarche. Dans les deux cas, il conservera les raisonnements du cycle d'abstraction.

Rappelons également que d'autres démarches peuvent aisément être associées aux raisonnements de Merise. Citons par exemple des « mariages » réussis avec SDM/S et Eurométhod.

Les démarches préconisées par la méthode Merise ne sont pas une obligation mais une proposition, surtout en absence d'alternative.

Personnalisation des démarches

La démarche est un guide général pour dérouler un projet ; elle ne doit pas être un carcan.

Par principe, une démarche-type se doit de présenter l'exhaustivité des tâches nécessaires, tous projets confondus. Or chaque projet présente des caractéristiques propres (nature, taille, contexte technique, enjeux, etc.) qui rendent nécessaire un adaptation de la démarche. Par exemples :

- en étude préalable, la tâche d'évaluation des solutions techniques sera développée si le projet vise à déterminer l'architecture la plus adaptée, mais a contrario sera réduite si la configuration est figée d'avance ;
- en étude préalable, l'analyse de l'existant peut être abrégée lorsque aucun problème particulier n'est prévisible, mais devra être approfondie lorsque la situation actuelle s'avère complexe ou confuse ;
- en étude détaillée, l'étude des procédures de secours et des procédures de transition est fortement conditionnée par le contexte de chaque projet.

Avant d'engager une étape, il est donc indispensable que le responsable de projet réfléchisse à la personnalisation de sa démarche pour le projet. Il pourra par exemple s'inspirer de la technique suivante :

Personnalisation pour le projet Z

Étude préalable - démarche-type générale	Ignorer	Réduire	Normal	Développer
Lancement			X	
Analyse de l'existant		X		
Analyse des flux			X	
Données actuelles		X		
Informatisation actuelle	X			
Organisation actuelle		X		
Bilan		X		
Besoins à satisfaire			X	
Conception de solutions			X	
Objectifs et contraintes				X
MCD futur			X	
… etc				

Des moyens

La conception puis la réalisation d'un logiciel, support d'un système d'information constituent une suite de processus complexes qui font intervenir un nombre important d'acteurs et de partenaires qui doivent s'organiser et disposer d'outils adaptés.

Une organisation

Coordonner la diversité des savoir-faire pour respecter les engagements et améliorer la qualité, prévoir, organiser et suivre les ressources et les activités, voilà l'objectif de la maîtrise d'un projet.

Dans ce domaine, la méthode Merise se borne à proposer des principes généraux qui doivent être développés et adaptés au contexte de chaque organisme. Ces principes devront être en cohérence d'une part avec la méthode de conduite de projets adoptée, d'autre part avec les règles et recommandations d'assurance qua- lité en vigueur dans l'entreprise.

Ces principes généraux d'organisation se traduiront essentiellement par une structure type de projet articulée autour des groupes de projet, de pilotage et de validation. Ils seront présents, ainsi que les répartitions des tâches au chapitre 17.

Des outils

Si dans la première période de Merise, on a pu ou dû se satisfaire de modélisations manuelles, aujourd'hui, la mise en œuvre d'une méthode passe nécessairement par l'utilisation d'outils logiciels adaptés. On peut même affirmer que l'outil est le partenaire indispensable d'une méthode et interagit sur les évolutions et le devenir d'une méthode.

Dans le cas de la méthode Merise, le marché offre et a offert une diversité de produits qui, bien que différents, proposent un ensemble de fonctionnalités standards nécessaires à la mise en œuvre de la méthode Merise. Les principes de ces outils et l'illustration au travers de Win'Design font l'objet du chapitre 18.

Une adaptation au BPR

Les formalismes de modélisation proposés par la méthode Merise sont avant tout dédiés à la conception de systèmes d'information, qui s'inscrivent dans les disciplines de la gestion.

Dans les années quatre-vingt-dix, ce monde de la gestion a été traversé par le courant managérial du Business Process Reengineering qui proposait de nouvelles manières de se structurer et de s'organiser, accordant une place centrale aux technologies de l'information et aux systèmes d'information.

Il était donc tentant de voir si, avec des finalités différentes, les outils de modélisation utilisés dans la méthode Merise, trouvaient un usage pertinent dans le domaine du BPR. Cette adaptation, qui ouvre des perspectives très stimulantes, est présentée au chapitre 19.

15

La démarche classique de Merise

Principes généraux

Découpage en étapes

La démarche classique de Merise couvre trois grandes périodes : la planification des systèmes d'information, le développement d'un projet et la maintenance de l'application. Chacune de ces périodes se décompose en étapes successives comme l'illustre la figure 15.1:

Cette démarche n'est pas spécifique à la méthode Merise. Il faut cependant remarquer la part importante faite à la conception (étude préalable, étude détaillée, étude technique); en particulier les passages successifs de l'étude qui permettent de dégrossir progressivement les différents choix que l'on a déjà mis en évidence dans la courbe dite du soleil (voir chapitre 3).

A chacune des étapes de la démarche, les concepteurs appliqueront les différents raisonnements nécessaires à la construction du système d'information, parcourant les niveaux d'abstraction de la méthode. C'est le couplage démarche - niveaux d'abstraction qui fait la spécificité et la richesse de la méthode Merise. Ce couplage ne sera pas aussi étroit tout au long de la démarche. La méthode Merise s'est positionné avant tout comme une méthode d'ingénierie de systèmes d'information et naturellement nous développerons les aspects liés à ces étapes tout en rappelant que Merise est également une structure d'accueil pour d'autres raisonnements développés en dehors de la méthode, en particulier dans le cadre de la réalisation, avec les apports du génie logiciel

Dans la pratique, ce sont les étapes de conception d'un projet (de l'étude préalable à l'étude technique) qui sont les plus connues et utilisées, essentiellement à cause de l'efficacité des modélisations mises en œuvre.

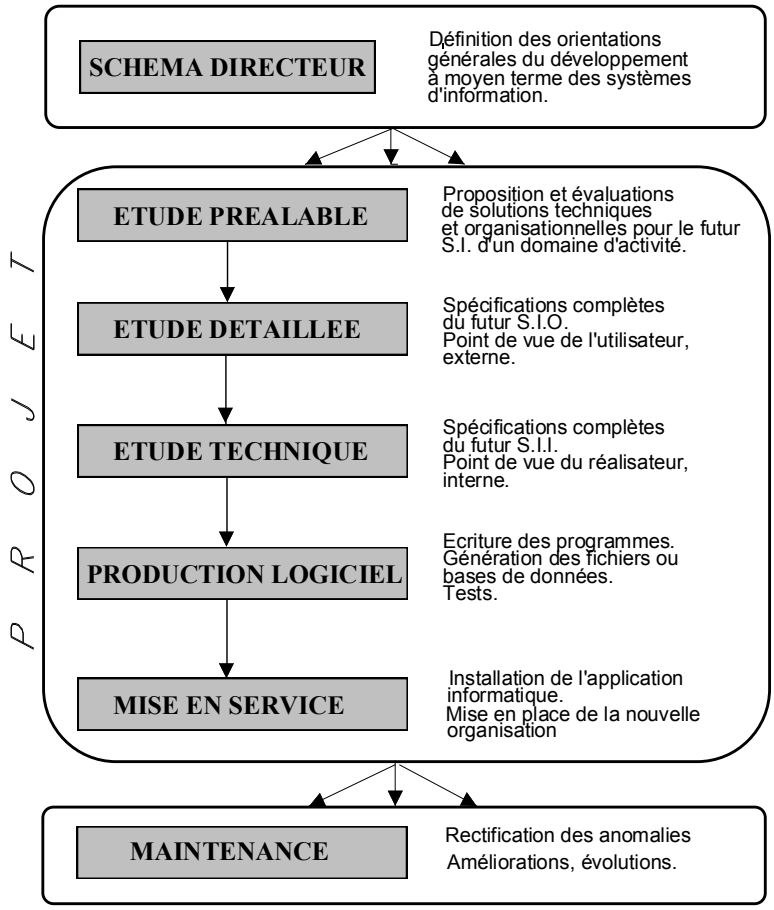


Figure 15.1 : Les étapes de la démarche classique Merise.

Portée des étapes

La portée des différentes étapes pour un système d'information est la suivante :

- le schéma directeur concerne l'entreprise (ou un secteur majeur) dans son ensemble,
- l'étude préalable porte sur un domaine, sous-système d'information de l'entreprise défini lors du schéma directeur,
- l'étude détaillée, l'étude technique, la production du logiciel, la mise en service et la maintenance portent sur des projets relatifs à un domaine spécifique et peuvent être découpés en applications.

L'étude des systèmes d'information s'effectue à partir d'un découpage initié dans le ***schéma directeur*** identifiant les différents domaines d'activité. C'est à partir de ce découpage que sont déterminés et planifiés les différents projets associés à l'informatisation des systèmes d'information.

L'étude préalable, consacrée à l'élaboration et la proposition de solutions,

s'effectue pour un projet. L'étude préalable peut à son tour conclure à un découpage en plusieurs sous-projets, pour des raisons de taille ou de complexité.

L'étude détaillée, l'étude technique, la production du logiciel et la mise en service s'effectuent par projet ou sous-projet. Eventuellement, à l'intérieur de chacune de ces étapes, le projet ou sous-projet peut être découpé en plusieurs lots, si cela s'avère nécessaire pour des raisons soit de complexité du champ de l'étude, soit pour des impératifs de suivi ou d'organisation.

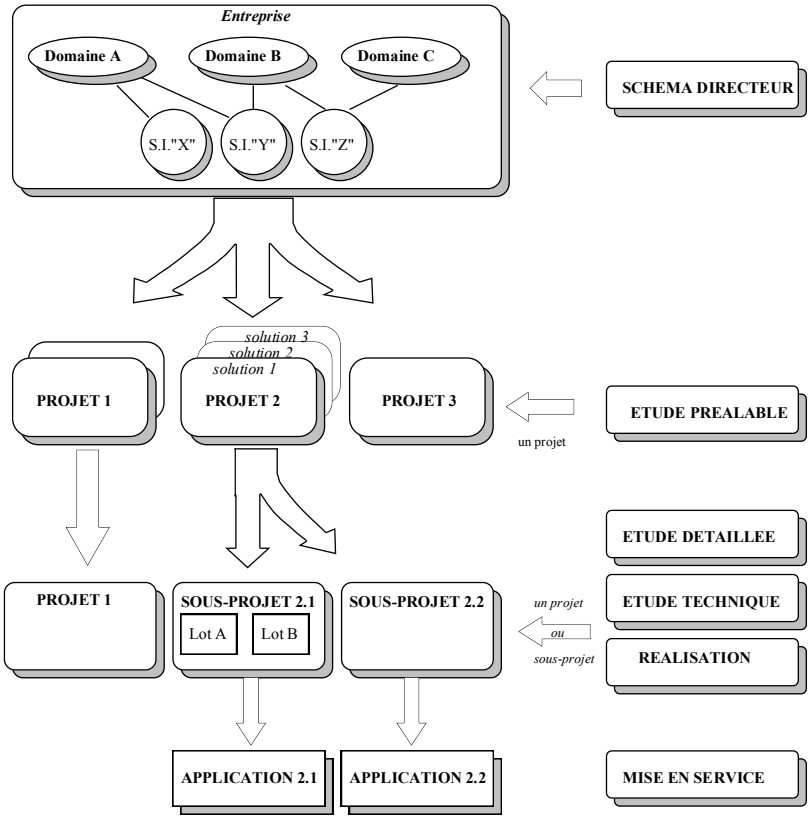


Figure 15.2 Découpage et portée des différentes étapes de la démarche classique de Merise.

Dans ce chapitre, nous focaliserons la mise en œuvre de la démarche sur la période de développement du projet, et en particulier sur les étapes de conception. Les aspects de l'application de la méthode Merise au schéma directeur et à la maintenance, relativement marginalisés dans la pratique, sont exposés dans les annexes correspondantes enregistrées sur le CD.

L'étude préalable

Objectifs de l'étude préalable

S'inspirant des démarches de l'ingénierie, la conception d'un système d'information nécessite, avant la spécification exhaustive d'un projet, une étape plus sommaire et globale, proposant des choix et les évaluant: l'étude préalable.

L'étude préalable a pour objectifs :

- L'analyse et l'évaluation critique du fonctionnement du système d'information actuel.
- L'élaboration de solutions en précisant :
  - les processus de fonctionnement du domaine,
  - la perception des informations,
  - les modes d'organisation,
  - le degré et le type d'automatisation.
- L'évaluation des solutions proposées en termes de :
  - équipements informatiques,
  - coûts et délais de mise en œuvre,
  - conséquences sur l'organisation générale de l'entreprise,
  - scénario de mise en œuvre.

L'étude préalable privilégie la diversité des solutions par rapport au détail des spécifications.

Le sous-ensemble représentatif

La conduite d'une étude préalable se trouve confrontée à deux exigences contradictoires. D'une part, elle doit être de courte durée conditionnée par un budget souvent limité, d'autre part, elle doit conduire à une analyse suffisamment complète pour éclairer les décisions du maître d'ouvrage engageant toute la suite du projet.

Disposant d'un temps total de travail limité, l'équipe de projet se pose la question « Quel est le sous-ensemble qui représente le mieux le domaine étudié ? ».

Pour choisir ce sous-ensemble représentatif et mettre en évidence les processus majeurs du domaine, le concepteur prend en compte le fonctionnement normal, en négligeant les situations exceptionnelles, les processus de faible fréquence et les opérations marginales. Ce sous-ensemble doit représenter l'ossature du système d'information où les activités principales et les données

fondamentales sont exprimées.

La conception du futur système d'information repose sur les différentes solutions élaborées autour des éléments constituant le sous-ensemble représentatif. Le choix des éléments constituant le sous-ensemble représentatif conditionne donc fortement l'orientation future du fonctionnement du système d'information.

Les phases de l'étude préalable

L'étude préalable se décompose en quatre phases présentées en Figure 15.3

Les objectifs de la phase d'analyse de l'existant sont de comprendre et formaliser le fonctionnement du système actuel, et diagnostiquer ses dysfonctionnements sur les plans de la gestion, de l'organisation et des solutions techniques.

Les objectifs de la phase de conception sont d'élaborer et formaliser des solutions de fonctionnement du futur système d'information.

Les objectifs de la phase d'évaluation sont d'évaluer chacune des solutions élaborées dans la phase précédente sur les aspects fonctionnels, organisationnels, techniques, financiers, charges de développement et planning.. Lors de cette phase seront aussi proposés des scénarios de mise en service.

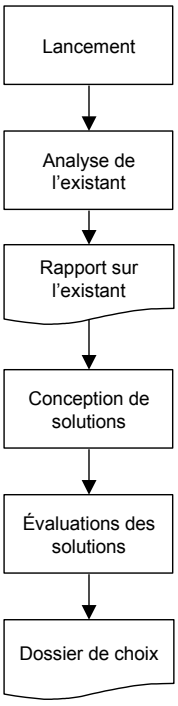


Figure 15.3 : Les quatre phases de l'étude préalable.

Chaque phase comporte un ensemble de tâches qui s'enchaînent telle une

procédure Les phases et tâches du déroulement présenté de l'étude préalable constituent la trame générale de la démarche classique de la méthode Merise. Toutefois, l'importance accordée aux tâches et à leur ordonnancement au sein de chaque phase dépend en grande partie du contexte, de la taille et du type de problème rencontré, de la connaissance que le concepteur peut avoir du domaine.

Phase de lancement

La phase de lancement initialise l'étude préalable.. Elle se décompose en quatre tâches présentées sur la figure 15.4.

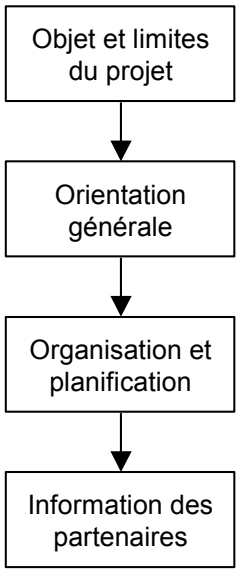


Figure 15.4 : Le lancement de l'étude préalable.

Cette phase assure d'abord un premier cadrage du projet en précisant son contour Elle rappelle les orientations du schéma directeur, éventuellement réactualisées; à défaut, le demandeur de l'étude devra rédiger ou approuver une courte note d'orientation

Cette phase définit ensuite les modalités pratiques de déroulement et d'organisation de l'étude préalable en précisant:

- L'organisation des structures du projet (groupe de pilotage, groupe de projet, groupe de validation) et la répartition des tâches
- Un calendrier
- Le cadre budgétaire.
- Les contraintes d'environnement pesant sur le projet.

Tous ces éléments seront réunis pour constituer une note de lancement.

Cette phase donne enfin lieu à une information auprès des différentes parties

prenantes sur les enjeux de cette étude préalable pour l'entreprise.

Phase d'analyse de l'existant

Avant de concevoir un nouveau système d'information, il est indispensable de bien connaître le domaine concerné Bien que cette connaissance puisse déjà être compilée dans des documentations, il est indispensable de recueillir ou réactualiser ces informations en interrogeant les gestionnaires et les utilisateurs et de procéder à une analyse critique. L'analyse de l'existant se décompose en quatre tâches présentées sur la figure 15.5.

Bien que nécessaire à la compréhension de l'activité du domaine, l'étude de l'existant ne doit pas être la tâche majeure de l'étude préalable. Il ne faut surtout pas rechercher l'exhaustivité du recensement des données et des traitements actuels, sous peine de dépassement de budget... L'étude de l'existant doit s'attacher à mettre en évidence les activités principales et informations associées, ainsi que les dysfonctionnements majeurs. Il ne faut pas transformer l'analyse en audit. Sur le terrain, bien des critiques de lourdeur attribuées à la méthode Merise le furent à cause d'analyses de l'existant complètes et systématiques, dont le bénéfice pour le futur système restait limité.

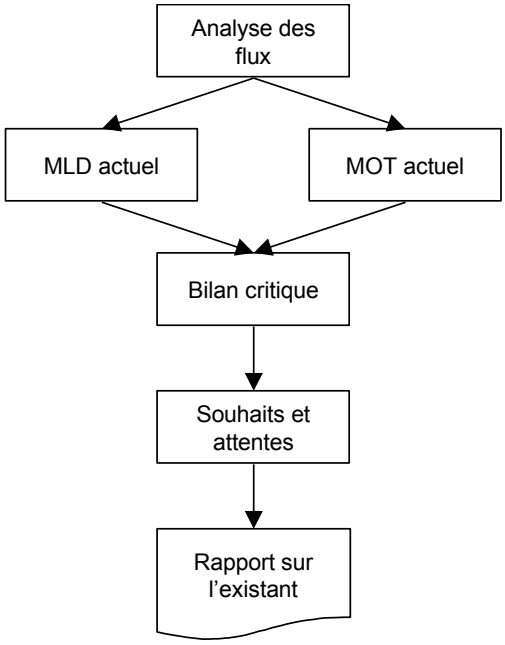


Figure 15.5 : L'analyse de l'existant dans l'étude préalable.

Analyse des flux d'informations

Objectifs et résultats attendus

Il s'agit de mettre en évidence rapidement :

- Les principales informations qui circulent dans le système d'information

de l'entreprise ; ces flux d'informations peuvent correspondre à des supports très différents (imprimés, communications verbales, téléphoniques...).

- Les principales unités actives qui participent à ces échanges ; ces acteurs peuvent appartenir au domaine d'application (poste de travail, services...) ou à son environnement (autres unités, client, fournisseur...).

Avec un formalisme rudimentaire (voir chapitre 5 « Découpage en domaines et analyse des flux »), l'analyse des flux permet de mettre en évidence :

- La délimitation du domaine.
- Les activités principales du domaine.
- La nature et la signification des informations échangées.

Raisonnements utilisés

Différentes sources peuvent servir de base à une analyse des flux d'informations :

- L'organigramme des services (acteurs) ;
- Des documents réels recueillis (flux d'informations) ;
- Des procédures prédéfinies (acteurs et flux) ;
- L'interview des gestionnaires et des utilisateurs.

Dans tous les cas, on cherchera à mettre en évidence, puis à nommer, les acteurs concernés et les flux d'informations. On représentera ensuite graphiquement le diagramme des flux correspondant. On pourra commencer par un diagramme brut qui évoluera vers un diagramme de contexte (cf. Chap. 5).

Analyse du modèle organisationnel actuel

Objectifs et résultats attendus

L'objectif est de décrire globalement le fonctionnement du système actuel en termes d'organisation. Le résultat attendu est une description succincte du modèle organisationnel des traitements actuels (MOT actuel), cela en termes de:

- Postes de travail avec environnement humain et technique.
- Procédures organisationnelles majeures.

La construction d'un modèle organisationnel actuel poursuit plusieurs objectifs. Tout d'abord servir de point de départ au processus d'abstraction permettant la construction du modèle conceptuel de traitements actuel puis futur. Ensuite établir un état des lieux du fonctionnement actuel permettant d'évaluer l'écart

avec la situation future proposée, et d'élaborer un scénario pour la transition entre les deux situations. Enfin apprécier le fonctionnement général du domaine afin d'extraire les éléments marquants à retenir pour le sous-ensemble représentatif.

Raisonnements utilisés

Le concepteur met œuvre le formalisme présenté au chapitre 8. L'élaboration de ce modèle ne s'effectue pas par la décomposition des opérations d'un modèle conceptuel, mais par la formalisation directe de l'enchaînement des tâches effectuées dans les différents postes. On illustrera en particulier les procédures que le gestionnaire ou l'utilisateur considèrent comme représentatives.

L'utilisation du modèle organisationnel des traitements (MOT) en étude préalable peut être facultative si l'aspect organisationnel est simple dans le domaine. Le diagramme des flux suffit alors pour exprimer le fonctionnement global du domaine.

Analyse du modèle logique des données actuel

Objectifs et résultats attendus

L'objectif est de connaître et de décrire les fichiers informatisés actuellement existants. Les résultats attendus sont :

- Le modèle logique de données reconstitué à partir des structures de bases de données ou fichiers existants.
- Le volume des données actuellement mémorisées et l'évolution prévisible de ces volumes.

Comme précédemment, si les données actuelles étaient à reconsidérer totalement, cette tâche pourrait devenir inutile.

Raisonnements utilisés

Quelle que soit l'expression physique des données actuelles, on s'efforcera de les modéliser sous la forme d'un MLD « relationnel ». Des outils de « reverse engineering » (cf. Chap. 18) sont particulièrement utiles pour cette tâche.

Bilan critique de la situation actuelle

Objectifs et résultats attendus

Il ne suffit pas simplement de décrire le système d'information actuel du domaine étudié; il faut porter un jugement lucide sur ce système

- Quels sont les points à préserver ? .
- Quels sont ses dysfonctionnements ?

Sur le plan technique : obsolescence ou inadéquation du matériel ; difficultés pour exploiter ou maintenir

les systèmes informatiques...

Sur le plan organisationnel : lourdeur du fonctionnement, insatisfaction grandissante des utilisateurs ou développement de circuits parallèles.

En matière de gestion : décalage important entre les choix de gestion actuellement mis en œuvre et ceux correspondant au système d'information actuellement opérationnel.

L'analyse points forts-points faibles tiendra compte des différents interlocuteurs dont les jugements peuvent être différents; l'équipe de conception peut, elle-même, apporter des éléments critiques intéressants.

Souhaits et attentes à satisfaire

Objectifs et résultats attendus

Les interviews ont été l'occasion non seulement de recueillir les avis sur la situation actuelle mais aussi d'entendre les besoins exprimés par les gestionnaires et les utilisateurs. Ces souhaits et attentes serviront de matière première à l'élaboration des solutions futures.

Raisonnements utilisés

Les besoins exprimés doivent être qualifiés, classés par nature et par thème, hiérarchisés et priorisés. Leur présentation peut être assortie de commentaires du groupe de projet, en particulier sur la compatibilité de ces souhaits et attentes avec les orientations générales du projet rappelées dans la phase de lancement.

Phase de conception de solutions

Cette phase se concrétisera par l'élaboration de plusieurs modèles du futur système d'information du domaine :

- Modèle conceptuel des données (MCD).
- Modèle conceptuel des traitements (MCT).
- Modèle organisationnel des données (MOD).
- Modèle(s) organisationnel(s) des traitements (MOT).

Mais auparavant, il est nécessaire de clarifier les orientations du futur système d'information.

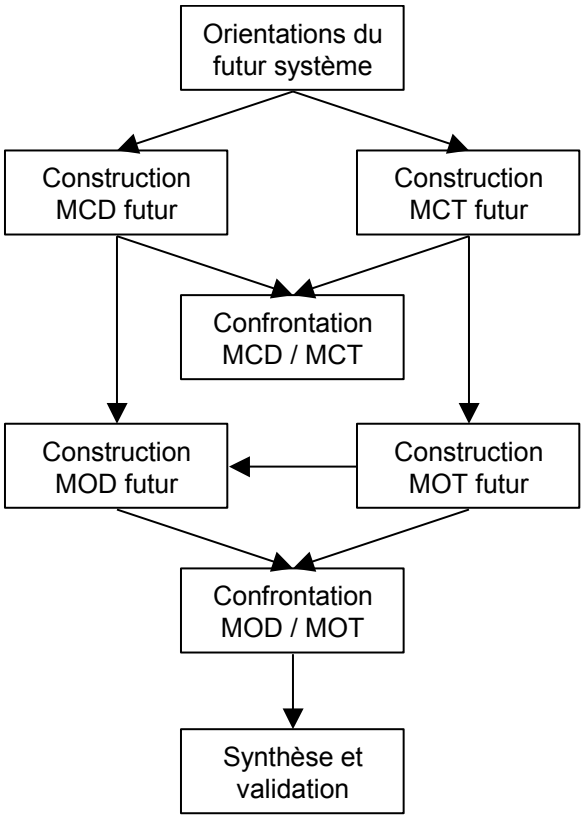


Figure 15.6 : La phase de conception de solutions de l'étude préalable

Orientations du futur système d'information

Lors du lancement, l'équipe de projet aura recueilli les orientations générales du projet, provenant du schéma directeur ou du groupe de pilotage. L'analyse de l'existant a permis de rencontrer des gestionnaires et des utilisateurs en prise avec leur système d'information actuel au quotidien. Ils ont formulé des souhaits et des attentes concernant le futur système d'information.

Ces orientations, objectifs, contraintes, souhaits, attentes constituent souvent un ensemble volumineux, hétéroclite, parfois contradictoire qu'il est nécessaire de trier et de structurer.

Pour guider efficacement toutes les étapes suivantes, les objectifs et les contraintes doivent être peu nombreux, immédiatement compréhensibles et cohérents entre eux. L'équipe de projet doit donc faire un travail de synthèse pour dégager les objectifs significatifs.

Sur cette base affermie, l'équipe de projet élaborera les différents modèles du futur système d'information, selon le schéma de la figure 15.6.

Construction du modèle conceptuel de données futur

Objectifs et résultats attendus

Le modèle conceptuel de données formalise la signification des principaux concepts et informations qui seront utilisés pour le système d'information futur.

Raisonnements utilisés

Pour construire le futur MCD, l'équipe de projet applique l'ensemble des règles de construction d'un modèle conceptuel de données.

Le modèle spécifié comporte :

- les entités avec leur identifiant et quelques propriétés ;
- les relations avec leurs éventuelles propriétés et cardinalités ;

On procède souvent à la construction directe du futur MCD, en précisant éventuellement les concepts déjà utilisés dans le passé et ceux nouveaux dans le domaine. On constate souvent une grande similitude entre MCD actuel et MCD futur pour le choix des principales entités et relations ; c'est un élément de grande stabilité dans l'entreprise.

En pratique, on aura recours à une approche plutôt inductive s'appuyant d'une part sur des structures issues des données actuelles, d'autre part sur les concepts métier évoqués par les gestionnaires et les utilisateurs dans leurs discours, leurs écrits ou sur des documents circulant dans l'entreprise, et recueillis lors des interviews

En étude préalable, un MCD constitue davantage une structure qu'un descriptif : la quasi-totalité des entités et relations modélisables dans le domaine est exprimée ; par contre, les propriétés les caractérisant peuvent être très limitées, elles servent surtout à stimuler la compréhension du modèle par les lecteurs.

Construction du modèle conceptuel de traitements futur

Objectifs et résultats attendus

L'objectif visé est de spécifier le fonctionnement général du domaine conformément aux nouvelles orientations de gestion et sans référence explicite à l'organisation des ressources qui seront utilisées.

Le résultat attendu est un modèle conceptuel de traitements comprenant :

- La liste des acteurs ainsi que les événements émis et les résultats reçus.
- La liste des processus et les événements déclencheurs.
- Le diagramme d'enchaînement des opérations dans le nouveau système.
- La description succincte du contenu de chaque opération.

En étude préalable, le MCT futur permet surtout de confirmer les limites du domaine en exprimant formellement les fonctions remplies par celui-ci, puis de

servir de point de départ à l'élaboration de solutions organisationnelles.

Raisonnements utilisés

Le modèle conceptuel de traitements se spécifie conformément aux règles du formalisme.

Un modèle conceptuel de traitements conçu dans une étude préalable met en évidence les processus majeurs (ceux qui sont retenus comme représentatifs), décrivant les opérations et leurs enchaînements. Le concepteur porte alors plus d'attention à l'ordonnancement des opérations qu'à la description détaillée du contenu des messages et des opérations.

Confrontation MCD / MCT

Objectifs et résultats attendus

L'équipe de projet doit régulièrement s'assurer que les modélisations des données et des traitements effectuées respectent une cohérence mutuelle. En particulier, on vérifie la convergence des deux modèles sur les concepts retenus comme représentatifs dans le sous-ensemble d'étude.

Raisonnements utilisés

On met en oeuvre la technique de relecture croisée présentée au chapitre 11.

Construction de modèles organisationnels de traitements

Objectifs et résultats attendus

En étude préalable, le modèle organisationnel des traitements joue un rôle fondamental. Nous avons vu précédemment que le niveau conceptuel exprime un caractère général et de stabilité. Le niveau organisationnel permet d'exprimer la variété des solutions proposées, conformément aux orientations d'organisation. Pour chaque solution d'organisation envisagée, le concepteur produit :

- Une définition des postes avec leurs ressources humaines et techniques.
- La description des procédures organisationnelles présentant l'enchaînement des principales tâches représentatives de l'activité du domaine, en indiquant leur fréquence d'utilisation.
- L'illustration éventuelle de quelques tâches automatisées considérées comme critiques à l'aide d'une maquette.

L'élaboration des modèles organisationnels de traitements est une des principales difficultés de l'étude préalable: la description des différentes tâches doit être suffisamment explicite pour que l'utilisateur puisse étayer son choix, sans pour cela exiger un travail trop détaillé et trop complexe.

Raisonnements utilisés

Chaque proposition de modèle organisationnel de traitements est exprimée conformément aux règles du formalisme. Le concepteur s'efforcera d'élaborer différentes variantes en jouant sur :

- la diversité des postes et leur environnement informatique,
- le degré d'automatisation,
- les enchaînements entre tâches.

Construction du modèle organisationnel de données

Objectifs et résultats attendus

Répercuter sur le MCD les choix d'organisation spécifiés dans les MOT et préciser ainsi la future organisation des données à mémoriser.

Raisonnements utilisés

S'exprimant dans le même formalisme que le MCD, le MOD en apparaît plus comme le prolongement que comme un nouveau modèle.

La production d'un MOD n'est pratiquement nécessaire qu'en cas de répartition organisationnelle (cf. Chap. 10).

Confrontation MOD/MOT

Objectifs et résultats attendus

Il est important de s'assurer que les modèles précédemment élaborés, MOD et MOT, peuvent fonctionner ensemble, constituant un système d'information cohérent. On vérifie globalement que les traitements disposent bien des données nécessaires (entités, relations) et que les données modélisées ont toutes une utilité dans les traitements. En étude préalable, le MOD se différenciant très peu du MCD, on effectue fréquemment cette confrontation entre le MCD et le MOT.

Cette confrontation permet d'obtenir une liste de diagnostics d'anomalies concernant le MCD-MOD et le MOT. En pratique, cette cohérence est réalisée en continu, à l'issue de la modélisation de chaque procédure organisationnelle.

Raisonnements utilisés

En étude préalable, on ne connaît pas en général la totalité des propriétés. La technique à utiliser est donc celle de la grille de cohérence globale présentée au chapitre 11.

Cette technique permet à l'équipe de projet de repérer elle-même et de corriger de nombreux problèmes. Il est donc conseillé de procéder à ces contrôles avant de soumettre les différents modèles à la validation des gestionnaires et des utilisateurs

Synthèse et validation des solutions proposées

Objectifs et résultats attendus

Pour valider la phase de conception des solutions, il est nécessaire de pouvoir présenter le futur système d'information aux différentes parties prenantes du projet. Cette validation devra mettre l'accent sur les choix retenus dans les différents modèles, les caractères distinctifs entre les différentes variantes et les choix appelant décision de la part des gestionnaires et des utilisateurs.

L'ensemble des modèles traduisant les solutions proposées pour le futur système d'information est exprimé dans un formalisme certes rigoureux et concis, mais pouvant parfois apparaître comme trop abstrait. L'équipe de projet devra donc s'efforcer d'en faire une présentation compréhensible et commentée, en particulier pour le MCD.

Phase d'évaluation et d'appréciation des solutions

Cette phase se décompose en sept tâches présentées à la figure 15.7.

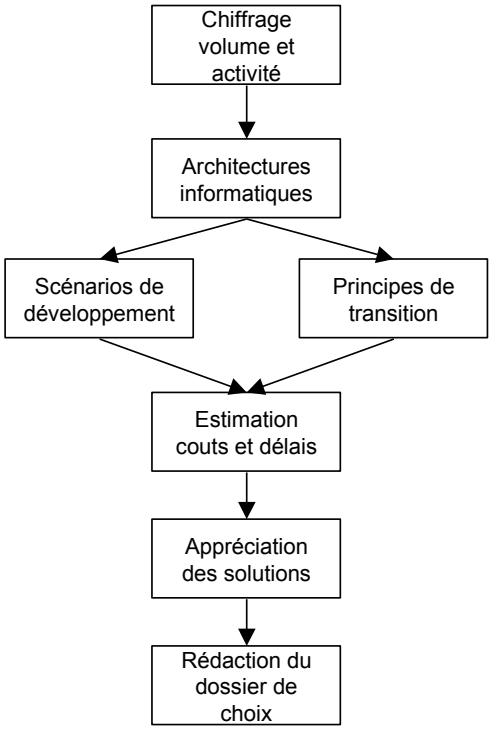


Figure 15.7 : La phase d'évaluation et d'appréciation des solutions de l'étude préalable.

Chiffrage du volume et de l'activité

Objectifs et résultats attendus

Cette tâche consiste à :

- Estimer le volume global des futures données à mémoriser.

- Evaluer la future la charge du système informatique directement liée à l'accès aux données à partir des traitements

Ce chiffrage, évidemment sommaire, servira à dimensionner les ressources informatiques proposées dans la tâche suivante

Raisonnements utilisés

Pour l'estimation du volume des données, le concepteur travaille sur le MOD et applique la démarche de quantification (cf. Chap. 10). Il peut éventuellement effectuer ce chiffrage sur le MLD brut, après transformation. L'ensemble des propriétés n'étant pas défini, le concepteur pourra procéder à une estimation globale de la taille des entités et des relations.

Pour l'estimation des accès, le concepteur pourra s'appuyer sur les actions issues des grilles de cohérence de la confrontation MOD / MOT de la phase précédente. Il prêtera une attention particulière aux tâches du MOT à fréquence élevée ou à temps de réponse exigeant ainsi qu'aux tâches à volume de données important.

Architectures informatiques

Objectifs et résultats attendus

Dans cette tâche, le concepteur définit l'architecture du système informatique qui supportera le futur système d'information.

Chaque solution s'exprime, entre autre, par :

- Le type d'architecture envisagée (centralisée, client-serveur, intranet, ..)
- Le nombre, type et répartition des postes utilisateurs (clients).
- Le nombre, type et répartition des serveurs.
- Les capacités de mémorisation des données.
- Le dimensionnement des réseaux de télécommunication.
- Les logiciels de base utilisés (système d'exploitation, SGBD, langages, ..).

L'importance et les résultats attendus de cette tâche varieront selon qu'il s'agit de d'un système existant (éventuellement à étendre) ou d'un nouveau système.

Raisonnements utilisés

Le problème abordé n'implique pas directement des raisonnements propres à la méthode Merise. Il fait appel à des connaissances plus informatiques sur les possibilités des différents matériels. On pourra se référer à des éléments méthodologiques spécifiques (méthode TACT [Alakl & Lalanne 89]).

Principes de transition du système actuel au système futur

Objectifs et résultats attendus

Cette tâche consiste à définir le processus de transition de systèmes lors de la mise en service. Le concepteur doit clairement définir la manière dont s'effectuera le passage entre le fonctionnement actuel et le fonctionnement futur, surtout si l'organisation doit profondément évoluer. Il peut opter pour différents modes.

- Le basculement instantané.
- Le rodage sur un site pilote.
- L'installation progressive des fonctions.
- L e fonctionnement en double

Ces différents modes de mise en service peuvent, dans certaines situations, être mixés.

Raisonnements utilisés

Quelle que soit la solution choisie, le concepteur l'exprime en utilisant les règles et principes d'un modèle organisationnel de traitements. Le choix d'un mode de mise en service s'avère parfois assez délicat, et peut avoir des conséquences sur la viabilité de la solution organisationnelle normale retenue pour le futur système en terme de risques, de délais et de coûts. C'est pourquoi ces principes doivent être définis dès l'étude préalable.

Scénarios de développement

Objectifs et résultats attendus

Il s'agit de préciser comment seront conduites les étapes ultérieures, en particulier les études détaillées et technique, la réalisation et la mise en service. L'équipe de projet s'attachera à préciser les points suivants :

- Découpage en sous-projets ou lots (cf. Principes généraux de la démarche)
- Préconisation ou contraintes d'enchaînement du développement.
- Modalités de développement (interne, accompagné, sous-traité)

Raisonnements utilisés

Le concepteur pourra s'appuyer sur les MCT et MOT pour définir l'éventuel découpage en lots ou sous-projets qui seront eux-mêmes exprimés sous la forme de MOT et MOD.

Estimation des délais et des coûts

Objectifs et résultats attendus

A partir de l'ensemble de toutes les spécifications élaborées dans les tâches précédentes, il s'agit de fournir des évaluations sur les délais concernant les

étapes suivantes (étude détaillée et étude technique, réalisation, mise en service...) et sur les différents coûts associés (matériel, logiciel, personnel).

Le concepteur présente les postes de coût associés aux différentes composantes des étapes ultérieures :

Ces coûts peuvent être exprimés en unités monétaires ou en unités de ressource humaine. De la même façon, on établira un planning prévisionnel, éventuellement en dates relatives, pour les étapes ultérieures.

Raisonnements utilisés

L'évaluation des charges fait appel à des raisonnements et des modèles d'estimations que nous ne développerons pas ici et qui pourraient faire l'objet d'un ouvrage spécifique. Plusieurs méthodes de conduite de projet proposent de tels modèles dont beaucoup restent encore « protégés » comme savoir-faire des sociétés utilisatrices. L'évaluation du volume de l'étude détaillée peut s'appuyer sur le nombre de tâches à automatiser, pondéré par un coefficient grossier de complexité de la tâche.

Le planning sera représenté par un diagramme de Gantt ou un réseau Pert.

Appréciation des différentes solutions proposées

Objectifs et résultats attendus

Il s'agit ici d'exposer comment les différentes solutions répondent aux objectifs et aux contraintes retenus précédemment pour le système d'information, en particulier dans les phases de lancement et de conception des solutions.

Raisonnements utilisés

La nature même de la tâche d'appréciation ne relève pas directement des raisonnements formels de la méthode Merise. Le concepteur a cependant intérêt à s'appuyer sur l'ensemble des modèles élaborés dans les tâches précédentes.

Rédaction du dossier de choix

Objectifs et résultats attendus

Il s'agit d'élaborer les documents permettant la prise de décision par les responsables de l'organisme, en particulier ceux qui sont présents dans le comité de pilotage. Le rapport final d'étude préalable se compose habituellement de deux documents.

- Le dossier de choix, un document de synthèse d'environ 30 pages, destiné au groupe de pilotage afin d'étayer sa prise de décision :
- Le dossier d'étude préalable qui réunit l'ensemble des documents élaborés au fur et à mesure de l'étape

Raisonnements utilisés

Des plans-types ont été élaborés soit par des sociétés de service, soit par les services méthode de certaines entreprises ou administrations.

Décision sur l'étude préalable

Cette étape n'appartient plus formellement à l'étape mais vient la conclure.. Le groupe de projet a remis au groupe de pilotage les dossiers de fin d'étude préalable après avis du groupe de validation. Il appartient donc au groupe de pilotage de se prononcer sur la suite à donner au futur projet.

Selon le cas, cela pourra se traduire par :

- Le choix de développement d'une des solutions proposées.
- La demande d'un complément d'étude pour mieux départager certaines solutions.
- Le report ou l'abandon du projet.

L'étude détaillée

Objectifs de l'étude détaillée

L'étude préalable, en proposant des solutions, a permis au groupe de pilotage de choisir le profil général du futur système d'information. Cependant, les spécifications élaborées sont insuffisantes pour permettre une réalisation immédiate :

L'étude détaillée viendra don étendre l'étude préalable avec pour objectifs:

- La description de tous les processus composant le fonctionnement du futur système.
- La définition exhaustive des informations utilisées et mémorisées.
- La spécification complète des tâches à effectuer, en particulier pour celles à informatiser.
- La description des procédures exceptionnelles, les phases transitoires et le fonctionnement dégradé.

L'étude détaillée permet de spécifier l'intégralité du fonctionnement du futur système d'information organisationnel:

L'étude détaillée produit ainsi un véritable cahier des charges utilisateur et constitue la base de l'engagement que prend le concepteur vis-à-vis de l'utilisateur.

L'étude détaillée peut être découpée en plusieurs sous-projets sur la base des propositions de scénarios approuvés dans l'étude préalable. Chaque sous-projet

sera conduit selon la démarche de la phase.

Les phases de l'étude détaillée

L'étude détaillée est menée en deux phases majeures: une phase de spécifications générales et une phase de spécifications détaillées. La phase de spécifications générales permet d'étendre les modélisations de l'étude préalable à l'ensemble de l'activité étudiée. La phase de spécifications détaillées fournit une description détaillée du fonctionnement du futur système d'information tel qu'il sera perçu par les futurs utilisateurs.

Outre cette analyse détaillée du fonctionnement « normal » du futur système d'information, le concepteur devra également spécifier :

- Les procédures organisationnelles de la mise en service permettant de passer du fonctionnement du système actuel au système futur, ou *procédures transitoires.*
- Les procédures organisationnelles à appliquer lors d'une indisponibilité des ressources informatiques, ou *procédures de secours.*

L'étude détaillée se conclue enfin par la réactualisation des estimations et la rédaction du rapport final. La figure 15.8 présente l'enchaînement des phases de l'étude détaillée.

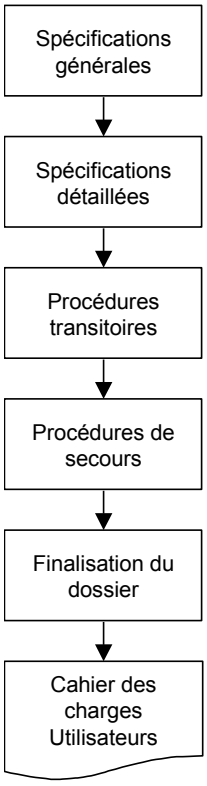


Figure 15.8 : Les phases de l'étude détaillée.

Phase de spécifications générales

Cette phase vise à étendre, généraliser les spécifications de l'étude préalable à l'ensemble du domaine retenu pour l'étude détaillée. Cette phase se décompose en quatre tâches (voir figure 15.9).

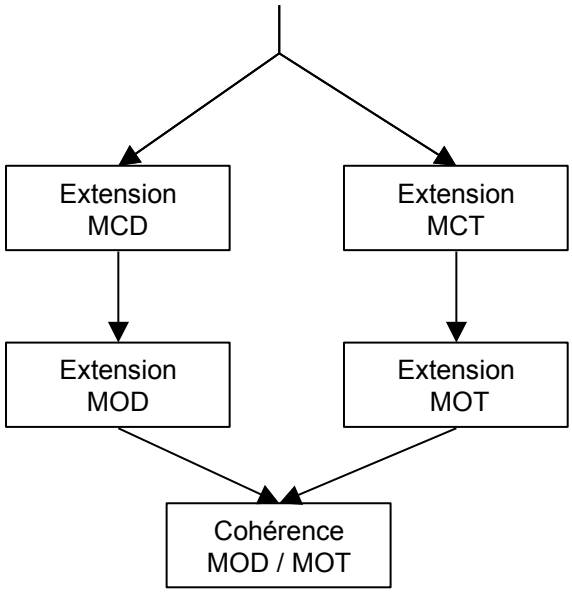


Figure 15.9 : Phase de spécifications générales de l'étude détaillée.

Extension du modèle conceptuel de données

Objectifs et résultats attendus

L'objectif de cette tâche est de compléter le modèle conceptuel de données par les concepts secondaires dont la modélisation n'a pas été prise en compte lors de l'étude préalable et d'enrichir la liste des propriétés des entités et des relations du modèle.

Raisonnements utilisés

Le concepteur pourra faire appel à des modélisations avancées : sous-types, contraintes de valeurs et inter-relations, règles.

Extension du modèle conceptuel de traitements

Objectifs et résultats attendus

Il s'agit de reprendre le modèle conceptuel de traitement issu de l'étude préalable et le compléter par les concepts écartés du sous-ensemble représentatif. Les présentations des résultats sont semblables à celles de l'étude préalable ; seul le champ de l'analyse est étendu.

Raisonnements utilisés

Il s'agit de la reprise des éléments du modèle conceptuel de traitements et de la mise en œuvre de tous les formalismes associés à ce modèle.

Extension du modèle organisationnel des données

Objectifs et résultats attendus

Il s'agit d'étendre le modèle organisationnel de données (MOD) de l'étude préalable, dans le cadre de la solution retenue pour le futur système.

Raisonnements utilisés

Le concepteur applique l'ensemble des raisonnements présentés au chapitre 10, en particulier au niveau du type et de la taille des propriétés. En cas de répartition organisationnelle, il est indispensable de présenter les MOD locaux.

Extension du modèle organisationnel de traitements

Objectifs et résultats attendus

Il s'agit de fournir une description complète de l'ensemble des procédures. Le modèle organisationnel de l'étude détaillée propose une description complète du fonctionnement du futur système, vu de l'utilisateur. Il devra donc être établi en étroite collaboration avec lui.

Le concepteur ne présente pas ici la description détaillée de chaque tâche. Celle-ci sera abordée dans la phase suivante. Le concepteur présente le modèle organisationnel par les documents suivants :

Raisonnements utilisés

L'intégralité des règles du formalisme organisationnel des traitements est appliquée. Le concepteur veillera à analyser tous les fonctionnements du futur système, y compris ceux correspondant à des situations particulières. Signalons que ce document servira de base pour la rédaction du manuel des procédures d'utilisation du futur système.

Cohérence MOD / MOT

Objectifs et résultats attendus

Comme en étude préalable, cette tâche de confrontation permet d'une part de s'assurer de la cohérence dans l'utilité et l'utilisation des données dans les traitements, d'autre part d'enrichir les modélisations respectives.

Raisonnements utilisés

A ce stade, le concepteur utilise la grille de cohérence globale telle que présentée au chapitre 11.

Phase de spécifications détaillées

Cette phase constitue le cœur de l'étude détaillée. C'est au cours de cette phase que se construit la description complète et détaillée du fonctionnement du futur système d'information dans sa partie externe, vu du système d'information organisationnelle. La figure 15.10 présente la décomposition en tâches.

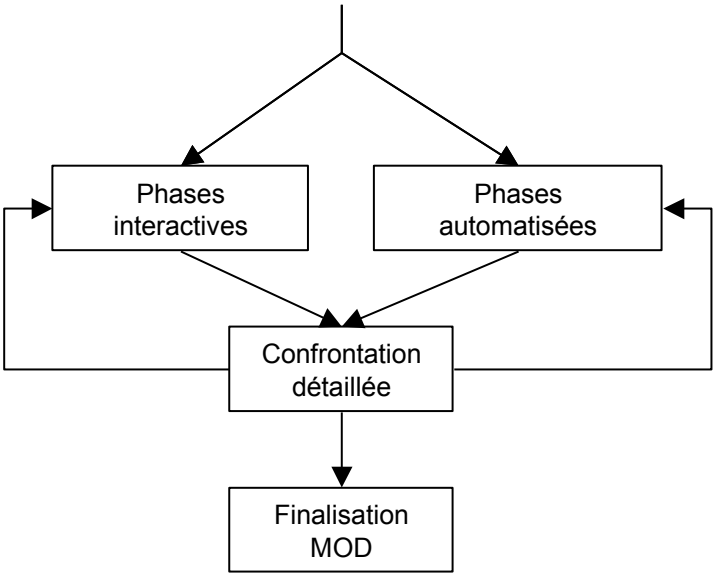


Figure 15.10 : La phase de spécifications détaillées de l'étude détaillée.

Spécification détaillée des phases interactives

Objectifs et résultats attendus

Cette tâche consiste à spécifier intégralement les traitements à effectuer pour informatiser les phases ou tâches interactives ou conversationnelles.

Cette description détaillée de l'informatisation d'une phase interactive correspond en fait à une première modélisation logique de traitement sous la forme de procédure logique. Si la tâche est relativement simple, spécifier la tâche sera équivalent à spécifier l'ULT associée à cette tâche. Si la phase est plus complexe, la phase sera décomposée en tâches élémentaires, correspondant à des ULT, donnant naissance à une procédure logique.

Quelque soit le cas, pour chaque ULT, on précisera :

- La présentation détaillée de l'interface avec les informations correspondantes (écran, état, badge, ticket...).
- Les règles de traitement à appliquer (contrôles à la saisie, calculs arithmétiques et logiques, règles de présentation pour la restitution des informations).

- Les règles et les actions effectuées sur les données mémorisées à partir des informations utilisées dans l'ULT.
- Les messages et diagnostics d'erreurs propres à l'ULT.

Raisonnements utilisés

Le concepteur met en oeuvre le premier niveau de modélisation logique des traitements. La spécification détaillée de chaque tâche, de chaque ULT est exprimée directement par le concepteur et s'enrichira lors du processus de confrontation détaillé. En fait, la tâche sera intégralement et correctement décrite à l'issue de la confrontation détaillée.

Pour cette tâche de spécification détaillée, l'utilisation d'outils de maquettage constitue aujourd'hui le moyen le plus efficace pour réaliser rapidement les spécifications externes de l'interface, assurer automatiquement la confrontation détaillée et faciliter la validation par les utilisateurs.

Spécification détaillée des phases automatiques

Objectifs et résultats attendus

Il s'agit de spécifier intégralement les traitements à effectuer pour des phases ou tâches à automatiser ne demandant que la ressource machine; ces tâches sont souvent appelées batch. La différence majeure avec la spécification des tâches conversationnelles provient de l'absence de spécificité liée aux postes informatique et à l'interactivité de l'interface.

Ces tâches automatiques concernent :

- La restitution d'états (statistiques, par exemple).
- La mise à jour en masse.
- L'archivage et l'épuration de la mémoire court terme.
- Le transfert entre systèmes.

Ces tâches concernant fréquemment la restitution de résultats, la principale spécification comporte :

- La présentation des résultats en fonction des souhaits des gestionnaires.
- Les calculs et algorithmes.
- Les actions sur les données.

Raisonnements utilisés

A la différence des tâches conversationnelles, l'absence d'interactivité rend moins nécessaire de détailler la procédure logique associée. Par contre ce travail d'analyse logique des traitements devra s'effectuer lors de l'étude technique.

Confrontation détaillée

Objectifs et résultats attendus

Cette tâche consiste à confirmer définitivement la compatibilité entre les traitements à effectuer (les tâches ou les ULT primaires) et les données mémorisées (le modèle conceptuel/organisationnel de données).

A l'issue de cette opération de confrontation détaillée, et sous réserve qu'aucune modification ne soit intervenue sur les données ou les traitements suite à un enrichissement, le modèle conceptuel / organisationnel de données est déclaré cohérent par rapport aux tâches du modèle organisationnel de traitements représentées essentiellement par leur modèle externe.

En fait, le concepteur pratique un processus itératif :

- Confrontation.
- Enrichissement du modèle conceptuel / organisationnel de données.
- Enrichissement du modèle organisationnel de traitements.

Raisonnements utilisés

A partir de la description de la tâche ou des ULT la composant, le concepteur constitue le modèle externe et peut appliquer l'algorithme de confrontation détaillée présenté au chapitre 10 « Confrontation algorithmique détaillée ». Comme déjà rappelé, le recours à des outils de maquettage peut réaliser l'essentiel de ce travail.

Finalisation du modèle organisationnel de données

Objectifs et résultats attendus

Il s'agit tout d'abord de prendre en compte les enrichissements issus de la confrontation et à compléter ce MOD pour en constituer la version définitive.

Raisonnements utilisés

L'équipe de projet confirme les choix concernant le MOD précédent :

- Prise en compte exhaustive des informations métier à mémoriser.
- Chiffrage des entités, des relations et des propriétés.
- Répartition sur les différents sites organisationnels.

L'équipe applique ces choix aux nouvelles informations du MCD validé. Elle complète enfin le MOD sous les trois aspects suivants :

- Propriétés exprimant des « états ».
- Prise en compte des durées de vie.
- Modèles organisationnels de données d'archives.

Phase de spécification des procédures transitoires

Ces procédures concernent la période de mise en service du nouveau système d'information, et précisent les conditions dans lesquelles s'effectuera le transfert. Bien que provisoires, la spécification de ces procédures est indispensable à la réussite du futur système. La période de mise en service est une étape critique et il importe que son organisation en soit clairement exprimée. L'essentiel des spécifications propres à cette transition est du niveau organisationnel de traitements. Comme l'illustre la figure 15.11, on abordera les problèmes suivants :

- La récupération et le transfert des données.
- Les principes du basculement entre le système actuel et le système futur.
- Le ou les modèles organisationnels de traitements (MOT) durant la période transitoire.

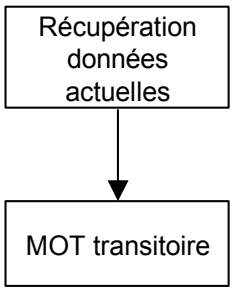


Figure 15.11 : La phase de spécification des procédures transitoires de l'étude détaillée

Récupération et transfert des données actuelles

Objectifs et résultats attendus

Il s'agit d'abord de définir la nature des informations à récupérer du système actuel, puis de spécifier les traitements prenant en charge le transfert, ainsi que celles qui permettent un chargement initial.

Dans de très nombreux cas, les utilisateurs désirent récupérer tout ou partie des informations existantes, mémorisées sous une forme informatique (transfert) ou manuelle (chargement initial).

Le concepteur spécifie, avec le même degré de détail que pour les tâches normales, l'ensemble des tâches permettant d'effectuer ce transfert ou ce chargement initial. Ces tâches sont généralement en batch

En ce qui concerne les tâches de transfert, le concepteur doit exprimer :

- La structure des informations actuelles à récupérer (schéma de fichier, portion de schéma de base de données).
- Les éventuelles règles de traitement (contrôle de vraisemblance, filtrage préalable, concentration ou éclatement de valeurs...).

- Les éléments du modèle conceptuel futur concerné par cette mise à jour.
- Le volume à transférer.

En ce qui concerne les tâches de chargement initial, le concepteur doit exprimer :

- La nature des tâches (saisie transactionnelle directe, encodage en masse et mise à jour en différé).
- La présentation des informations à saisir (dessin d'écran ou de bordereau).
- Les règles de traitement.
- Le volume à saisir.

Raisonnements utilisés

Hormis le caractère provisoire de ces tâches, les raisonnements nécessaires à leur spécification sont ceux utilisés pour la spécification détaillée des tâches normales.

L'expression de ces tâches de transfert et chargement initial est indispensable pour évaluer correctement les charges liées à la mise en service.

Modèle organisationnel de traitements transitoire

Objectifs et résultats attendus

L'étude préalable avait permis de définir les principes de passage du système d'information actuel au système d'information futur.

L'étude détaillée doit spécifier en détail les différentes procédures de l'organisation provisoire, sous forme d'un ou plusieurs MOT transitoires.

Raisonnements utilisés

Le concepteur utilise intégralement les règles et formalismes d'élaboration d'un modèle organisationnel de traitements. Il doit cependant être attentif:

- Aux disponibilités partielles des ressources techniques de certains postes.
- A la faisabilité de certaines procédures (disponibilité des informations, ordonnancement des tâches).
- A l'appréciation de la charge de travail requise pour un poste.

Phase de spécification des procédures de secours

Le modèle organisationnel du système en fonctionnement normal est fondé sur une disponibilité des ressources informatiques. En cas d'incident privant l'entreprise de ces ressources pour une durée plus ou moins longue, il est

indispensable de prévoir les procédures de fonctionnement à appliquer. Ces dispositions concernent les utilisateurs ; elles ne préjugent pas des dispositifs techniques retenus pour prévenir de tels incidents ou pour en minimiser l'impact. La figure 15.12 présente les tâches de la démarche.

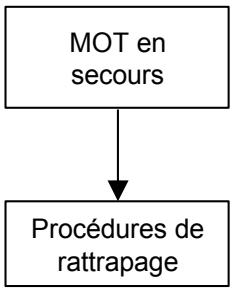


Figure 15 .12 : La phase de spécification des procédures de secours de l'étude détaillée

Modèle organisationnel des traitements en secours

Objectifs et résultats attendus

Il s'agit de proposer et de décrire l'organisation à appliquer lors d'une indisponibilité des ressources informatiques. L'indisponibilité des ressources informatiques n'a que peu d'impact sur la structuration des informations mémorisées, mais elle remet totalement en cause l'équilibre manuel/informatisé.

Le concepteur doit définir le fonctionnement à adopter, donc proposer un modèle organisationnel spécifique. Il est indispensable de prévoir ces différentes procédures, car une improvisation lors de l'incident a toujours des conséquences néfastes, surtout si le taux d'automatisation est important avec des tâches en réponse immédiate. Différents types de stratégies peuvent être adoptés :

- L'attente.
- Le retour au manuel ; il faut alors spécifier complètement les nouvelles tâches à assurer (contenu, documents, répartition)

Raisonnements utilisés

L'expression de ce modèle organisationnel de traitements se fait avec les modélisations habituelles.

Procédures de rattrapage de l'activité

Objectifs et résultats attendus

Cette tâche consiste à spécifier dans quelles conditions s'effectuera la reprise des informations accumulées lors d'un incident. Suivant la stratégie retenue face à l'incident, le rattrapage se fera d'une façon « naturelle » ou devra être organisé.

Même si la solution retenue ne nécessite pas de formalisation particulière, le concepteur devra clairement préciser la conduite à tenir pour cette tâche de récupération.

Raisonnements utilisés

S'agissant essentiellement de spécification de tâches, le concepteur met en œuvre les raisonnements du modèle organisationnel des traitements, et plus particulièrement permettant la description détaillée d'une tâche.

Finalisation de l'étude détaillée

Cette phase conclue l'étude détaillée par une validation générale des spécifications proposées, une réactualisation des estimations et de la planification des étapes suivantes et par la rédaction finale du dossier d'étude détaillée (figure 15.13).

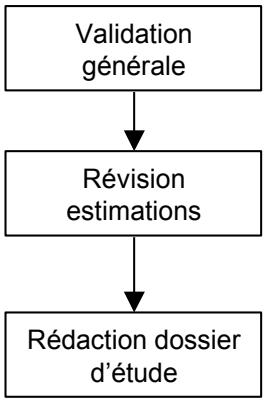


Figure 15.13 : Phase de finalisation de l'étude détaillée.

Validation générale

Objectifs et résultats attendus

Cette tâche consiste à présenter au groupe de validation, représentant les gestionnaires et les utilisateurs, l'ensemble des spécifications générales et détaillées élaborées dans cette étape. Bien que de telles présentations partielles aient pu avoir lieu au cours de l'étape, seule cette dernière validation permet une véritable récapitulation de l'ensemble finalisé..

Cette présentation permet d'obtenir la validation du métier, indispensable à la poursuite du projet.

Raisonnements utilisés

Comme dans pour les validations précédentes, le groupe de projet doit être très attentif aux modalités de présentation en évitant le simple « déballage » des modélisations. La réussite du projet tient autant à la qualité de la validation qu'à la qualité des spécifications.

Révision des estimations et de la planification

Objectifs et résultats attendus

Les spécifications détaillées des tâches à informatiser ainsi que la définition complète des données à mémoriser permettent de d'affiner les précédentes estimations

Ces nouvelles estimations se traduisent par :

- Un chiffrage détaillé des charges de développement par tâche
- Le choix, le dimensionnement et l'affectation des moyens de développement.
- La réactualisation du planning (découpage, répartition des tâches,…).
- Le réajustement de l'architecture informatique envisagée.

Raisonnements utilisés

Le concepteur fait appel aux mêmes techniques déjà utilisées dans la tâche d'estimation en étude préalable.

Rédaction du dossier d'étude détaillée

Objectifs et résultats attendus

Le dossier d'étude détaillée matérialise l'ensemble des travaux réalisés dans les différentes phases de l'étude. Ce dossier peut comporter plusieurs documents :

- Une note de synthèse, destinée au groupe de pilotage, qui reprend les points essentiels du futur système ainsi que les éléments de chiffrage.
- Le rapport d'étude détaillé, destiné aux intervenants qui prendront en charge la suite du développement, qui comprend l'intégrale des spécifications générales et détaillées élaborées dans l'étape.
- Le contenu des spécifications réalisées avec un outil (modélisations, maquettes)

Ce dossier fournit

- aux utilisateurs une description détaillée de leur futur système d'information ;
- aux informaticiens, la base pour l'élaboration des spécifications

informatiques correspondantes ;

- aux organisateurs, les éléments pour la mise en place des nouvelles procédures.

Raisonnements utilisés

La rédaction de ce dossier est évidemment progressive au cours de l'étude détaillée.

Comme pour le dossier d'étude préalable, il existe des plans-types qui facilitent l'élaboration des différents documents.

L'étude technique

Objectifs de l'étude technique

L'étude détaillée a permis d'obtenir l'ensemble des spécifications du futur système, du point de vue utilisateur. L'étude technique constitue le complément de spécifications informatiques nécessaires pour assurer la réalisation du futur système.

L'étude technique permet de définir complètement :

- La structure physique des données (fichiers ou bases de données).
- Les programmes, modules ou composants à réaliser ou intégrer.
- Les procédures techniques de sécurité.
- La planification de la réalisation.

Les raisonnements utilisés sont :

- La modélisation logique puis physique de données (MLD, MPD).
- La modélisation logique puis physique de traitements (MLT, MPT).

L'étude technique permet d'établir le cahier des charges de réalisation qui, en association avec le cahier des charges utilisateurs, constitue le document contractuel pour la production de logiciels.

Les phases de l'étude technique

L'étude technique est menée en deux phases : tout d'abord la définition des architectures de données et de programmes, et ensuite la préparation de la réalisation (voir figure 15.14).

Bien que l'étude technique soit une étape importante et indispensable dans la démarche de conception et de réalisation pour l'informatisation d'un système d'information, nous ne lui accorderons pas, dans le cadre de cet ouvrage centré sur Merise, de développements aussi importants que pour les étapes précédentes. En effet, nombre des raisonnements utilisés relèvent plus du génie

logiciel que de l'ingénierie de système d'information et sont développés dans d'autres ouvrages.

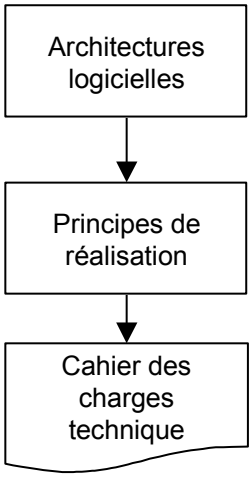


Figure 15.14 : Les phases de l'étude technique.

Architectures logicielles

Cette phase, conduisant à la définition des architectures tant au niveau des données que des programmes, se décompose en deux tâches majeures présentées à la figure 15.15.

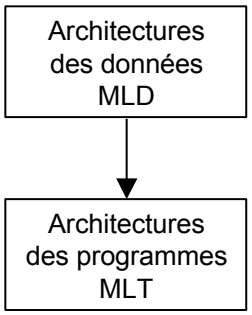


Figure 15.15 : La phase d'architectures logicielles de l'étude technique.

Architectures technique des données

Objectifs et résultats attendus

Il s'agit de définir une description informatique opérationnelle de l'organisation des données mémorisées. Cette organisation doit être d'une part conforme à la sémantique exprimée par le modèle organisationnel de données spécifié, d'autre part de permettre l'implantation sur le système informatique cible.

Selon les systèmes de gestion de bases de données (SGBD) utilisés pour gérer les données, le concepteur définie:

- le schéma de la base,
- l'allocation des espaces physiques,
- les dispositifs de protection d'accès et de confidentialité,
- les procédures de sécurité.

Raisonnements utilisés

Le MOD spécifié en étude détaillée, est d'abord traduit en MLD relationnel « brut » selon les règles présentées au chapitre 13

Puis, en respectant des objectifs généraux de performances d'accès et de stockage, le concepteur réalise une optimisation en tenant compte des possibilités techniques du matériel et du logiciel de gestion des données retenu, en appliquant les principes présentés au chapitre l4.

Ce modèle est enfin transcrit dans le langage du SGBD.

La plupart des outils évoqués au chapitre 18 automatisent en partie ces travaux.

Architecture technique des programmes

Objectifs et résultats attendus

Il s'agit de définir une description complète de la structuration technique du logiciel en fonction des outils de production de logiciels, ainsi que des logiciels de développement utilisés.

Suivant la technologie et les outils qui seront utilisés pour la production, le concepteur spécifie :

- Les composants logiciels à construire ou à utiliser.
- Les dialogues, modules ou transactions à programmer.
- L'enchaînement ou les échanges entre les différents modules ou composants.
- La répartition des traitements entre client et serveur.
- La description technique de chaque module ou composant (l'expression informatique des algorithmes, l'accès aux données, les sécurités techniques).

Raisonnements utilisés

Pour élaborer cette architecture et le découpage en modules ou composants le concepteur peut certes s'appuyer sur la modélisation logique des traitements. Il peut également faire appel à des approches plus de génie logiciel proposées par les méthodes objets. Dans cette tâche, le concepteur tient compte essentiellement des capacités des logiciels systèmes et du matériel.

Dans la mesure où l'enchaînement des dialogues exprimés dans l'étude détaillée

prend en compte des éléments d'ergonomie attachés à différents postes de travail, le concepteur s'efforcera de respecter cet enchaînement. Les spécifications des transactions devront également respecter les standards de qualité définis dans l'entreprise (structure standard, appellation des programmes et des données, commentaires...).

Pour la production d'états, la disponibilité d'outil de production rapide de rapports ou de langage d'interrogation doit être prise en compte par le concepteur.

Préparation de la réalisation

Cette phase regroupe un ensemble de préconisations nécessaires pour aborder l'étape de réalisation. La figure 15.16 décline ces différentes tâches qui ne seront que succinctement décrites.

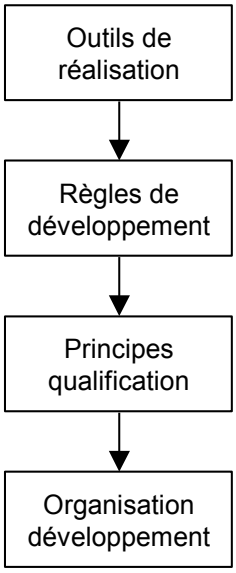


Figure 15.16 : La phase de préparation de la réalisation de l'étude technique.

Outils de réalisation

Il s'agit de recenser le ou les outils de réalisation retenus et d'en mentionner les possibilités et les contraintes. Rappelons que le concepteur avait déjà partiellement tenu compte de ces aspects dans les spécifications détaillées.

Règles de développement

Quelle que soit la méthode de génie logiciel appliquée, il est nécessaire d'énoncer l'ensemble des règles de construction et d'écriture du logiciel qui devront être respectées.

Ces règles peuvent entre autre porter sur :

- La structuration de la programmation.
- Le nommage et les commentaires.
- La documentation du code.

Principes de qualification

Dans cette tâche, on définit les modalités de tests que seront appliqués dans l'étape de production du logiciel :

- Test unitaire
- Test d'intégration
- Jeu d'essais
- Gestion des versions et des configurations logicielles

Organisation du développement

Il s'agit, sur les principes déjà esquissés en étude préalable (scénarios), de fixer l'organisation générale de l'étape de production du logiciel en précisant :

- La répartition des travaux entre les équipes.
- Le planning général de production du logiciel.
- Les modalités d'assurance qualité.

La production du logiciel

Objectifs de la production du logiciel

L'ensemble des spécifications précédentes, étude détaillée et étude technique, proposait un système d'information « papier ». Cette étape consiste à réaliser concrètement dans des langages, sur du matériel, l'ensemble de ces spécifications. A l'issue de cette étape, matérialisée par une recette technique, l'ensemble du système devra être conforme aux spécifications fonctionnelles et techniques.

Les phases de la production du logiciel

Nous ne développerons pas les différentes tâches de la production du logiciel. Tous les savoir-faire de génie logiciel et les méthodes de conduite de projet décrivent abondamment cette étape. De plus, les raisonnements de conception spécifiques à Merise ne concernent plus cette étape

Rappelons toutefois que, malgré la rapide évolution des outils d'aide à la réalisation et les efforts de réutilisation de composants logiciels, cette étape

reste encore l'une des plus lourdes dans démarche d'informatisation d'un système d'information organisationnel.

La figure 15.17 présente les principales phases usuelles de cette étape.

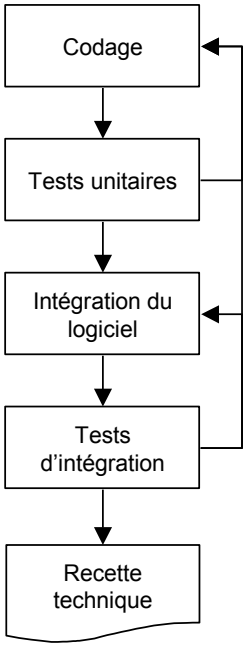


Figure 19.1 : Les phases de la production du logiciel.

La recette avant lancement portera sur le logiciel et sa documentation ; elle sera menée par les utilisateurs et le service d'exploitation au cours de l'étape suivante de mise en service.

La mise en service

Objectifs de la mise en service

L'ensemble des programmes développés et testés associés à une structure d'accueil de données mémorisées constitue une application informatique. Sa greffe sur l'entreprise, en lui ajoutant sa dimension organisationnelle, permet à cette application de devenir le support d'un système d'information.

Aussi, un soin tout particulier doit-il être apporté pour la véritable naissance du futur système d'information. C'est au cours de cette étape que la responsabilité du système d'information est transférée du groupe de conception et de réalisation (maîtrise d'œuvre), aux gestionnaires et aux utilisateurs (maîtrise d'ouvrage).

L'objectif principal de cette étape est de rendre opérationnel le système d'information. A l'issue de cette étape, les parties prenantes dans le projet peuvent se prononcer sur la recette définitive, et dissoudre la plupart des

structures spécifiques ayant permis la concrétisation du projet. Le fonctionnement de l'application informatique relève alors de l'exploitation.

Les phases de la mise en service

Si l'on constate dans la pratique de la démarche de Merise, une certaine convergence sur les découpages adoptés pour les étapes de conception, il faut admettre que cette étape de mis en service, lorsqu'elle est mentionnée, peut recouvrir une très grande variété de contenus et découpages.

La figure 15.18 propose un déroulement en quatre phases. Toutefois, compte tenu de la remarque précédente, nous ne développerons que très succinctement les tâches qui les composent.

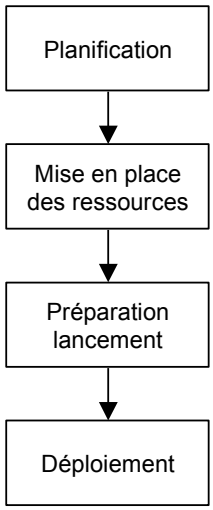


Figure 15.18 : Les phases de la mise en service.

Planification de la mise en service

Il s'agit de déterminer un scénario détaillé et réactualisé de l'ensemble des travaux nécessaires pour la mise en exploitation du nouveau système. Lors de l'étude préalable, le concepteur avait proposé une esquisse des moyens à prévoir pour l'installation. Depuis cette étape, les spécifications se sont affinées ; tous ces éléments permettent de produire un planning où toutes les tâches de mise en service sont explicitées. Ce scénario actualisé contient quatre calendriers particuliers :

- Mise en place des moyens techniques.
- Mise en place de l'organisation.
- Mise en place des ressources humaines.
- Mise en place de l'information externe.

La mise en service commence souvent dès la fin de l'étude détaillée ; elle se

déroule donc d'abord en parallèle avec l'étude technique et la réalisation ; puis à la suite de la réalisation dont elle met en exploitation le résultat.

Il est donc important de synchroniser très exactement les plannings correspondants.

Mise en place des ressources

La mise en place des ressources touche aussi bien les ressources techniques que les ressources humaines.

On retrouve dans cette phase les tâches suivantes :

- La mise en place des moyens techniques.
- La préparation des jeux d'essai.
- La rédaction de la documentation utilisateur.
- La mise en place de la nouvelle organisation.
- La mise en place des ressources humaines.

Préparation du lancement

Cette phase regroupe un ensemble de tâches de formation et d'information qui doivent intervenir avant le lancement effectif du système d'information.

Ces tâches concernent entre autre :

- La formation du personnel utilisateur et informaticien.
- L'information interne.
- L'information externe.
- La mise au point du plan de lancement détaillé

Déploiement

Cette phase correspond à l'installation de l'application qui va progressivement devenir opérationnelle.

On distingue globalement les tâches suivantes :

- La recette avant lancement qui s'applique sur les jeux d'essais précédemment constitués.
- Le lancement effectif conformément au scénario établi
- Une période probatoire durant laquelle les équipes de développement interviennent pour corriger les inévitables « pannes de jeunesse » et apporter les conseils nécessaires à une bonne utilisation du nouveau système.
- La recette définitive prononcée par la maîtrise d'ouvrage qui conclue à

l'acceptation totale du nouveau système d'information et met un terme à la mission de la maîtrise d'œuvre.

16

La démarche rapide de Merise

Principes généraux

Contexte d'émergence d'une démarche rapide

Les démarches rapides sont apparues dans la décennie 90 à la fois en réponse aux critiques de lourdeur à l'encontre des démarches classiques et sous l'influence de l'évolution des technologies et des nouvelles contraintes, en particulier économiques.

Aujourd'hui, ces deux approches doivent être considérées comme complémentaires, chacune adaptée à des catégories de projets différentes.

Critiques de la démarche « classique »

Les principales critiques faites à l'approche classique trouvent essentiellement leurs justifications dans les excès et les erreurs de conduite de projets, surtout constatés dans les années 80 pour des projets de taille importante. On peut notamment citer :

- le dérapage des délais et des coûts ainsi que les dérives de contenu,
- la nécessité d'un engagement technologique à long terme,
- un cycle de conception / réalisation / mise en oeuvre trop long, conduisant à un manque de visibilité et. nécessitant des choix souvent présentés comme irréversibles.

En réalité, il est plus vraisemblable, sans nier les échecs passés, que ces défauts soient plus imputables à la nature des projets, à leur dimension et à leur complexité (dans certains cas non maîtrisable) qu'à la démarche proprement dite ou aux méthodes utilisées. Dans le domaine informatique, cette décennie 80 fut marquée par une volonté de généralisation et de mise en cohérence des systèmes d'information des entreprises et des administrations, engageant fortement leur stratégie, impliquant des remises en cause souvent lourdes des principes de gestion et d'organisation, nécessitant de nombreux préalables à l'engagement des projets informatiques. Or le plus souvent, ce sont ces projets informatiques qui ont supporté le coût et la responsabilité de l'ensemble de

l'opération, renforçant encore plus l'impression de lourdeur et la difficulté de maîtriser objectifs et résultats.

Des évolutions technologiques

La décennie 90 a principalement vu le micro-ordinateur s'imposer comme le poste informatique normal dans les environnements organisationnels. Ses capacités et sa puissance, alliées à une grande richesse et diversité de logiciels disponibles en ont fait un composant majeur tant dans les architectures client-serveur que intra/internet. Du point de vue de l'utilisateur, cette généralisation du micro-ordinateur s'est traduite par :

- une standardisation de l'ergonomie visuelle et mentale liée aux environnements graphiques et en particulier à l'hégémonie de Windows ®,
- une intégration des fonctionnalités (applications métier, bureautique, messagerie, communications, intra/internet,…),
- une popularisation de l' environnement informatique.

Bien que toutes les nombreuses évolutions technologiques aient pu influencer l'environnement de conception de systèmes d'information, nous retiendrons plus particulièrement :

- L'affirmation et la généralisation des SGBD relationnels.
- La richesse, la diversité et la puissance des outils de développement.
- L'essor de la programmation orientée objet et l'utilisation de composants réutilisables.
- La disponibilité d'outils de conception et de prototypage performants.
- L'ouverture des systèmes d'informations vers le Net.

De nouvelles exigences

Toujours dans cette décennie 90, sûrement sous l'effet de conditions économiques difficiles, les entreprises privilégient des projets dont la taille, l'ampleur et l'ambition sont notablement plus réduites qu'auparavant. La plupart des projets sont alors engagés selon une démarche et un management de projet très différents, en préconisant notamment :

- une participation actives des utilisateurs,
- un cycle itératif de conception/réalisation/amélioration,
- l'exigence d'une maîtrise des coûts et des délais,
- des équipes plus compactes, expérimentées et polyvalentes,
- un contenu fonctionnel restreint et connu du projet,

- un contexte de choix de gestion et d'organisation stable et explicité.

La démarche RAD

Cette démarche dite « Rapid Application Development » est apparue au début des années 90, en s'opposant aux démarches en cascade, jugées trop lourdes et trop contraignantes pour le développement d'applications petites et moyennes. Elle suppose qu'au début du processus de développement, l'utilisateur n'a qu'une idée très sommaire de ses besoins. Elle se propose donc de rendre l'expression et la spécification des besoins progressives et pratiquement simultanées avec la conception et la réalisation.

Cette démarche s'appuie sur :

- Des préconisations d'organisation du travail, où notamment les utilisateurs sont associés de manière continue à l'ensemble du cycle de développement.
- La définition préalable d'une enveloppe de budget temps et coût (« Faire ce que l'on peut avec ce que l'on donne » au lieu de « Donner ce qu'il faut pour faire ce que l'on veut »).
- L'utilisation systématique de techniques de maquettage -prototypage.
- Un découpage relativement fin de l'application dès le début du projet ; chaque fonction ou lot découpé suit les différentes phases de la démarche, jusqu'à la finalisation complète du projet.
- Des itérations successives, essentiellement localisées dans la phase de réalisation.

La figure 16.1 présente le découpage de la démarche RAD.

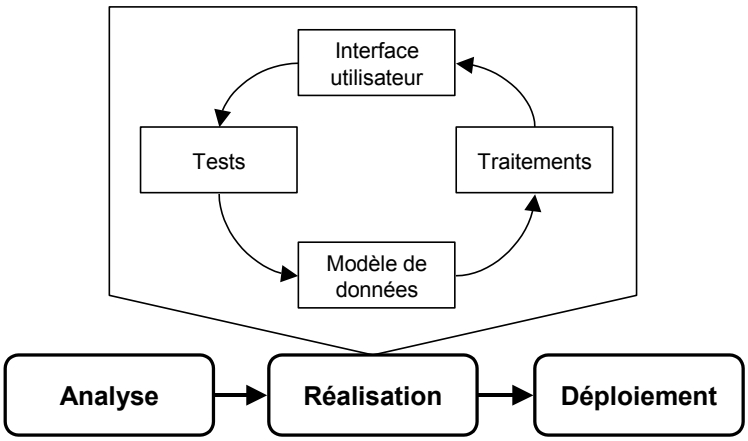


Figure 16.1 : Phases de la démarche RAD

Positionnement de la démarche rapide

La démarche rapide de Merise proposée s'appuie sur

- La démarche classique initialement préconisée dans la méthode Merise, en s'inspirant des principes de certaines de ses étapes, en raison de leurs caractères éprouvés pour la conduite de projets de taille moyenne ou importante.
- Les techniques de maquettage – prototypage, qui s'avèrent déterminantes dans les phases de spécifications détaillées pour permettre l'expression et la validation des besoins utilisateurs.
- La démarche RAD dont certains principes peuvent être utilisés, notamment pour l'approche itérative de la conception, pour le passage du maquettage - prototypage à la réalisation, pour le découpage de l'application en modules ainsi que pour l'organisation et la répartition des rôles dans le projet.

La démarche rapide de Merise se démarque toutefois d'une démarche exploratoire. Cette dernière se caractérise par l'absence de projet spécifié, par des itérations rapides entre le développement d'une proposition, l'utilisation d'un module, la recherche de son adéquation et sa correction. Cette démarche exploratoire qui peut paraître séduisante pour certains informaticiens en raison de sa simplicité de mise en oeuvre et des raccourcis méthodologiques, est peu compatible avec un système stabilisé et ne donne aucune garantie de pérennité de l'investissement effectué.

La démarche rapide de Merise proposée concerne essentiellement l'aménagement de la démarche et non les raisonnements exprimés par les différentes modélisations. Ces modèles et les formalismes qui permettent de les représenter constituent indéniablement le « génotype » de la méthode Merise. Ils sont spécifiques et adaptés à la nature du problème à traiter (la conception d'un système d'information) et indépendants du type de projet (taille et complexité, système informatique cible). Le respect de ces principes de modélisations joue un rôle fondamental dans la communication entre l'ensemble des partenaires impliqués dans la conception et le développement des systèmes d'information, et elles contribuent à l'amélioration de la qualité.

Aussi, dans la démarche rapide, ces modèles sont-ils conservés sans aucune modification des concepts-types, des symbolisations graphiques et des règles de modélisation. Seul leur usage, l'ampleur de la modélisation et le degré de détail sont adaptés au contexte de la conception rapide.

Découpage en étapes

La démarche rapide se positionne uniquement sur la période de développement d'un projet. A l'instar de la démarche classique, elle se décompose en étapes puis en phases au cours desquelles le concepteur met en œuvre les différents raisonnements de conception et de réalisation.

La figure 16.2 présente les quatre étapes de la démarche rapide.

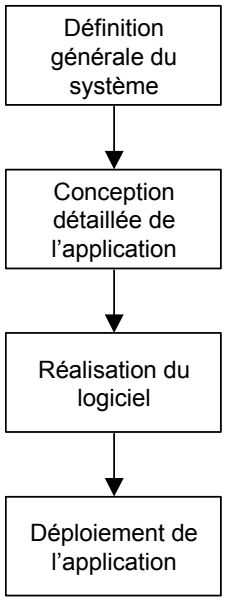


Figure 16.2 : Les étapes de la démarche rapide de Merise

La définition générale du système

Objectifs de la définition générale du système

Cette étape doit réunir les objectifs fondamentaux de l'étude préalable et des spécifications générales de la démarche "classique", et ce dans un délai relativement restreint.

La définition générale du système doit tout d'abord permettre au concepteur-réalisateur de comprendre très tôt l'étendue et la complexité du projet, afin d'en évaluer la charge et le délai ou si le projet envisagé tient dans les budgets fixés à priori. Elle doit ensuite s'efforcer de formaliser un avant-projet ou une proposition destinée à la maîtrise d'ouvrage permettant de s'assurer de la compréhension réciproque du projet. Elle permet enfin d'esquisser des spécifications générales de l'application.

Les phases de la définition générale du système

L'étape de définition générale du système se décompose en deux phases présentées sur la figure 16..3.

Elle est concrétisée par la production de deux documents.

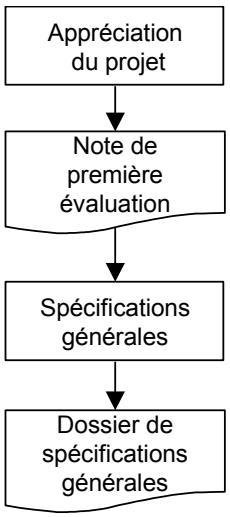


Figure 16.3 : Les phases de la définition générale du système

Phase d'appréciation du projet

Cette phase, très courte doit permettre au concepteur-réalisateur d'effectuer une appréciation rapide du contour et des enjeux, indiquant une faisabilité globale du projet. L'évaluation sommaire résultante est communiquée au maître d'ouvrage

Positionnement du projet

Il s'agit globalement de qualifier la nature et le contenu du projet, à savoir :

- Quelle sont l'étendue et la complexité estimées du projet?
- Le projet a t-il des objectifs bien définis ?
- Les incertitudes organisationnelles et informatiques sont-elles limitées ?
- Ce projet relève t-il vraiment d'un développement rapide ?
- Connaît-on assez bien le domaine d'activité ?
- A-t-on fait quelque chose d'équivalent ?
- Les utilisateurs compétents sont-ils suffisamment disponibles ?

Délimitation des fonctions à informatiser

L'objectif est de recenser rapidement les principales fonctions à informatiser et délimiter ainsi le périmètre fonctionnel du projet. Cette tâche est également l'occasion de mieux connaître les motifs de l'informatisation, les souhaits et attentes des utilisateurs, ainsi que les contraintes d'environnement organisationnelles et informatiques.

On utilisera essentiellement une modélisation à base de diagramme des flux (chapitre 5).

Nous suggérons de construire deux diagrammes des flux :

- Un premier diagramme de type contextuel représentant les échanges du système avec les acteurs de son environnement.
- Un second explicitant le fonctionnement général du système étudié, de type macro-MCT.

Evaluation des enjeux

Ainsi apprécié en terme de complexité et fonctionnalités, le concepteur peut mieux appréhender les enjeux de ce projet et justifier l'adoption d'une démarche rapide. Le concepteur évalue également la compatibilité globale de l'enveloppe budgétaire envisagée (en temps et en ressources) avec le contour général du projet.

Les conclusions de cette phase sont synthétisées dans une note de quelques pages résumant les lignes directrices du projet. Son contenu doit, d'une part permettre au maître d'ouvrage de s'assurer que sa demande a été globalement cernée, d'autre part lui donner en retour des éléments d'évaluation sur la taille du projet. A ce niveau, il ne peut y avoir aucun élément de solution proposé

Phase de spécifications générales

Au cours de cette phase, le concepteur va compléter sa connaissance des activités du champ de l'étude pour élaborer les spécifications générales de la future application, notamment définir :

- les données à mémoriser,
- les fonctions à informatiser,
- les éventuelles interrelations avec d'autres systèmes.

La figure 16.4 présente les tâches qui sont menées en cohérence mutuelle.

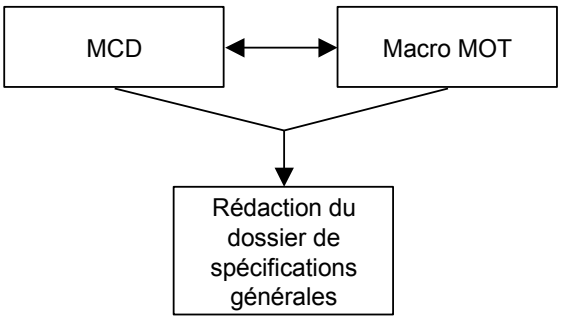


Figure 16.4 : La phase de spécifications générale de la définition générale du système.

Sur la base de cette définition des fonctionnalités générales, le concepteur

pourra procéder à un découpage en "modules" pour préparer l'étape suivante.

Modélisation conceptuelle des données

Cette modélisation reste incontestablement la plus efficace pour capter et formaliser efficacement la signification générale des informations utilisées.. Au niveau de cette phase, un MCD doit être quasiment complet et vérifié en termes d'entités et de relations; on peut toutefois se satisfaire d'une définition partielle des propriétés. Cet aspect structurel est assez important car, d'une part il fournit le support du dialogue sur la compréhension des informations utilisées (même s'il n'est pas explicitement communiqué aux utilisateurs), d'autre part il sert de point de départ pour la construction future des tables (même si un travail d'optimisation est nécessaire ultérieurement).

La mise en œuvre d'une démarche rapide ne saurait être associée à l'utilisation d'un formalisme modélisation des données « allégé ».La pratique a suffisamment démontré la puissance du formalisme entité – relation dont l'utilisation peut être modulée en fonction des besoins d'expression.

Macro-modélisation organisationnelle des traitements

Dans des spécifications générales, il est indispensable d'établir la liste des fonctions à informatiser et d'en présenter des schémas d'enchaînement fonctionnel d'utilisation. Bien que l'on puisse se contenter d'une simple énumération, une représentation sous la forme d'un diagramme général de phases ou macro MOT fournit un complément explicatif efficace.

Dans la démarche rapide, par rapport au MOT classique, ce type de modèle sera appliqué en :

- négligeant éventuellement la répartition en postes,
- s'intéressant d'abord aux phases interactives directement liées à une procédure,
- gardant un niveau de détail compatible avec les spécifications générales.

L'utilisation d'une telle technique n'est pas obligatoire. Elle reste à l'appréciation du concepteur selon la complexité du système et les nécessités de formalisation de la solution pour en faciliter la compréhension mutuelle.

Rédaction du dossier de spécifications générales

Cette tâche conclue l'étape de définition générale du système qui doit fournir un descriptif général de la future application, présenté sous la forme d'un *dossier de spécifications générales.*

Il est évident que la taille de ce dossier est proportionnelle à celle du projet. Il est hors de propos de demander la production de documents aussi épais que ceux d'une étude destinée à un grand ou moyen projet; l'excès inverse (c'est à

dire pas de documentation de spécifications générales) est tout aussi critiquable. Le contenu type d'un tel document pourrait être :

- un rappel de la nature de la demande et du contexte du projet,
- la présentation des processus à informatiser (diagrammes de flux)
- les principales fonctionnalités proposées (macro-MOT),
- les informations gérées informatiquement (MCD),
- les implications matérielles (configuration),
- une réévaluation de la durée.

La conception détaillée de l'application

Objectifs de la conception détaillée de l'application

Cette étape doit fournir les spécifications complètes du contenu de l'application à réaliser, des points de vue de l'utilisateur et de l'informaticien. Dans la démarche proposée, cette étape est principalement basée sur l'utilisation de maquettage dynamique (pour tout ou partie du projet).

Sur la base des spécifications générales élaborées lors de l'étape précédente, le concepteur va imaginer, pour les fonctionnalités déterminées, des enchaînements de dialogues possibles. Il va également devoir définir les données associées à la maquette.

A l'issue de cette étape, le concepteur doit obtenir l'accord des futurs utilisateurs sur le contenu et la forme de l'application à livrer.

Si le passage de la conception détaillée à la réalisation s'effectue par module, l'accord sur les spécifications détaillées s'effectuera au niveau de chaque module.

Les phases de la conception détaillée de l'application

L'étape de conception détaillée de l'application se décompose en deux phases itératives comme l'illustre la figure 16.5:

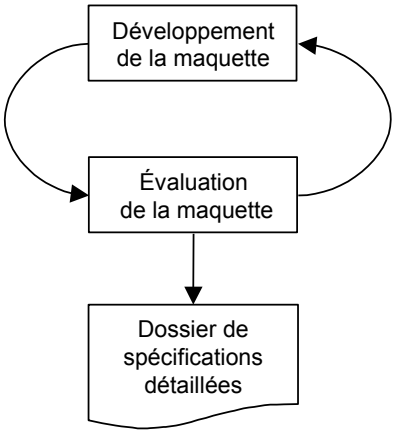


Figure 16.5 : Les phases de la conception détaillées.

Elle se concrétise par la maquette associée à un dossier de spécifications détaillées.

Phase de développement de la maquette

L'une des originalités de la démarche proposée est le recours quasiment systématique aux techniques de maquettage dynamique rendues facilement utilisables soit avec les environnements de développement disponibles en micro-informatique, soit avec des outils de conceptions disposant de cette fonctionnalité (cf. chapitre 18). Cette technique permet d'exprimer en détail les spécifications externes de la future application et d'améliorer ainsi la compréhension des besoins des utilisateurs, donc de minimiser les risques ultérieurs de développement.

Cette phase de développement de la maquette, fortement itérative, se décompose en tâches présentées à la figure 16.6.

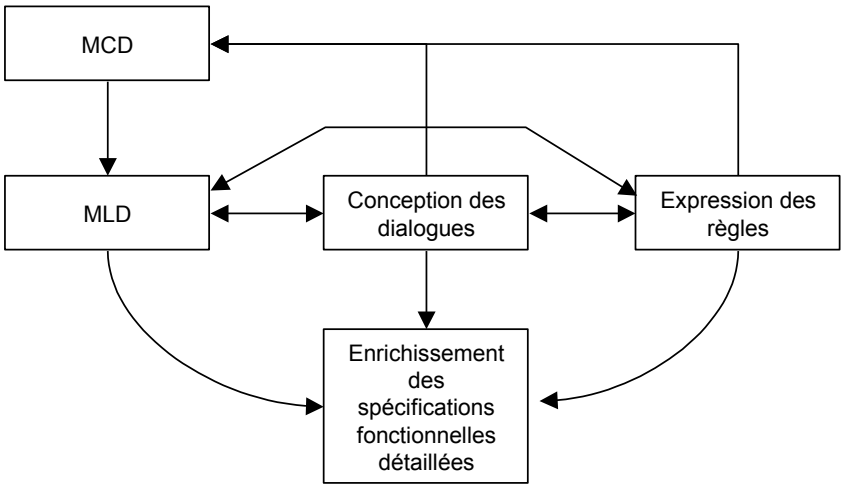


Figure 16.6 : La phase de développement de la maquette de la conception détaillée.

Rôle du maquettage

La place importante prise par le maquettage - prototypage nous conduit à préciser le vocabulaire et l'usage de cette technique. Nous distinguons la maquette du prototype. Un prototype est destiné à tester, dans des conditions normales d'utilisation, certaines fonctionnalités de l'application. Une maquette est essentiellement mise en oeuvre pour valider l'interface homme/machine avec des degrés de réalisme variables (maquette statique, maquette dynamique).

Intérêt du maquettage

L'interactivité étant le caractère prédominant des applications actuelles, il n'est pas toujours aisé d'en traduire tous les aspects dans une spécification formelle. En conséquence, il est indispensable de recourir aux techniques de maquettage qui jouent aujourd'hui un rôle essentiel dans le processus de conception et réalisation des applications :

Avantages du maquettage	*Mais...*
✓ Validation dynamique des besoins utilisateurs	✓ La maquette n'est pas le système final
✓ Détection anticipée des incompréhensions entre utilisateurs et développeurs Démonstration rapide de la faisabilité et de l'utilité de certaines fonctions	✓ La maquette n'est pas destinée à valider la conception du logiciel mais à valider les besoins des utilisateurs
✓ Évaluation des éléments critiques de l'application	✓ Le maquettage nécessite une participation active de l'utilisateur

Diversité de maquettages

On distingue généralement deux grandes catégories de maquettes :

- les maquettes statiques proposant des dessins d'écrans ou d'éditions sans possibilité d'interactivité,
- les maquettes dynamiques proposant une interaction élémentaire (la logique de dialogue) et un enchaînement des écrans ou éditions.

En complément, et chaque fois que cela pourra faciliter la compréhension du métier de l'utilisateur, la maquette pourra accueillir les différentes règles de traitement mises en œuvre.

L'approche par maquettage dynamique, qui a notre préférence, permet d'obtenir rapidement une validation de la solution proposée à l'utilisateur, par une présentation visuellement et dynamiquement la plus proche possible de l'application définitive.

Maquette jetable ou construction progressive du système ?

L'utilisation des techniques de maquettage comme support d'aide aux spécifications externes du futur système pose le problème de la réutilisabilité de la maquette dans la réalisation proprement dite. Deux approches s'opposent :

- la construction d'une maquette jetable, élaborée de préférence avec des outils simples et rapides,
- la construction progressive de la future application par l'utilisation restreinte des outils de développement (dessins d'écran, programmation ponctuelle, enchaînements).

Chaque solution présente des avantages et des inconvénients que l'on peut synthétiser dans le tableau suivant :

	Avantages…	*Inconvénients…*
Maquette jetable	✓ Rapidité et simplicité de construction et de modification ✓ Possibilité d'ajouter des éléments facilitant la validation ✓ Support d'une documentation fonctionnelle et ergonomique. ✓ Frontière nette entre la conception et la réalisation.	✓ Recommencer le travail pour la réalisation, en particulier pour les dessins d'écran. ✓ Indisponibilité d'objets d'interface entre maquette et réalisation. ✓ Limitations du réalisme
Construction progressive	✓ Pas de rupture entre conception et réalisation. ✓ Familiarisation anticipée et relativement stable de l'utilisateur à la future application. ✓ Réalisme poussé de la maquette	✓ Maîtrise indispensable du langage de développement. ✓ Temps et contraintes de développement. ✓ Mélanger concevoir et réaliser. ✓ Se dispenser de documentation fonctionnelle

Dans notre contexte, la construction d'une maquette doit se concentrer sur l'interface homme/machine sans se préoccuper des problèmes d'architecture logicielle. De plus la validation par les utilisateurs des règles de traitement (en particulier tous les algorithmes de calcul et de mises à jour) ne peut pas être réalisée dynamiquement par le biais du maquettage, mais nécessite une présentation descriptive qui peut difficilement être exprimée dans un langage de programmation.

Pour ces motifs, nous suggérons, dans la démarche rapide de Merise, l'approche par une maquette « jetable ». Toutefois, le concepteur devra, dans le

choix de ses outils de maquettage, veiller aux possibilités de récupération vers l'environnement de réalisation des composants essentiels de la maquette, en particulier les dessins d'écran.

Modélisation des données

Dans une conception à base de maquettage, il est indispensable d'associer les données utilisées dans la maquette (dans les dialogues et les règles) à des données modélisées. Cette mise en correspondance des données contribue fortement à la mise en cohérence telle que préconisée dans le chapitre 11.

Le niveau de modèle peut être conceptuel ou logique. Comme la plupart des outils de maquettage dynamique, quel que soit le type de maquettage retenu, s'appuient sur une modélisation logique de données, nous proposons la démarche suivante :

- Enrichir progressivement le MCD élaboré dans la phase de conception générale.
- Transformer ce MCD en MLD relationnel en appliquant les règles présentées au chapitre 12
- Utiliser les données logiques dans la construction de la maquette.

Il est cependant souhaitable que le MCD reste la référence de la modélisation des données dans des spécifications fonctionnelles détaillées. Ainsi au cours de la construction itérative de la maquette, les deux niveaux de modèles de données s'élaboreront progressivement.

Conception des dialogues

L'aspect visuel des dialogues est l'atout principal du processus de conception par maquettage, surtout par son aspect communication – validation. La construction des dialogues constitue donc la tâche centrale de la phase de développement de la maquette.

On distingue trois aspects dans la construction d'un dialogue :

Le design de la présentation qui consiste à agencer les objets graphiques d'interface (champs de saisie, combo et list box , radio boutons, cases à cocher, boutons, cadres, onglets, etc.) sur la fenêtre de dialogue conformément aux finalités fonctionnelles de la tâche correspondante. Ce travail fait appel aux principes d'ergonomie visuelle et mentale qui constituent éventuellement des normes ou des standards.

L'expression de la logique du dialogue qui consiste d'une part à associer le contenu d'objets d'interface aux données modélisées (directement ou via une règle) ainsi que les valeurs prédéterminées ou contrôlées, d'autre part à spécifier les comportements d'interaction avec ou entre ces objets d'interface, comportements qui pourront éventuellement être simulés

dans un maquettage dynamique.

La définition des enchaînements entre dialogues, souvent concrétisés par des boutons ou des menus, qui exprime la procédure de travail proposée à l'utilisateur. Le choix des enchaînements à offrir reste toutefois assez délicat à déterminer car il doit rester compatible avec les procédures organisationnelles liées à la fonction à informatiser. En complément, le concepteur peut également utiliser comme support de sa réflexion, la modélisation logique des traitements proposée par Merise (cf. chap. 11)

Expression des règles

La spécification détaillée des règles dans une approche par maquettage pose plusieurs problèmes :

- L'expression des règles de traitement (calcul et mise à jour des données) est indispensable dans un processus de conception.
- Quelle que soit la technique utilisée, on ne peut, en conception, valider des règles par démonstration ou par test ; il est nécessaire de les formaliser.
- Pour que cette formalisation soit compréhensible par l'utilisateur, il faut que le langage utilisé reste relativement naturel (ce qui exclut un langage de programmation)

En conséquence, nous préconisons dans la démarche rapide de Merise, l'expression des règles dans un formalisme de type langage naturel structuré, intégrant les données modélisées. Eventuellement, ces règles peuvent être associées à des objets graphiques d'interface qui situent leur application, mais leur validation ne peut se faire que par la relecture de l'expression formalisée.

Comme dans les tâches précédentes, la construction progressive des règles vient enrichir les données modélisées et assurer leur cohérence avec les traitements.

Enrichissement des spécifications fonctionnelles détaillées

La maquette est principalement orientée vers l'utilisateur afin de favoriser la validation. Les éléments ayant servi à sa construction qui constituent les spécifications fonctionnelles détaillées doivent être formalisés et complétés par d'autres éléments qui ne peuvent trouver place dans la maquette.

Parmi documents de spécifications fonctionnelles, on trouvera :

- Les modèles de données (MCD, MLD).
- Le descriptif des fonctions réalisées dans la maquette (présentation du dialogue, données concernées, règles de traitement utilisées),

- L'association des fonctions de la maquette aux phases du macro-MOT élaboré au niveau des spécifications générales
- La liste des fonctions qui n'ont pas fait l'objet de maquettage (dialogues simples, traitements batch, certaines éditions)

Ces documents sont le "reflet" de la maquette. Ils ne sont pas à priori destinés à une présentation à l'utilisateur, mais constituent plutôt des documents internes au développement. Cependant, la validation de la maquette induira la "validation" de ces documents qui serviront ainsi de références pour la réalisation définitive de l'application.

Phase d'évaluation de la maquette

L'objectif essentiel de cette phase est la présentation de la maquette à l'utilisateur en vue de la validation des propositions du concepteur. Afin de maîtriser la conception itérative par maquettage, il est nécessaire de bien distinguer deux tâches : la présentation de la maquette et la formalisation des amendements.

Présentation de la maquette

Pour que cette présentation se déroule efficacement, il faut veiller aux conditions de validation.

Le contexte et le rôle de validation doivent être expliqués à l'utilisateur. La validation par maquettage dynamique n'est pas un test du produit. Il sert à présenter d'une façon la plus réaliste possible la future application. Ce réalisme concerne exclusivement les interfaces homme/machine. Or d'autres éléments entrent en jeu dans les spécifications détaillées de l'application : positionnement de l'application dans le processus organisationnel de l'activité, interrelations avec d'autres systèmes, conditions d'utilisation, potentiels et ouverture de l'application (surtout en termes de données), règles de gestion.

A l'exception des toutes petites applications, il est préférable d'effectuer la validation par "module". L'utilisateur peut facilement se concentrer sur un sujet; l'utilisateur hésite moins à demander des modifications; les modifications ont une propagation limitée au niveau de la phase de validation.

La réussite d'une approche par maquettage dépend très fortement de la qualité de cette présentation aussi le concepteur doit apporter un soin particulier à l'animation de la séance de validation. Elle ne doit pas se réduire à une démonstration de la maquette; bien d'autres éléments constituent les spécifications détaillées de l'application: diagrammes de procédures, règles de gestion, éventuellement modèles de données, qui doivent également être présentés.

Formalisation des amendements

La présentation – validation a normalement généré des remarques et suggestions de la part des utilisateurs qui ont été recueillies par le concepteur.

Les résultats cette présentation de la maquette sont rassemblés dans un compte rendu d'évaluation qui reprend les maquettes et autres documents présentés avec mention des commentaires et demandes de modifications exprimées.

Sur la base de cette évaluation, le concepteur procède aux amendements nécessaires sur la maquette et les spécifications détaillées associées, puis réitère le cycle développement – évaluation.

La réalisation du logiciel

Cette étape, qui reste toujours essentielle dans le développement d'une application, met en oeuvre des techniques et des savoir-faire qui sont peu concernés par les apports méthodologiques de la méthode Merise; aussi cette étape ne sera pas détaillée en phases.

La réalisation de l'application définitive se fait en reprenant des composants déjà élaborés pour la maquette, en y rajoutant les éléments non simulables dans la maquette, en particulier des règles de traitement.

Dans la démarche proposée, on suggère d'aborder cette étape de réalisation dans la continuité de la maquette dynamique et en adoptant une démarche dite "en spirale". Cette dernière préconise un découpage en "modules fonctionnels" de taille limitée permettant un cycle conception-réalisation très rapide; l'application complète se construisant par une succession de cycles sur les modules

Dans cette étape, le réalisateur tient compte également des aspects sécurité, confidentialité et gestion de l'exploitation nécessaires à l'application.

Par rapport aux aspects abordés dans la démarche proposée, on rappellera deux tâches spécifiques à cette étape : l'optimisation de la base de données et la prise en compte des spécificités de l'environnement de développement retenu.

L'optimisation de la base de données

Dans l'étape de maquettage, nous avions préconisé de travailler sur un modèle logique brut ou très sommairement "optimisé". A l'issue de la conception détaillée de l'application, le concepteur dispose d'une description complète et validée des données (sous une forme conceptuelle et logique), accompagnée de l'ensemble des traitements à mettre en oeuvre. Il peut donc procéder, en toute connaissance, aux optimisations nécessaires.

Les techniques d'optimisation sont celles utilisées classiquement dans la méthode Merise pour les bases relationnelles (cf. chapitre 14).

Prise en compte des spécificités de l'environnement de développement

Les spécifications issues du maquettage ainsi que les éléments éventuellement

récupérés peuvent nécessiter une adaptation aux contraintes du logiciel de développement. Dans des cas extrêmes, cela peut conduire à une réinterprétation complète de la maquette.

Cette étape de réalisation du logiciel se concrétise bien évidemment par la recette technique de l'application complète en état de fonctionnement. Outre ce résultat fondamental, certains documents doivent accompagner le produit, à savoir :

- une *documentation technique* du produit, nécessaire aux opérations de maintenance;
- une *documentation utilisateur*, pour la partie intégrée dans le logiciel; l'autre partie pouvant être rédigée dans l'étape suivante.

Le déploiement

A ce niveau, suivant la nature et l'ampleur de l'application développée, la mise en service peut prendre les différentes formes évoquées dans la démarche classique.

Cette étape consiste à installer l'application sur un ou plusieurs sites pilotes afin de vérifier son acceptation par les utilisateurs. Elle permet également d'apporter d'une part les différentes corrections qui fiabiliseront l'application, d'autre part les modifications mineures demandées par les utilisateurs.

Dans tous les cas, on retrouvera au moins les tâches suivantes :

- Mise en place des moyens techniques, de l'organisation et des ressources nécessaires.
- Finalisation de la documentation utilisateur.
- Période probatoire.
- Recette définitive et généralisation.

17

L'organisation d'un projet Merise

Les deux composantes fondamentales de la méthode Merise présentées dans les chapitres précédents, les raisonnements (deuxième et troisième parties) et la démarche (quatrième partie) ne sont pas suffisantes. En effet, elles présentent, lors de la mise en application pratique, une lacune qui peut s'avérer fatale pour l'accomplissement du projet : la maîtrise du déroulement du projet et la responsabilité des choix et des décisions.

A l'issue de chaque étape de la démarche, voire de chaque phase, l'application des raisonnements de la méthode Merise a permis de proposer des solutions, à tous les niveaux, pour l'informatisation du futur système d'information. La maîtrise du déroulement du projet s'intéresse aux questions suivantes :

- Quelles sont les décisions attendues ?
- De quelle nature sont-elles ?
- Quelles sont les responsabilités à engager ?
- Quelles sont les ressources à mettre en œuvre ?

Jusqu'à présent, dans le déroulement de la méthode Merise, nous avons évoqué différents intervenants, notamment le concepteur, le réalisateur, le décideur. Il est nécessaire de se demander :

- Quels sont les différents types d'intervenants pour une mise en œuvre opérationnelle de la méthode Merise ?
- Quelles sont les structures à mettre en place pour assurer une maîtrise effective d'un projet?

Initialement, les promoteurs de la méthode Merise, préoccupés en priorité par une contribution en termes de démarche et de raisonnement, avaient laissé aux utilisateurs le soin d'apporter des réponses à ces questions. Les années d'expérience accumulées ont permis de conclure que ces problèmes devaient être abordés dans une méthode, d'autant qu'un certain nombre de traits généraux se dessinent aujourd'hui.

Enfin, la mise en œuvre opérationnelle d'une méthode passe maintenant obligatoirement par des outils logiciels d'aide à la conception et à la documentation.(chapitre 18)

Nous abordons dans ce chapitre, consacré aux moyens de mise en œuvre de la méthode Merise, d'une part les structures et l'organisation des différents intervenants dans un projet utilisant Merise et d'autre part le rôle de ces différentes structures au cours de la démarche.

Structures et intervenants dans un projet merise

Nous distinguons trois types de structures, tout d'abord les *structures permanentes*, ensuite les *structures spécifiques au projet* et enfin les *structures de conseil.*

Les structures permanentes

Nous regroupons, sous cette appellation, les différentes structures dont l'existence est indépendante d'un projet de conception et de réalisation de système d'information. Nous analysons, cependant, ces structures par rapport aux fonctions qu'elles exercent vis-à-vis d'un projet de système d'information.

Professionnels de la gestion d'un domaine d'activité

Nous regroupons, sous cette appellation, toutes les personnes exerçant une fonction, un métier, quel que soit leur niveau de responsabilité, dans un domaine d'activité. Le domaine peut recouvrir, soit une activité de gestion classique (vente, ressources humaines, comptabilité), soit une activité à caractère plus technique (recherche, production...).

Cette structure est fréquemment désignée par les utilisateurs. Elle est la destinataire de l'application informatique qui supportera son système d'information informatisé. En règle générale, cette structure permanente joue le rôle de maîtrise d'ouvrage dans un projet.

Rappelons que la compétence des personnes travaillant dans cette structure concerne avant tout leur domaine d'activité. Bien que ces utilisateurs/gestionnaires doivent s'impliquer dans la conception de leur système d'information, on ne peut exiger de ces personnes une maîtrise confirmée des démarches et des raisonnements utilisés lors de la conception et de la réalisation d'un système d'information informatisé.

Professionnels de la conception et de la réalisation de système d'information

Nous regroupons sous cette appellation les différents métiers dont la fonction concourt à la conception et la réalisation d'un système d'information

informatisé. Cette structure est fréquemment, et parfois abusivement, désignée par « les informaticiens ». En règle générale, cette structure permanente joue le rôle de maîtrise d'œuvre.

Les personnes exerçant dans cette structure présentent des compétences très diverses : organisateurs, concepteurs, réalisateurs, spécialistes matériels... Elles ont généralement acquis une expérience au cours de la conception et la réalisation de plusieurs projets dans des domaines d'activités divers.

Les structures de projet

Ces structures, sous forme de groupe, sont constituées à l'occasion d'un projet (voir figure 17.1). Ces groupes sont établis autour des rôles suivants : production du projet, validation du projet et maîtrise du projet. Leur composition et leur rôle varient au cours du cycle de vie du projet.

Groupe de pilotage

Ce groupe a la responsabilité de la maîtrise de projet en termes de décisions, de coûts et de délais. Ses participants se prononcent sur les choix proposés par le groupe projet. De par son rôle, ce groupe de pilotage est composé de responsables hiérarchiques

- de la maîtrise d'ouvrage ;
- de la maîtrise d'œuvre ;
- de la direction générale de l'entreprise.

Groupe de projet

Ce groupe a la responsabilité de la production du projet suivant les différentes phases de l'étude. La diversité des raisonnements à appliquer au cours de la démarche requiert des compétences variables. Ce groupe sera donc composé, dans des proportions variant avec le déroulement du projet,

- de professionnels de la gestion ;
- de professionnels de la conception ;
- de professionnels de la réalisation.

Quelles que soient les phases, il apparaît souhaitable qu'une mixité utilisateurs-informaticiens soit respectée. Ainsi, dans les étapes de conception du système d'information organisationnel (étude préalable, étude détaillée), il est indispensable que la majorité des participants au groupe de projet soient des professionnels de la gestion du domaine. Par contre, lors des étapes plus techniques (étude technique, réalisation) il est incontestable que le groupe de projet est constitué en quasi-totalité d'informaticiens.

D'une façon générale, on peut distinguer au sein du groupe de projet les

fonctions et compétences suivantes :

- un responsable système d'information, représentant permanent des intérêts de la maîtrise d'ouvrage, leader des utilisateurs, maîtrisant totalement l'activité du domaine étudié ;
- des utilisateurs-concepteurs, participants actifs à la conception de leur système d'information, maîtrisant certains points particuliers du domaine ou présentant des aptitudes à la modélisation ;
- un chef de projet, professionnel de l'informatisation, leader des informaticiens, maîtrisant totalement les méthodes et techniques de la conception et de la réalisation des systèmes d'information ;
- des informaticiens-réalisateurs, participants actifs à la conception et à la réalisation du système d'information.

Étant donné le rôle actif de ce groupe de projet, les participants sont généralement mobilisés à plein temps, et selon leurs interventions sur les différentes phases du projet.

Groupe de validation

Ce groupe est la représentation consultative des futurs utilisateurs du système. Bien qu'il ne participe pas activement à la production de l'étude et de la réalisation du projet, son avis est indispensable pour valider les propositions émises par le groupe de projet. Ce groupe est constitué essentiellement d'utilisateurs. Lors de sa constitution, on veillera à respecter une représentativité reconnue par l'ensemble des usagers.

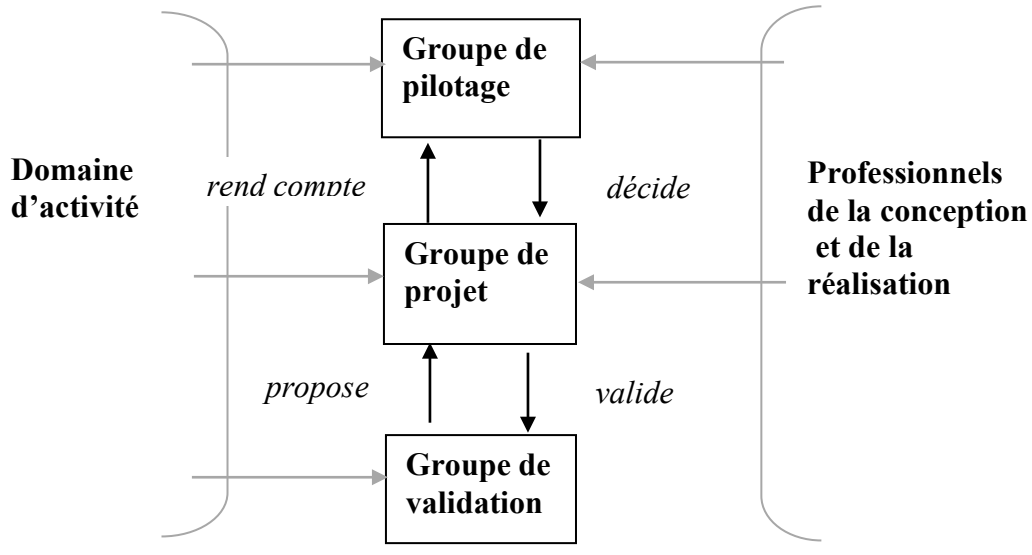


Figure 17.1: Structures dans un projet Merise.

Les structures de conseil

La conception et la réalisation d'un système d'information font appel à des connaissances très variées qui parfois ne sont pas toutes réunies au sein du groupe de projet. Afin de pallier à d'éventuelles lacunes, les structures de projet peuvent solliciter ponctuellement l'intervention de spécialistes dans différentes disciplines mises en œuvre dans la conception et la réalisation du système d'information. Nous pouvons distinguer différentes spécialités, dans le contexte des raisonnements de la méthode Merise :

- *Spécialiste du domaine d'activité* : La conception d'un nouveau système d'information est souvent l'occasion de rénover les techniques utilisées dans l'activité : nouvelle méthode de gestion de production, changement d'organisation... L'intervention du spécialiste porte alors sur le contenu même de l'activité et la manière de l'exercer.
- *Spécialiste de la méthode Merise* : Bien que facile à mettre en œuvre, la maîtrise complète de cette méthode requiert une pratique certaine, surtout pour l'élaboration des différents modèles et le respect de la démarche. L'intervention du spécialiste est alors d'ordre méthodologique afin d'éviter l'enlisement du projet sur des problèmes de formalisation et de démarche.
- *Spécialiste de l'organisation* : La méthode Merise a mis en évidence le rôle important joué par l'organisation dans la conception et le fonctionnement d'un système d'information. L'intervention du spécialiste porte alors sur l'appréciation des postes, des procédures, sur l'ergonomie générale de l'organisation humaine et matérielle.
- *Spécialiste des techniques informatiques* : Un projet de conception et de réalisation d'un système d'information comporte indéniablement une part importante de technique informatique, connaissance des matériels, problèmes de télécommunications, outils de productions, langages... L'intervention du spécialiste porte alors sur l'aide au choix du matériel, l'estimation des coûts et délais suivant les conditions de réalisation, la résolution de problèmes ponctuels lors de la réalisation.

Ces différentes compétences peuvent soit être disponibles au sein de l'entreprise, soit être ponctuelement recherchées à l'extérieur.

Rôle des structures dans la démarche

Les différentes structures précédemment décrites, et en particulier celles propres au projet, jouent des rôles divers au cours du déroulement du projet. L'organisation générale proposée pour Merise doit être adaptée dans le contexte particulier de chaque entreprise. Suivant la taille de l'entreprise et

l'ampleur du projet, on ne mobilisera pas les mêmes ressources. De plus, les personnalités des différents intervenants et les contingences matérielles ne permettent pas toujours de mettre en place une organisation idéale.

Nous voulons présenter, dans le contexte de ce document, des principes généraux sur les rôles respectifs des différents intervenants. Nous incitons vivement chaque entreprise utilisatrice de la méthode Merise à réfléchir à une personnalisation de cette organisation.

Classification des activités

Les récentes réflexions dans le domaine du génie logiciel et plus particulièrement dans le domaine de l'assurance qualité proposent une typologie des différentes activités émergeant dans un projet d'étude et de réalisation d'un système d'information. On distingue schématiquement les activités suivantes :

- *Produire*, c'est-à-dire exercer une activité intellectuelle ou physique contribuant à l'élaboration du projet, par exemple participer à la construction d'un modèle, écrire un programme, faire un test...
- *Documenter*, on distingue cette activité de production qui est destinée à faciliter la lecture des spécifications du projet ; par exemple rédiger un rapport d'étude, décrire un modèle, documenter un programme, écrire un manuel utilisateur...
- *Décider*, c'est-à-dire agir sur le déroulement du projet en termes de choix sur les coûts, délais ou qualité.
- *Valider*, c'est-à-dire se prononcer sur la conformité de la production par rapport aux souhaits et spécifications initiales.
- *Conseiller*, c'est-à-dire apporter un avis, une contribution ponctuelle lors de la production.
- *Piloter*, c'est-à-dire donner les orientations et les choix au groupe de projet, diriger l'équipe.
- *Approuver* les décisions prises par un autre intervenant, généralement en termes de respect de coût, de délai et de qualité.

Activités des structures de projet au cours de la démarche

Nous proposons, sous la forme d'un tableau succinct (voir figure 17.2) une répartition des différents rôles des intervenants dans la mise en œuvre de la méthode Merise. Nous invitons le lecteur à personnaliser cette grille en fonction des particularités de son entreprise.

	groupe pilotage	groupe projet				groupe validation
		Resp. SI	Utilisateur	Chef projet	Informaticien	
Etude préalable	décide	pilote produit	produit	produit documente	produit documente	valide
Etude détaillée	approuve	valide décide	produit	pilote produit documente	produit documente	valide
Etude technique	approuve			pilote produit décide	produit documente	
Production logiciel	approuve			pilote décide valide	produit documente	
Mise en service	décide	pilote produit valide	produit	produit valide	produit	valide

Figure 17.2 : Les rôles des structures

La validation, facteur de qualité

La démarche qualité est devenue un élément important du cadre méthodologique de conception de systèmes d'information et de réalisation d'applications informatiques. Il est donc naturel de retrouver cette préoccupation dans la mise en oeuvre de la méthode Merise.

L'un des principes de base de la qualité est la conformité du produit fourni aux exigences du client. Cette conformité passe entre autre par la validation des spécifications du produit.

La figure 17.3 rappelle les étapes successives permettant d'obtenir une qualité globale.

En informatisation de système d'information, les développeurs se préoccupent prioritairement de la qualité de la production, car elle reste interne à la maîtrise d'œuvre. Les deux autres aspects de la qualité impliquent une relation entre la maîtrise d'œuvre (utilisateurs) et la maîtrise d'ouvrages (informaticiens).

Dans la méthode Merise, ces deux "maillons" de la relation Utilisateurs - Informaticiens sont une préoccupation régulière tout au long du processus de conception. Elle se concrétise par l'implication de représentants de la maîtrise d'ouvrage dans le groupe de projet ainsi que la présence d'un groupe de validation.

Bien que fortement souhaitable, la participation de représentants d'utilisateurs aux travaux du groupe de projet, du moins dans les étapes d'étude préalable et d'étude détaillée, reste tributaire de leur disponibilité et de leur aptitude à la conception. Il faut cependant tenir compte de leur représentativité limitée.

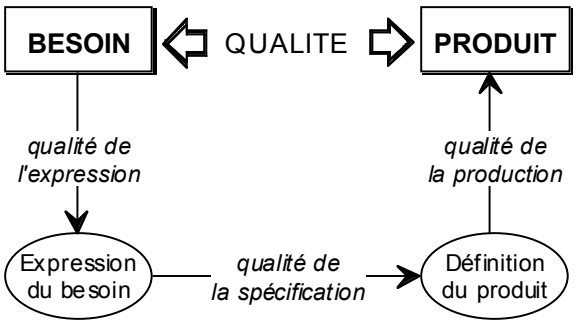


Figure 17.3 : Les maillons de la qualité globale.

Quant au groupe de validation, son rôle est indispensable dans le processus de validation. Par expérience, nous avons constaté que les projets où un tel groupe (quel que soit par ailleurs son intitulé, mais distinct du groupe de pilotage) était absent, on rencontrait de plus grandes difficultés dans l'acceptation de l'application par les utilisateurs.

Modalités de validation

La qualité de l'expression des besoins dépend essentiellement de la représentativité des utilisateurs tant dans le groupe de projet que dans le groupe de validation. Leur validation passe par des échanges réguliers entre les participants à ces groupes et le reste des utilisateurs concernés par le projet.

La qualité des spécifications est le maillon essentiel de la relation utilisateurs - informaticiens. La validation passe d'abord par une compréhension des spécifications et peut prendre plusieurs formes dont les conditions de mise en oeuvre sont différentes (panels, interviews, groupe de travail, ...).

Présentation des modélisations

La méthode Merise accorde une part importante à la modélisation pour l'expression des spécifications. L'expérience nous a confirmé l'efficacité de ce mode de spécification. Toutefois, le caractère formel des modélisations (en particulier dans les données) nécessite une certaine maîtrise des formalismes que l'on ne peut pas toujours attendre de tous les utilisateurs.

Aussi, l'utilisation directe des modélisations comme seul support de validation des spécifications ne peut pratiquement se réaliser qu'au sein du groupe de projet. Même dans ce cas, le chef de projet (informaticien) doit s'assurer de la maîtrise de la formalisation et ne pas hésiter à expliciter la signification des modélisations auprès des participants - utilisateurs. Par contre, la présentation directe et non commentée[a] des modélisations auprès du groupe de validation est, à l'expérience, à déconseiller, surtout pour les modélisations conceptuelles de données.

[a] Le comble étant d'adresser par courrier les modélisations aux membres du groupe de validation en demandant en retour leur accord pour validation !

Dans tous les cas, la présentation de modèles auprès du groupe de validation ne peut se faire que dans le cadre de la présentation de solutions, accompagnée d'explications, avec un important effort de pédagogie (sans pour cela se transformer en "cours" de modélisation ...).

Présentation des solutions

Il s'agit en fait de l'animation de réunions de validation. Sur la base des éléments de solutions élaborés par le groupe de projet, comprenant bien sûr les modélisations, le groupe de projet construit une présentation de ces solutions. Les animateurs feront appel aux divers modes et techniques de présentation.

Les modèles élaborés par l'équipe de projet, sur lesquels les spécifications s'appuient, doivent être cependant présentés aux gestionnaires et utilisateurs du domaine pour qu'ils puissent confirmer si les options prises, les modélisations retenues, expriment correctement les choix et les significations fondamentales du système d'information du domaine.

Ces personnes sont expertes dans leur domaine d'application, mais rarement dans les formalismes de Merise. Il est ainsi nécessaire de reformuler chaque modélisation en français. Dans cette reformulation :

- les concepts de base du domaine d'application feront l'objet d'une définition simple et rigoureuse,
- ces concepts seront illustrés par des exemples réalistes,
- le modèle sera expliqué et commenté, de préférence par parties (vues); de même, il faudra réduire le plus possible l'utilisation des termes techniques des formalismes (cardinalité, synchronisation...).

Bien préparée, une telle présentation contribuera ainsi à une meilleure compréhension du futur projet et facilitera la mise en service du futur système. Ce mode de validation est à conseiller en étude préalable et en phase de conception générale de l'étude détaillée.

Maquettage

En étude détaillée de la démarche « classique » ou dans le cadre de démarche « rapide » l'élaboration de l'interface utilisateur sous forme de maquettes ou de prototypes facilite grandement la validation par les utilisateurs (voir sur ce point le chapitre 16)

Compléments pour l'estimation des projets

Outre les ordres de grandeur pour l'estimation et la répartition des charges pour les différentes étapes du cycle de vie, Il peut être également intéressant de préciser, à l'intérieur de chaque étape, la répartition de la charge de travail par catégorie d'intervenants.

En partant de l'observation et de l'expérience de Cecima, largement confirmée par de nombreux professionnels, nous allons présenter des répartitions des charges de travail moyennes entre informaticiens et utilisateurs.(groupe de validation compris) On s'intéressera à deux types de répartitions : Une répartition par étape et une répartition par rôle.

Première répartition moyenne par étape et par rôle

Notre expérience nous conduit aux ordres de grandeur (exprimés en %) de la figure 17.4.

	Etude Préalable	Etude détaillée	Etude technique	Réalisation	Mise en service	Total
Informaticien	30	70	100	90	20	70
Utilisateur	70	30	0	10	80	30
Total	100	100	100	100	100	100

Figure 17.4 : Première répartition moyenne par étape et par rôle des charges de travail.

Il est à noter que la répartition de la charge de travail privilégie en étude préalable la part utilisateur (70%). Cette part se trouve inversée au niveau de l'étude détaillée (30%) par rapport aux informaticiens. La moyenne pondérée entre la charge de travail par rôle et la part de chaque étape dans le cycle de vie nous donne le total de la contribution des informaticiens (environ 70%) et des utilisateurs (30%) pour l'ensemble des étapes.

Seconde répartition moyenne par rôle et par étape

En fonction de la première répartition, la seconde répartition (voir figure 17.5) se déduit par calcul et indique, pour chaque rôle, la composition de sa charge totale de travail par étape.

	Etude Préalable	Etude détaillée	Etude technique	Réalisation	Mise en service	Total
Informaticien	5	25	20	45	5	100
Utilisateur	30	30	0	10	30	100

Figure 17.5 : Seconde répartition moyenne par étape des charges de travail.

On notera qu'à l'issue de la réalisation, l'informaticien a réalisé 95% de sa charge totale (pour l'informaticien, le projet est quasiment terminé !), alors que l'utilisateur conserve une charge non encore réalisée de 30% du total de sa

charge (pour l'utilisateur, le projet est loin d'être terminé !).

Guide de projet, manuel qualité

Un projet avec Merise implique avant tout des intervenants, qui mènent ensemble le projet (qualité, coût, délai) en utilisant à bon escient différents modèles du système d'information. Présenter au lecteur ces différentes notions, expliquer leurs tenants et aboutissants, tel est le but principal de cet ouvrage. Cet ouvrage peut aussi être utilisé par les équipes de projet, les reports étant accélérés par utilisation de l'index de fin de volume.

Il existe aussi un autre type de support, directement conçu pour faciliter la conduite de projet avec Merise : le *guide de projet*. Dans un tel guide, il s'agit moins de justifier les notions que de les appliquer rapidement et sûrement, d'où la structure habituelle d'un guide de projet :

- un ensemble de fiches d'action (une par tâche), rédigées dans un style le plus clair et le plus opérationnel possible ;
- un ensemble de fiches techniques complémentaires, rappelant comment utiliser telle ou telle technique de base en conception des systèmes d'information.

Il existe également un répertoire de références croisées entre fiches d'action et fiches techniques.

Remarques

Les fiches techniques constituent un lieu d'accueil privilégié pour les normes techniques et les standards de l'entreprise. Lorsque l'entreprise décide de s'engager explicitement sur la voie de la qualité, le guide de projet devient ainsi, tout naturellement, un manuel qualité.

Selon l'AFNOR, le manuel qualité décrit les dispositions prises par l'entreprise pour obtenir la qualité de ses produits ; en fonction de l'organisation de la société et de la nature générale des projets à traiter, il définit la technologie en vigueur pour assurer la qualité d'un développement, ici informatique.

18

Outils pour la mise en œuvre de Merise

Si des projets modestes peuvent se satisfaire aisément d'un ensemble de formulaires pour leur documentation, d'un tableau et de feutres pour la conception, le recours à des outils logiciels procure aux concepteurs et aux réalisateurs une assistance certaine.

La diffusion et l'intégration d'une méthode passent nécessairement par l'existence d'outils logiciels associés. Ces outils sont généralement conçus et diffusés par les promoteurs de la méthode. Dans le cas de la méthode Merise, son origine collective, source de richesse, a donné naissance à de nombreuses versions d'outils. Aujourd'hui seuls quelques-uns uns se sont imposés, ils assurent pour la plupart une bonne couverture fonctionnelle et méthodologique. (nous présentons plus loin l'un d'entre eux : Win'Design)

Les ateliers de génie logiciel

Les outils logiciels, support de la mise en œuvre de la méthode Merise ou d'autres méthodes, participent au courant plus large des ateliers de génie logiciel (CASE en anglais : Computer Aided Software Engeniering) couvrant tout ou partie des cycles de conception, développement et maintenance des applications informatiques.

Cette approche industrielle de la production de logiciels, initiée par l'informatique à vocation technique de type temps réel, s'est développée autour de deux axes :

- La conception du système d'information: *Upper Case* ou atelier de conception.
- La spécification du logiciel et la production du code: *Lower Case* ou atelier de réalisation.

Si pour le deuxième axe, la préoccupation essentielle de productivité a été et demeure dépendante de l'environnement technique de développement (SGBD, système d'exploitation, langage, interface utilisateur...), les outils relevant du

premier axe, au travers de méthodes de conception largement diffusées, ont une vocation plus large, et se prêtent plus facilement à une certaine normalisation et à une formalisation (niveau de modélisation, notions manipulées, représentation graphique, règles de comportement et de passage...

Dans les années 80, ces deux catégories d'outils ont eu des difficultés à communiquer, tant au plan des concepts utilisés, qu'au niveau technique. Ces difficultés étant souvent dues à une hétérogénéité des plates-formes matérielles supportant ces outils et à un problème de transformation des objets de conception en objets de réalisation; les méthodes étant sur ce point, notamment pour les traitements, relativement défaillantes.

Plus récemment, la cohabitation et la convergence de plusieurs courants marquent une étape importante de transition dans l'utilisation des ateliers et outils de génie logiciels :

- la convergence des environnements de développement (client-serveur, Internet, base de données relationnelles, langages, ..),
- l'utilisation de démarche de type "développement rapide",
- le développement en langage objet et la perspective de réutilisation "de composants logiciel" et d'objet « métier »,

rendent encore plus nécessaire la communication entre environnement de conception et de réalisation.

Ces dernières années la diffusion de méthode « orientée objet » comme UML, a pu faire penser que ce problème était quasiment résolu en uniformisant, par des concepts et des formalismes communs, la modélisation du système d'information organisationnel (S.I.O) et celle du système d'information informatisé (S.I.I).

Dans la réalité cette transition entre les « objets métiers »(S.I.O) et les « objets logiciel »(S.I.I) reste dans la même problématique que pour les méthodes « classiques »

La nécessité de communication est une des priorités majeures et l'évolution de ces outils se traduira, sous diverses formes, par une meilleure cohabitation pour assurer une continuité entre les spécifications externes d'un système d'information et la production des logiciels s'y appliquant. Actuellement la plupart des outils de conception, notamment ceux associés à Merise, permettent au moins la génération automatique des scripts de création et de modification SQL de bases de données directement exécutables par la plupart des SGBD, ainsi que la description des principaux triggers assurant l'intégrité de ces bases.

L'offre sur le marché, toutes catégories confondues, est actuellement importante et très diverse. Dans le domaine de la gestion, près d'une centaine d'outils sont recensés, et presque autant dans le domaine du temps réel. Il

existe par ailleurs de nombreux catalogues et enquêtes sur le sujet (CXP, YPHISE, AFCET, ADELI...) fournissant des synthèses et points comparatifs très utiles dans un domaine aussi ambitieux, mouvant et très prolixe en solutions, concepts et sigles divers.

La notion de dictionnaire

Avant de présenter, tout du moins pour ceux relevant de Merise, les caractéristiques générales des outils, il est utile de préciser la notion de dictionnaire. En effet, le dictionnaire, pivot des ateliers de génie logiciel, est déterminant pour qualifier le champ d'application et l'aptitude à communiquer et à évoluer de ces outils.

Le dictionnaire est destiné à mémoriser, gérer et contrôler les différents objets de conception et/ou de réalisation des systèmes d'information étudiés. Ainsi, il contiendra pour la méthode Merise, des notions telles que:

- entité, relation, propriété... pour le modèle conceptuel de données et le modèle organisationnel de données;
- opération, événement, acteur... pour le modèle conceptuel de traitements,
- tâche, procédure, règle, poste... pour le modèle organisationnel de traitements,
- record, item, table, attribut... pour le modèle logique de données,
- ULT (Unité Logique de Traitement), blocs... pour le modèle logique de traitements,
- programme, module, écran... pour les modèles physiques de traitements.

L'alimentation de ce dictionnaire peut s'effectuer progressivement au cours de la conception, enrichissant ainsi la base de description du système d'information étudié. Un tel dictionnaire permet, à tout instant de l'étude, la restitution d'une documentation à jour, normalisée et cohérente. L'intégration de tous les modèles permet également le recours à des références croisées, indispensables lors de modifications successives au cours de la conception, pour vérifier les règles du formalisme et garantir une cohérence des modèles.

Méta modèle

La modélisation des objets gérés par un dictionnaire est souvent représentée sous forme d'un modèle de type entité-relation, appelé aussi méta modèle ou modèle des modèles. Il décrit, dans ce formalisme, les objets utilisés dans une méthode, ainsi que leur articulation et certaines de leurs règles d'existence et de

contrôle. On trouvera également, dans les formes évoluées de ces méta modèles, l'association du formalisme et de la symbolique graphique représentant les différents objets.

La pluralité des méthodes incite souvent les fournisseurs d'ateliers de conception à proposer un accès au méta modèle de leur dictionnaire, permettant de personnaliser ou d'adapter le contexte descriptif de la méthode proposée, ainsi que certains comportements induits, notamment pour les contrôles propres à chaque modèle et à l'articulation inter modèles.

Cette ouverture du méta modèle devient également une nécessité pour accueillir des notions appartenant à d'autres outils, afin de procéder à des communications entre outils hétérogènes, comme le passage d'un outil de conception à un outil de réalisation. L'interface ou la passerelle entre ces outils pourra également faire l'objet d'un méta modèle, traduisant l'articulation entre les notions partagées.

Si cette vision du dictionnaire et l'aptitude à en modifier le contenu peut paraître séduisante, il ne faut cependant pas sous-estimer la difficulté et les dangers d'une personnalisation méthodologique mal maîtrisée. De plus, le comportement des outils ne s'adapte pas forcément fonctionnellement à toute évolution.

Types de dictionnaires et outils associés

La notion de dictionnaire se retrouve sous diverses appellations, recouvrant parfois des ambitions différentes. Ainsi les dénominations encyclopédie, base d'informations, gestionnaire banalisé d'informations, gestionnaire d'objets, référentiel, "repository" recouvrent à minima l'objectif de mémorisation des objets de conception et/ou de réalisation. Le caractère distinctif se révélera beaucoup plus dans l'organisation et les modalités d'accès à ces types de dictionnaires.

On peut ainsi distinguer trois grandes familles de dictionnaires qui induisent différentes architectures et différents usages des outils.

Les dictionnaires autonomes et communicants

Développés ces dernières années par nombre de promoteurs de la méthode Merise, ces outils ont assez tôt tiré partie de l'environnement de la micro-informatique et de la richesse de son interface utilisateur, pour proposer des fonctionnalités de modélisation, de contrôle et de documentation autour d'un dictionnaire couvrant l'ensemble de la méthode.

La plupart proposent également des points d'appui pour la réalisation, notamment par le maquettage d'écran et d'état, la transformation des modèles conceptuels de données en modèles logiques relationnels. On retrouve dans les

offres relatives à ces produits une architecture de postes de travail type micro autonome, ou des architectures réseau, pour partager les dictionnaires entre plusieurs utilisateurs.

Ces outils de conception disposent également fréquemment de procédures d'échanges sous forme d'import/export, pour établir des passerelles avec des outils de réalisation sur moyens ou grands systèmes. Les communications hétérogènes (ou interfaces) s'effectuent en général entre deux dictionnaires à travers une procédure de transformation permettant d'effectuer le passage d'objets de conception en objets de réalisation.

Les dictionnaires intégrés

Solution souvent proposée par des fournisseurs d'atelier de réalisation souhaitant remonter vers la conception, il s'agit d'outils organisés autour d'une structure de dictionnaire unique comprenant les objets de conception et de réalisation.

Ces outils ont donc dû arbitrer entre le niveau de modélisation associé à une méthode de conception et le niveau de spécification du logiciel; ce dernier conditionnant assez fortement par ses contraintes l'assemblage des deux niveaux, impose souvent une transition par des langages formels de spécifications.

La pratique peut mettre en évidence la difficulté pour ces outils d'avoir à la fois une large couverture du cycle de vie et une bonne performance sur chaque fonctionnalité. Ceci n'est pas surprenant, compte tenu :

- de la spécificité de chaque niveau abordé;
- des utilisateurs de métiers complémentaires, mais ayant des organisations et rythmes de travail différents;
- de l'impératif d'une cohérence permanente du dictionnaire.

Référentiels et plates-formes d'intégration

A la fin des années 80, une nouvelle perspective des ateliers de génie logiciel a été proposée, à travers des plates-formes d'intégration assurant :

- la gestion, le contrôle, la mémorisation de l'ensemble des composantes du système d'information;
- la normalisation de la perception du contenu des dictionnaires en rendant collectifs les méta modèles;
- l'utilisation d'outils différents et complémentaires, spécialisés dans une méthode ou une partie du cycle de vie, communiquant grâce à la normalisation des objets manipulés et reconnus par la plate-forme d'intégration.

Le dictionnaire, noyau de ces plates-formes, est dénommé *Référentiel* ou *Repository*.

L'intégration d'un outil et ses capacités à s'insérer dans une quelconque des phases de conception ou de développement sont donc assurées par une plate-forme commune, structure d'accueil d'outils différents, accédant aux données applicatives partagées. Ces données applicatives ont une structure et un format prédéfinis, par un modèle commun de données (ou méta modèle), et sont gérées par un dictionnaire centralisé : le *Référentiel*.

Par le passé, le monde de l'informatique et les luttes d'influence imposées par les conquêtes de marchés n'ont pas permis d'avoir des niveaux de normalisation suffisants pour assurer la portabilité des logiciels et la généralisation des méthodes.

L'approche par plates-formes d'intégration a fait immanquablement resurgir ces divergences (cf. l'échec d'AD CYCLE d'IBM). Cependant, si une normalisation du contenu des méta modèles paraît une nécessité pour une bonne communication entre outils hétérogènes, elle peut induire des effets réducteurs en matière d'innovation et d'évolution. .

Les grandes fonctions des ateliers de génie logiciel

On retrouve en général, articulées autour du dictionnaire, un ensemble de fonctions d'accès, de modélisation, de contrôle, de restitution, adaptées au formalisme des méthodes. Nous donnons ci-après quelques grandes orientations de ces fonctionnalités. On présentera plus loin et de façon plus détaillée un tel atelier de conception, l'atelier **Win'Design** développé par la société Cecima.

Aides à la conception

On regroupe sous cette appellation diverses fonctions qui aident le concepteur dans la mise en application des raisonnements, et plus particulièrement ceux de conception. On trouvera ainsi des outils assistant le concepteur dans :

- la manipulation des représentations graphiques des modèles;
- la vérification des règles de construction des différents modèles;
- la transformation d'un modèle en un autre (modèle conceptuel de données en modèle logique de données par exemple);
- le chiffrage et l'évaluation des volumes et activités;
- l'optimisation d'un modèle physique de données.

En règle générale, lorsqu'un raisonnement présente un caractère algorithmique ou semi-algorithmique, un outil logiciel d'aide à la conception pourra être utilisé.

Production de rapport

La vie du projet est jalonnée de multiples rapports destinés aux différents intervenants. Malgré le développement des outils logiciels et des supports informatisés des spécifications, le rapport écrit restera encore pour longtemps la forme conventionnelle de présentation des conclusions d'étude. L'essor de la bureautique a désormais banalisé le traitement de texte comme outil de production de documents.

La rédaction d'un rapport d'étude ne peut suffire à la restitution formelle des spécifications; il est indispensable d'y associer des éléments rédigés, permettant une compréhension plus large des idées exprimées. L'outil logiciel de production de rapports doit permettre :

- les fonctions classiques de mise en forme d'un traitement de texte;
- l'incrustation de fiches, tableaux ou listes en provenance de la documentation formelle gérée par le dictionnaire du système d'information;
- la manipulation de dessins (les schémas des modèles);
- l'indexation des éléments du rapport avec le dictionnaire afin de faciliter les mises à jour.

Simulation et maquettage

Ces types d'outils logiciels permettent d'illustrer et d'évaluer lors du processus de conception, le comportement du futur système d'information. Actuellement, les outils de simulation proposent une analyse du comportement dans le temps des modèles conceptuels ou organisationnels de traitements. Ces outils permettent de mettre en évidence les blocages éventuels, les points d'attente, et d'évaluer le découpage en tâches ou opérations dans le futur système d'information et d'en dimensionner les ressources.

Le maquettage permet d'illustrer le fonctionnement complet des données et traitements du futur système d'information. Il présente généralement les écrans associés aux tâches, les règles de traitements appliquées, l'utilisateur final peut ainsi faire part de ses réactions sur la maquette ce qui permet de confirmer, ou si nécessaire d'adapter, les spécifications.

Production de logiciel

Les outils logiciels correspondant à ce créneau se sont très tôt développés et bien souvent indépendamment des méthodes de conception de système d'information. Le développement d'applications dispose aujourd'hui d'un

ensemble d'outils regroupables sous le terme d'atelier de génie logiciel.

D'une façon générale, ces outils cherchent à accroître la productivité des réalisateurs, à apporter une meilleure portabilité du logiciel fourni et une facilité de maintenance. On retrouve généralement réunis dans cet atelier de production :

- des éditeurs de programmes sources multifenêtres;
- des gestionnaires de bibliothèques de modules de programmes et de classes d'objet ;
- des langages de programmation évolués ;
- des générateurs de programmes;
- des générateurs de jeux d'essai;
- des aides à la documentation;
- des descripteurs d'écrans;
- ...

Conduite de projet

Le passage d'une attitude artisanale à une démarche d'ingénierie pour la conception et la réalisation des systèmes d'information s'est accompagné de moyens pour la maîtrise des coûts et des délais. Là encore, des outils logiciels existent dans les domaines de l'ingénierie non informatique.

Il faut cependant adapter ces outils logiciels à la relative simplicité des projets à gérer (par comparaison à l'ingénierie pétrolière ou à des ouvrages d'art) ainsi qu'à la démarche spécifique.

On retrouve généralement dans ces outils de suivi de projet :

- l'ordonnancement et le suivi des délais des tâches et phases;
- la planification et le suivi des consommations de ressources;
- le suivi budgétaire de chacune des tâches ou phases;
- le contrôle et le classement de la documentation produite et utilisée.

Un exemple d'atelier de conception: Win'Design

Introduction

Win'Design conçu et développé par la société CECIMA, est le successeur du produit MESSAGE, dont la première version a été commercialisée en 1986.

Win'Design est ainsi la synthèse de nombreuses années d'expérience de mise en oeuvre d'outil de conception dans le cadre de projets très divers. Win'Design a su tirer partie à la fois des évolutions méthodologiques (extensions du formalisme de Merise, apport des méthodes orientées objet) et technologiques, aussi bien pour l'interface graphique (Windows®) que pour les environnements modernes de développement.

Outil indispensable de la mise en oeuvre de la méthode Merise pour l'approche système d'information, Win'Design est également un outil performant pour la construction des bases de données et le passage à la réalisation pour certains environnements de développement.

Utilisé dans tous les secteurs d'activité (administration, industrie, services, SSII,...), Win'Design est un atelier de conception complet, permettant d'aborder tous les modèles de données et de traitements. Ses nombreuses fonctionnalités constituent une véritable aide à la conception, aux contrôles, à la documentation dans le cadre de la mise en oeuvre de la méthode Merise 2ème génération, du schéma directeur à l'étude détaillée. Enfin, notons que Win'Design a été utilisé pour les illustrations et les modèles présentés tout au long de cet ouvrage.

Environnement de travail

Win'Design dispose d'un environnement de travail dont l'ergonomie est particulièrement adaptée aux taches de modélisation. :

L'espace de travail

« L'espace de travail », regroupe des modèles et des fichiers (par exemple des documents texte, des tableaux,..) ,autorise tous types d'organisation du travail (par modèle, projet, regroupement de projets,groupe de travail, sous ensemble d'activité,…).

Il permet ainsi d'adapter la visibilité des spécifications des systèmes d'information au contexte de travail des utilisateurs.

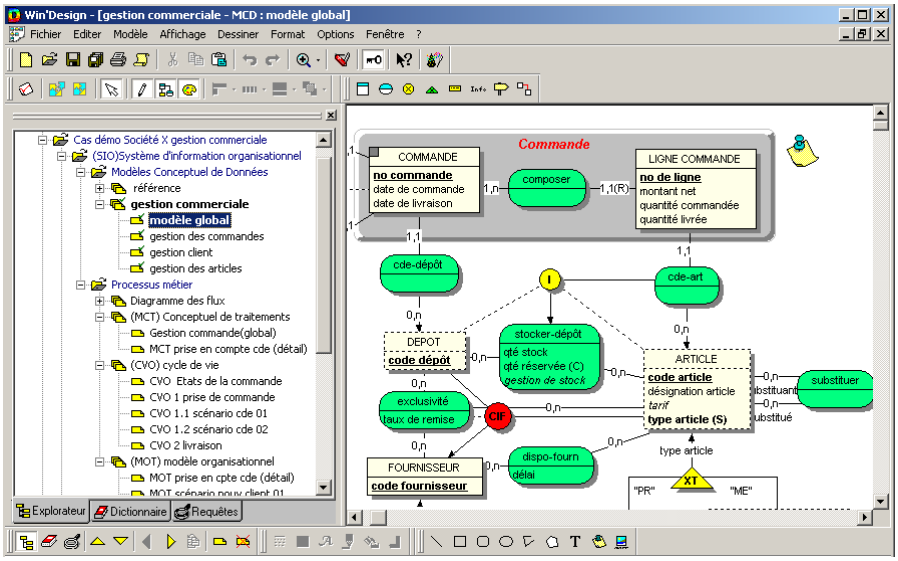


Figure 18.1: fenêtre principale de Win'Design avec l'espace de travail

Le dictionnaire

Il fournit la liste des tous les objets existants dans les modèles de l'espace de travail, avec leurs localisations (représentations multiples d'un même objet.).L'outil associé au dictionnaire permet également la recherche et l'affichage graphique de l'objet. , la réutilisation d'objets existants (par glisser déplacer sur le graphique).

Une liste complémentaire permet de visualiser toutes les références croisées de l'objet sélectionné (cf. figure 18.2).

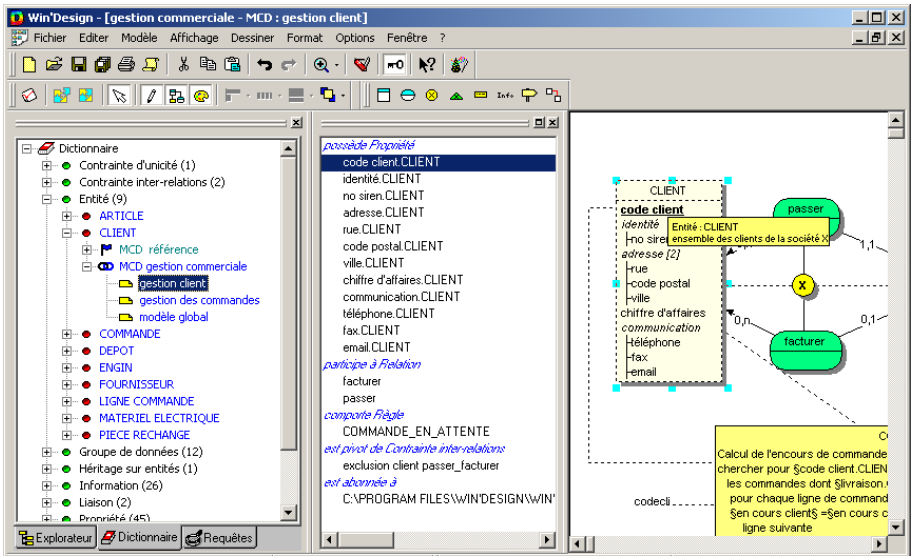


Figure 18.2: utilisation du dictionnaire d'objets

Contrôles et requêtes

Win'Design outre les contrôles immédiats lors de la construction du modèle, dispose d'une fonction de contrôles à la demande.

Ces contrôles concernent essentiellement les contrôles de complétude des objets dans le cadre d'un modèle.

Les contrôles peuvent être paramétrés et catalogués. Ainsi on pourra par type d'objet, indiquer quel est le type de contrôles à opérer et indiquer le degré de gravité. L'exécution des contrôles permet de disposer d'une fenêtre contenant les résultats, différenciant par couleur les degrés de gravité (erreur, alerte, information) avec inter-action entre la fenêtre de diagnostic et les fonctions de gestion des objets.

Ce dispositif de contrôle est complété par un système de requête multi modèles (exprimé en SQL simplifié)qui permet d'accéder sélectivement aux objets en fonction de ses caractéristiques ou liens (cf. figure 18.3)

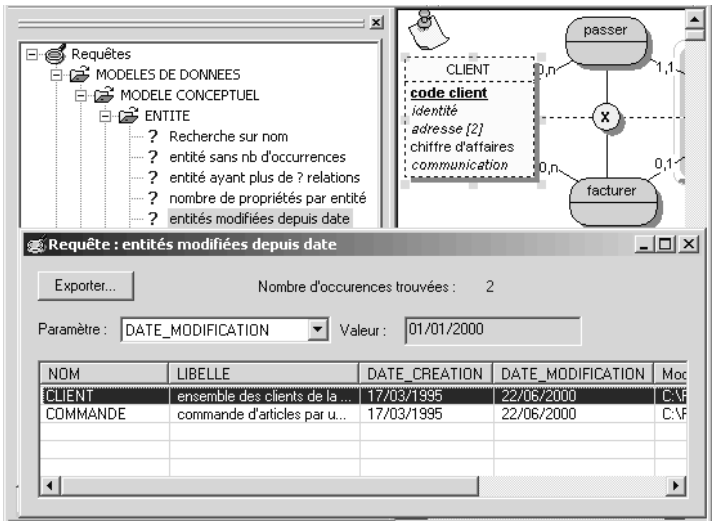


Figure 18.3: système de requêtes sur le dictionnaire

Documenteur

Les schémas peuvent être imprimés directement et être mis en page par une fonction spécifique, tenant notamment compte des spécificités de l'imprimante, portrait, paysage, échelle, sélection pages à éditer,…

Une autre fonction permet d'effectuer des extractions du dictionnaire et de présenter le dossier d'étude. Ce dossier est paramétrable dans sa forme (entête, pied de page, page de garde, table des matières,...) et dans son contenu (choix de l'édition, niveau de détail, sélection des objets,... cf. figure 18.4).

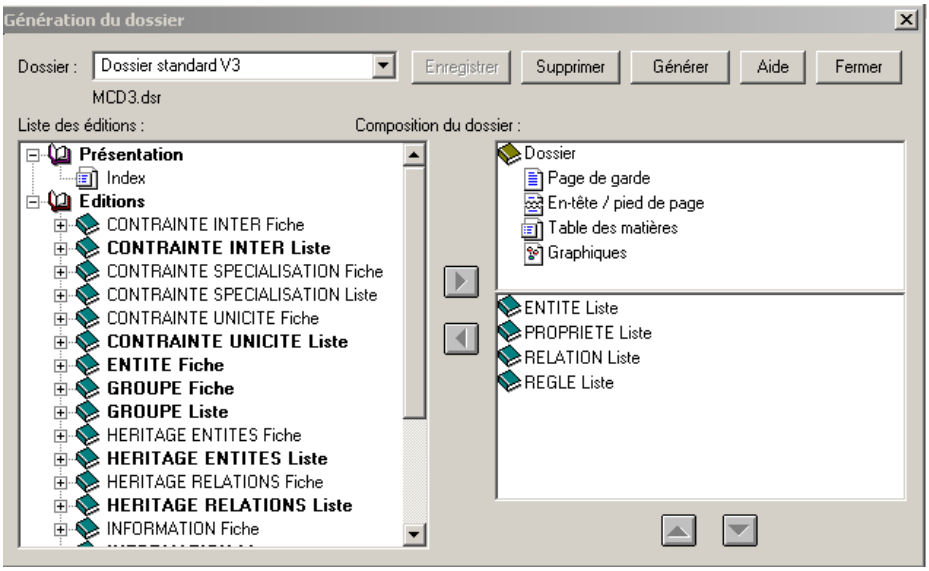


Figure 18.4: Gestion et paramétrage du dossier

Par ailleurs, la présentation des éditions peut faire l'objet d'une maquette réalisée à partir du traitement de texte Winword de Microsoft®, décrivant la forme et la structure de chaque édition. Ces maquettes peuvent être modifiées et personnalisées.

Fonctions spécifiques à chaque type de modèle

Modèle conceptuel de données

Win'Design prend en compte l'ensemble des notions du modèle entité relation étendu présenté dans cet ouvrage. Il permet ainsi de prendre en compte :

- *La généralisation et spécialisation* : Win'Design gère pour cela la relation d'héritage, l'héritage multiple, la contrainte sur sous-type, les critères de spécialisation,...(cf. figure 18.5);
- *Les contraintes :* Win'Design permet notamment la prise en compte des contraintes de stabilité (les relations définitives, les pattes verrouillées, les propriétés non modifiables), les contraintes d'unicité (dépendance fonctionnelle), les contraintes inter-relations (exclusion, inclusion, totalité, partition, simultanéité), les contraintes de valeur pour les propriétés (valeur minimale, maximale, par défaut, liste de valeurs, règles de détermination de valeur);

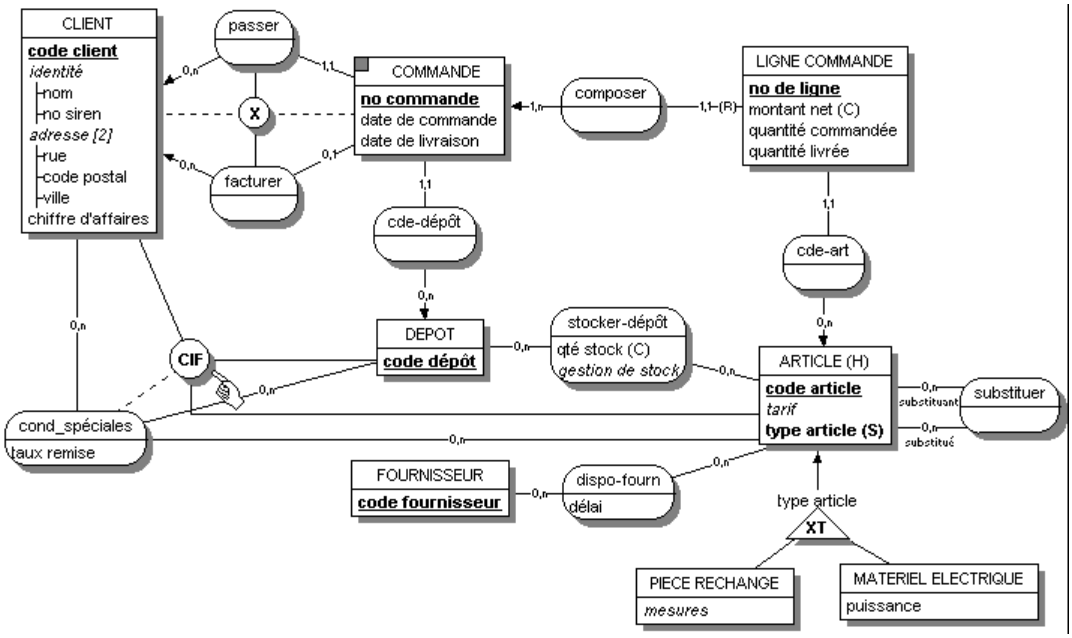


Figure 18.5: Exemple de M.C.D

- *Les règles :* Win'Design permet l'expression de n'importe quel type de règle (calcul, contrôle, sélection,...). Ces règles peuvent être dessinées dans le modèle et faire figurer les liens avec les propriétés utilisées dans la règle (cf. figure 18.6).

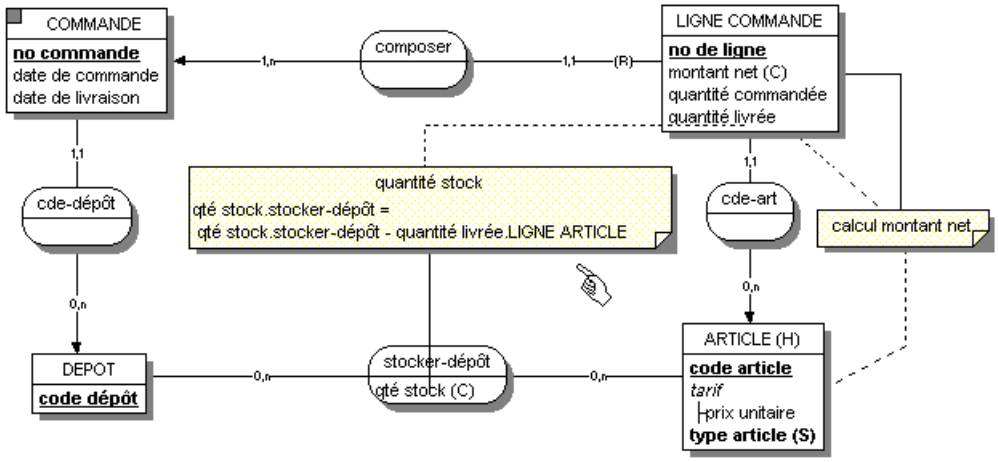


Figure 18.6: Modélisation des règles

- *L'historisation :* Win'Design permet pour les entités ou les relations de préciser le rythme et la nature de la conservation des dernières modifications, soit pour l'entité ou la relation dans sa totalité, soit pour l'une de ses propriétés. Cette modélisation est précisée par une

caractéristique d'unité de temps et de nombre de valeurs à conserver. (dans l'exemple l'entité "ARTICLE")

- *Les états associés aux entités ou aux relations :* Ces états permettent de traduire les différents stades de gestion perçus dans l'évolution du cycle de vie de l'entité ou de la relation. Ils sont organisés par typologie (cf. figure 18.7). Ces états peuvent être par ailleurs spécifiés dans les modèles de traitements soit à titre de pré ou de post condition d'un traitement, soit à l'occasion de la description du cycle de vie de l'objet lui-même, en entrée ou en sortie d'un traitement dit de transition.

Figure 18.7: Description des états d'une entité

- *Les propriétés composées :* Ces propriétés, (par exemple : adresse composée de rue, code postal, ville) peuvent être décrites jusqu'à trois niveaux de profondeur (cf. figure 18.8).

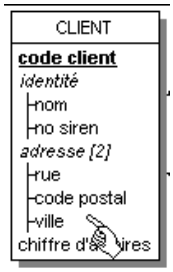


Figure 18. 8: Exemple d'une propriété composée et multivaluée

- *Les propriétés multi-valuées :* Ces propriétés permettent d'indiquer le nombre de répétition de valeurs de la propriété (intéressant pour ne pas nommer avec un indice les mêmes propriétés répétées plusieurs fois (exemple: pour décrire 2 adresses, au lieu de définir les propriétés Adresse1, Adresse2, il suffira d'indiquer sur la propriété le nombre de valeurs: adresse (2) (cf. figure 18.8).

Modèle Logique de données

Le modèle logique de données (MLD) est généré automatiquement à partir du modèle entité-relation (MCD/MOD). Cette génération prend en compte l'ensemble des caractéristiques de la modélisation du modèle conceptuel (cf. chapitre 13 "Modélisation logique des données") elles sont traduites au niveau relationnel, aussi bien au niveau de la description des tables et attributs que des clés primaires ou étrangères, des index, ainsi que les règles de contrôles d'intégrité référentielle (cf. figure 18.9).

Win'Design dispose également d'une fonction d'optimisation du placement graphique du modèle logique de données, fonction très utile notamment lors de la récupération par reverse d'une base de données existante.

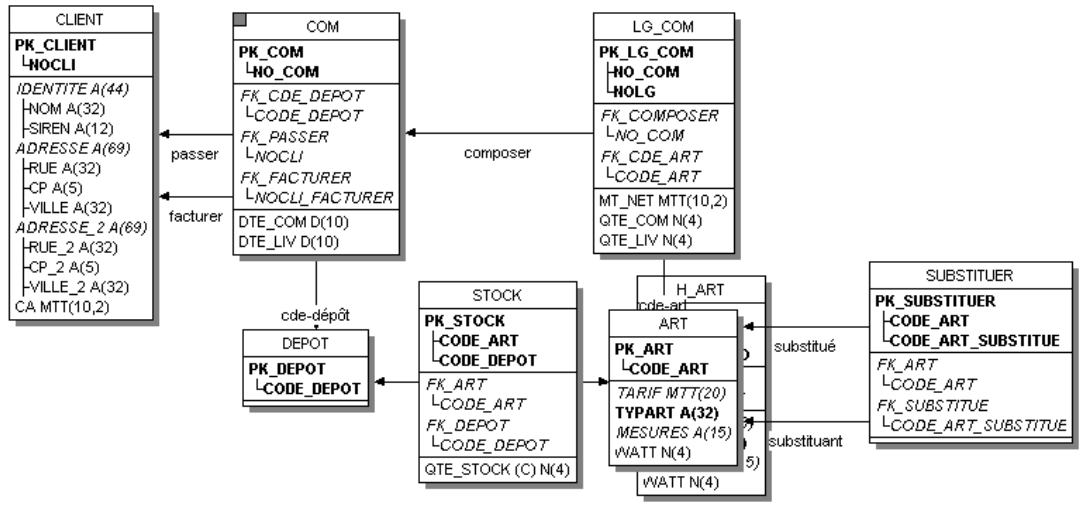


Figure 18.9: Exemple de M.L.D.

Au-delà de la transformation automatique, le modèle logique de données peut être modifié, éventuellement dénormalisé et géré comme tous les modèles de Win'Design. Win'Design permet également à partir d'un modèle logique de générer un modèle de type entité-relation correspondant.

Modèle Physique de données

Win'Design permet la génération automatique du modèle physique de données à partir du modèle logique, avec :

- *Paramétrage des pilotes (drivers) de génération des scripts :* Win'Design dispose d'une fonction permettant de paramétrer intégralement les principales règles syntaxiques ou les contraintes spécifiques à chaque SGBD, pour la création d'un script. Win'Design dispose en standard de la plupart des drivers pour les SGBD relationnels actuels.
- *Génération des scripts :* En fonction des caractéristiques du modèle logique

et du pilote de génération, Win'Design génère les scripts de création ou de modification des bases de données, ainsi que les descriptions de règles modélisées dans le MLR (cf. figure 18.10).

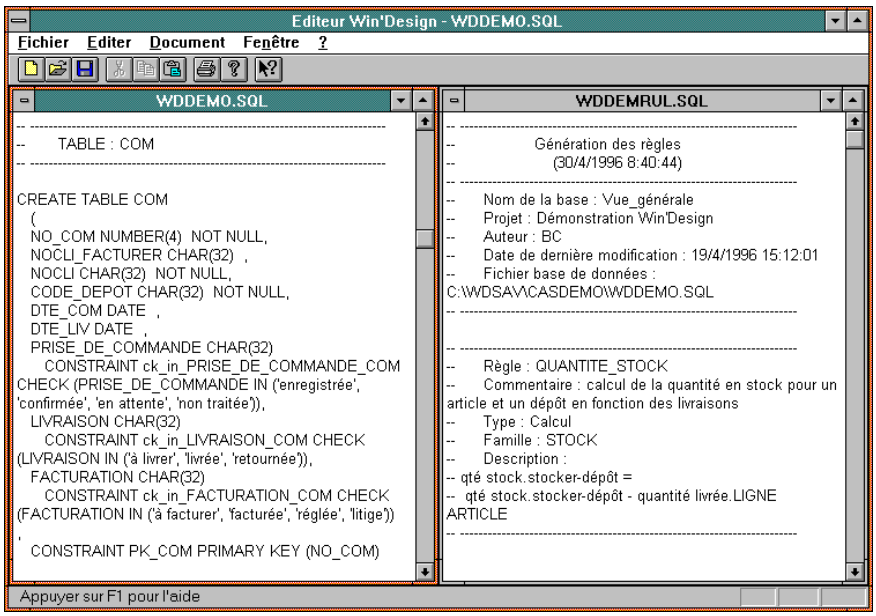


Figure 18.10: Exemple de script de création de base de données

Win'Design génère également les triggers d'intégrité référentielle pour les SGBD qui en tiennent compte (cf. figure 18.11).

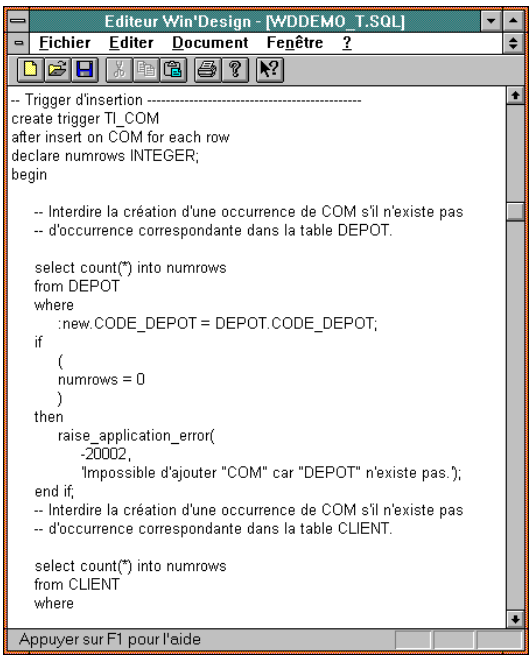


Figure 18.11: Exemple de génération de triggers

Reverse bases de données

Win'Design permet de récupérer le descriptif du contenu des bases de données à partir soit d'ODBC, directement de la structure physique de la base de données, soit du script lui-même, pour tous les SGBD suivant la norme SQL2.

Le modèle logique de données est alors déduit par Win'Design du modèle physique de données. Dans le cas où les clés étrangères n'existent pas dans le SGBD, Win'Design tente de les reconstituer. Le schéma peut être automatiquement optimisé dans son placement.

Win'Design propose aussi un découpage en "sous modèle" de l'ensemble du modèle, en fonction des tables les plus référencées par les autres ou des tables se référant à de nombreuses autres tables (correspondant en général aux notions majeures du modèle).

Les Modèles de Traitements

Win'Design prend en compte tous les modèles de traitements de la méthode Merise : Diagramme des flux, MCT, MOT, MLT, Diagramme d'état (cycle de vie) emprunté à l'approche "objet".

- Ces modèles, intégrés dans un même environnement, sont inter reliés et peuvent partager des concepts communs. Win'Design permet une unicité de définition et une réutilisation immédiate des objets communs aux différents niveaux d'abstraction (acteur, message, règle, poste ou unité organisationnelle, état). La figure 18.12 présente un MOT.

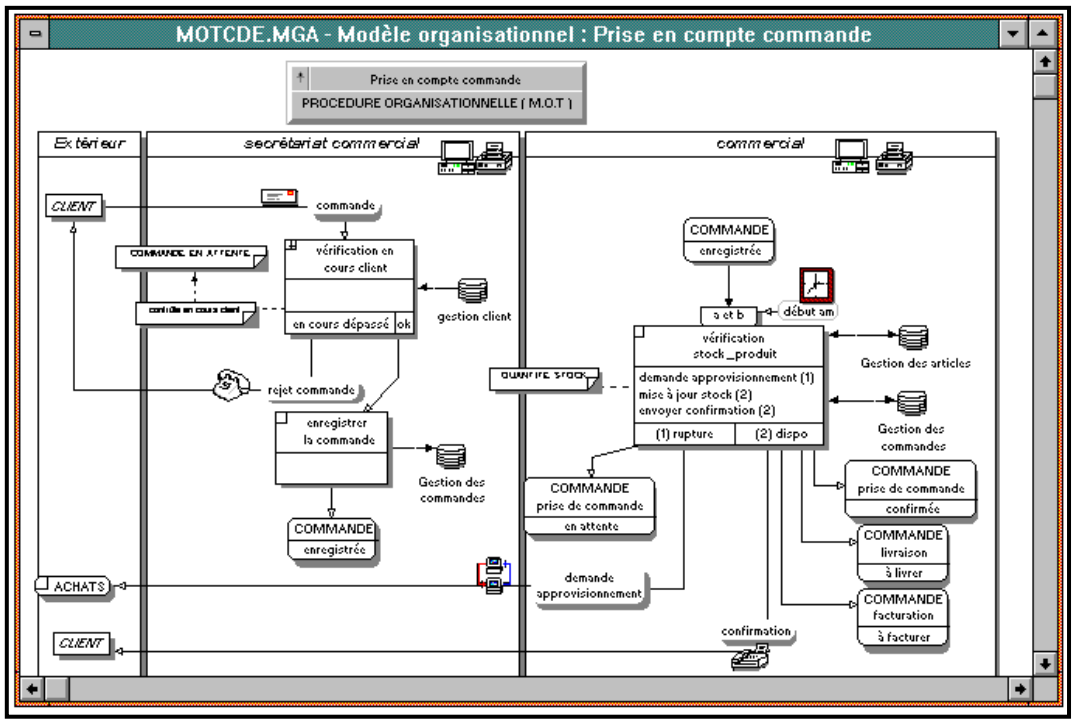


Figure 18.12: Exemple de M.O.T.

Conception par décomposition

La conception par décomposition est conforme aux approches anglo-saxonnes (SADT, SSADM,...). Win'Design permet ainsi :

- la décomposition d'un traitement dans un sous modèle présentant les traitements détaillés;
- la décomposition dans le même niveau d'abstraction ou avec changement de niveau;
- une aide à la décomposition d'un traitement à partir du détail de sa description et de son environnement (acteurs, messages, conditions,...);
- la décomposition à posteriori (mise à jour de l'environnement, rattachement);
- le passage automatique (aval et amont) entre un traitement et sa décomposition;
- la visualisation de la décomposition hiérarchique (gestionnaire de vues).

Association des modèles de données et de traitements

A chaque traitement (opération, tâche, unité logique..) peut être associé :

- *Le sous ensemble de données sur lequel il agit* : Ce sous ensemble est décrit dans le module de données de Win'Design et est représenté sous forme

d'icône dans les modèles de traitements. L'appel du modèle correspondant se fait par double clic sur l'icône représentant le sous-modèle de données, comme l'illustre la figure 18.13. Les actions (création, modification, lecture, suppression) peuvent alors être précisées pour chaque traitement sur chaque élément du sous-ensemble de données, appelée aussi "vue" ou "sous schéma».

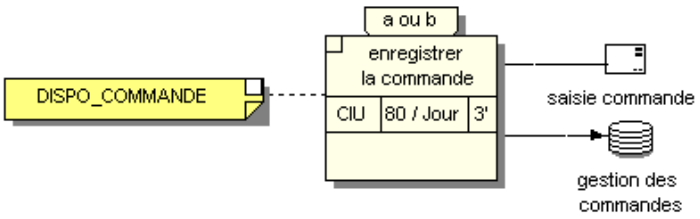


Figure 18.13: Accès aux données, association de règles et maquettes

- *les règles associées :* Les règles peuvent être définies dans les modèles de données indépendamment de leur contexte, et être réutilisées dans les modèles de traitements où elles sont associées aux traitements (contexte d'utilisation) .Elles peuvent être aussi définies directement à partir des modèles de traitements.
- *Les maquettes d'états ou d'écran :* L'association des maquettes aux traitements permet de resituer le contexte d'appel et d'exécution d'un écran ou d'une édition, la sélection de l'icône représentant une maquette provoquera l'affichage de celle ci

Maquettage de l'interface homme-machine

Les maquettes d'écran et d'état peuvent être spécifiées aussi bien dans leur forme, que dans leur contenu et comportement. (cf. figure 18.14). Plusieurs assistants permettent de s'appuyer sur les spécifications des autres modèles pour créer les maquettes (à partir des tables, vues logiques, vues externe, règles …)

Les objets dessinés (champ texte, liste déroulante, case à cocher, bouton, radio bouton, tableau, boite à onglet,…. objet externe de type « activeX »), sont associés à leur source de données (par référence au modèle logique de données).

Les comportements peuvent être spécifiés soit à partir des règles de gestion (cf. modèle de données), pour des calculs, contrôles,… soit à partir d'un éditeur Visual Basic, permettant de programmer le comportement évènementiel de l'objet et sa dépendance avec les autres objets. Dans ce dernier cas ,lors de la simulation de fonctionnement ,ces comportements sont réellement exécutés.

Par ailleurs, il est possible simplement de simuler le fonctionnement de la future application. En mode exécution chaque objet est activable avec son comportement standard et programmé, ceci permet d'évaluer et valider le fonctionnement d'un écran, des enchaînements d'écrans …

Enfin, la documentation des spécifications de ces maquettes constitue un véritable cahier des charges détaillé.

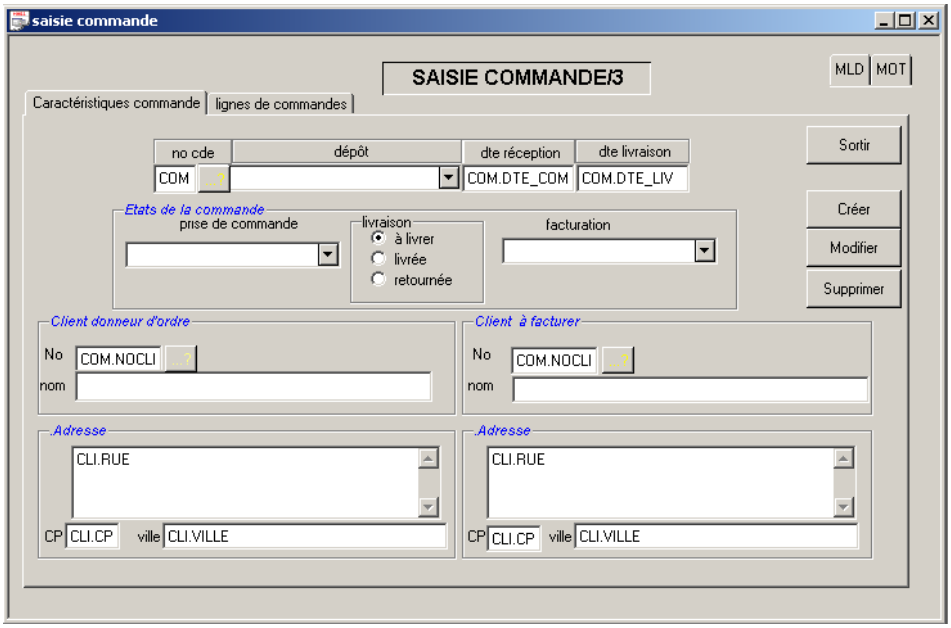


figure 18.14 : exemple d'une maquette d'écran

19

Merise et le BPR

Merise et la Renaissance ou la renaissance de Merise

Il y a des époques où tout change, où en l'espace de quelques années le monde se transforme de façon fondamentale et où les anciens paradigmes sont assez rapidement remplacés par de nouveaux modes de pensée et de vie. La Renaissance a été une telle époque de transition rapide vers un monde nouveau. De nombreux indices suggèrent que nous vivons actuellement une transition similaire, où les hypothèses de travail, qui étaient vraies il y a quelques années encore, sont profondément bouleversées et font place à des idées nouvelles.

Le but de ce chapitre est de montrer que Merise est l'enfant d'une telle transition, et si par certains aspects il est marqué par le paradigme ancien, il y a par contre d'autres facettes qui sont déjà annonciatrices du paradigme nouveau. Il faudra donc faire la part des choses et mettre en évidence les fondements sains qui pourraient constituer la base pour une nouvelle approche, tout en abandonnant d'autres principes qui n'ont plus aucun sens dans un monde en changement.

Certains y vont un peu vite et sont prêts à jeter l'enfant avec le bain : à leurs yeux Merise n'a plus aucune raison d'être à une époque où tout se décline en termes d'objets. Il n'est pourtant pas prouvé que la notion d'objet soit vraiment aussi fondamentale que certains ne le croient. Ce qui gêne le plus dans le concept d'objet, et même dans son nom, c'est qu'il est peu approprié pour tenir compte des êtres humains; à force de décrire des objets, on a tendance à oublier que le potentiel humain constitue la ressource la plus précieuse de l'entreprise et que toutes les technologies de communication et de traitement de l'information n'ont de sens que dans la mesure où elles permettent de mettre en valeur les compétences proprement humaines.

Or le courant managérial du Business Process Reengineering (B.P.R.) a mis en évidence le rôle des êtres humains et l'un de ses objectifs consiste justement à changer radicalement l'entreprise en vue de transformer les êtres humains-"objets" des organisations traditionnelles en êtres humains émancipés, pleinement responsables de leurs actions et jouissant d'une autonomie

suffisante pour mener avec succès leur propre projet d'entreprise.

C'est cette redécouverte du rôle fondamental assumé par l'être humain dans le tissu socio-économique qui n'est pas sans nous rappeler l'époque de la Renaissance. Le BPR, par de nombreux aspects, s'insère parfaitement dans cette vision nouvelle des choses, qui revalorise les compétences humaines, tout en mettant à sa disposition des outils bien plus puissants que tout ce qu'il a jamais eu entre les mains.

Le BPR constitue une première vague de changements, qui sont en train de transformer radicalement les entreprises. Une deuxième vague est constituée par l'expansion spectaculaire d'Internet et par l'émergence du commerce électronique, aussi bien entre entreprises qu'avec les consommateurs. D'autres vagues suivront, mais la méthode Merise a le potentiel de survivre à tous ces changements, car elle constitue un cadre de modélisation souple et adaptable au contexte technologique.

Le BPR, une première vague de changements

Il n'y a aucun doute que le BPR a fait son apparition avec les allures d'une mode ou d'une opération de marketing. Les théories propagées par Michael Hammer et James Champy dans de nombreux articles parus dans des revues réputées et surtout leur livre de référence *Reengineering the corporation : a manifesto for business revolution* [Hammer 93] avec ses thèses provocatrices ont soulevé suffisamment de poussière pour identifier le BPR à ces deux auteurs.

Selon leur définition, mille fois reproduite dans la presse, le "reengineering" serait *"une remise en cause fondamentale et une redéfinition radicale des processus opérationnels pour obtenir des gains spectaculaires dans les performances critiques que constituent aujourd'hui les coûts, la qualité, le service et la rapidité"*. En faisant miroiter des "gains spectaculaires" ils ont immédiatement attiré l'attention et la bienveillance des chefs d'entreprise, toujours fort sensibles à des perspectives de profit.

Les mesures radicales proposées par les deux auteurs se sont pourtant révélées être inapplicables dans la plupart des situations réelles, et après une phase d'euphorie on a rapidement dû déchanter et se fixer des objectifs plus modestes. Y. Tabourier attire d'ailleurs avec raison l'attention sur le fait que c'est uniquement dans la préface de l'édition française que Hammer et Champy mentionnent d'éventuels problèmes humains et sociaux liés à la cure radicale proposée par leur modèle de BPR virulent [Tabourier 94]. Ce qui fait que le BPR fut rapidement associé à un modèle d'organisation américain, n'ayant que peu de chance de succès dans le contexte socio-économique de "l'Ancien Continent".

Ce serait pourtant une grosse erreur que d'aborder le sujet de cette façon. Les idées fondamentales qui sont à la base du BPR sont bien plus anciennes et plus

solides que la présentation qui en est faite par Hammer et Champy. De nombreux auteurs américains et européens, tout en confirmant largement l'analyse de la situation par ces deux auteurs, proposent néanmoins des remèdes moins forts, qui ne risquent pas de provoquer le décès du patient par un choc thérapeutique. Des programmes de recherche ambitieux ont eu lieu au cours des années 80 dans les institutions les plus prestigieuses [Scott Morton 95] et ces études convergent largement vers une vision commune, celle d'une entreprise fragmentée, en réseau, dynamique, en changement permanent, par opposition à l'entreprise statique, hyperorganisée et complètement modélisée des années 70.

Cette dernière a nécessité la mise en place d'une base de données globale, intégrant pratiquement tous les aspects de l'entreprise et nécessitant des investissements importants en matériel et en logiciel. Le succès du modèle a fait croire en sa perennité, mais paradoxalement c'est la performance même de ce modèle qui a été à l'origine de sa remise en question.

En effet, la puissance des solutions informatiques mises en place après 1970 a rapidement mené à leur généralisation, engendrant de ce fait des marchés de masse, un accroissement spectaculaire des performances, associé à une chute tout aussi spectaculaire des prix. L'introduction de l'outil informatique s'est généralisée au niveau du poste de travail, envahissant progressivement la production, la conception de produits et la commercialisation de ces mêmes produits. Mais non seulement les vendeurs de produits et de services ont pu profiter de ce mouvement, mais aussi les clients (toute entreprise étant à la fois fournisseur et client). Les nouveaux moyens d'information ont progressivement transformé les clients passifs en sélectionneurs judicieux de fournisseurs, donnant la préférence à ceux qui le mieux pouvaient satisfaire leurs besoins spécifiques.

C'est ainsi que le client est devenu un acteur puissant et bien informé, engendrant un comportement nouveau de la part des entreprises, qui ont tout mis en oeuvre pour assouplir leur production et pour varier leurs produits en s'adaptant dynamiquement aux besoins changeants du client. Les technologies de l'information ont favorisé ce mouvement, mais les structures organisationnelles en place l'ont freiné.

Cette évolution a fait naître des réflexions fondamentales sur les structures organisationnelles adaptées à l'entreprise en mouvement. Il est vite devenu clair que les modèles mis en place par une informatique monolithique étaient bien trop lourds pour réagir assez rapidement aux changements de l'environnement socio-économique. L'implémentation de modèles gigantesques, à peine compréhensibles, avec spécialisation fonctionnelle des intervenants, ont engendré des circuits de travail complexes et longs. Les travaux circulent d'un expert à l'autre et chacun doit constater si oui ou non il y a une intervention à faire. Les temps de circulation et d'attente sont devenus

démesurés par rapport aux durées des interventions réellement productrices de valeur. De plus, aucun de ces experts ne se sent responsable envers le client, mais uniquement envers son chef.

Il n'est pas étonnant que dans ces conditions certains aient pu proposer d'oublier complètement l'ancien système, de recommencer à zéro et de réinventer un nouveau modèle d'entreprise. Le BPR à la Hammer et Champy en fut le résultat.

Mais en réalité le problème fondamental à résoudre est celui de trouver une structure d'entreprise adaptée au changement permanent. Toutes les théories récentes, toutes les solutions informatiques nouvelles ont pour but d'apporter un élément de solution à ce problème.

Et c'est en cela que consiste le renouvellement de paradigme actuellement en cours : nous passons d'une conception statique du monde, évoluant par sauts quantiques périodiques, à une conception dynamique, évoluant par des changements continuels. Le "Tout est Mouvement" de l'ancien philosophe grec Héraclite devient la trame sur laquelle se tisse la nouvelle entreprise. L'aptitude au changement devient un prérequis de base de l'entreprise dynamique et tout ce qui risque de freiner cette aptitude devra être examiné avec soin.

Parmi les facteurs d'inertie figurent les organisations fonctionnelles lourdes, la lenteur des mécanismes de restructuration interne, mais aussi les solutions informatiques dinosauresques, celles-là mêmes qui ont peu à peu déclenché cette avalanche de changements.

Le BPR consiste à mettre en place de nouveaux modes de restructuration dynamique et continuelle, permettant à l'entreprise de se métamorphoser rapidement en fonction des changements de son environnement.

En réinterprétant le BPR dans cette optique élargie, on doit se rendre compte qu'il ne disparaîtra pas de sitôt. Il changera peut-être d'étiquette, mais il ne disparaîtra plus, à moins que les changements ne cessent; mais ne serait-ce pas un peu comme la mort ?

Merise, un dépoussiérage nécessaire

Merise est un enfant de son temps, c'est-à-dire des années quatre-vingts. Il a puissamment contribué à concevoir les réalisations informatiques actuellement en place et avec la remise en cause de ces réalisations, l'outil qui a servi à les concevoir est lui aussi devenu suspect. On est tenté de le jeter avec les manuels de référence des matériels informatiques utilisés à l'époque.

Le problème, c'est qu'il n'y a rien pour le remplacer, et toutes les méthodes orientées objets proposées au fil des jours ne sont bien souvent que des méthodes de génie logiciel se préoccupant principalement des niveaux logique

et physique, à tel point qu'on en arrive à oublier la richesse originelle de Merise.

En effet, Merise prenait en compte l'entreprise globale, bien au-delà de considérations informatiques ou technologiques. En explorant les méthodes proposées en BPR on est toujours frappé par une constatation : "que ça ressemble fortement aux idées de base de Merise !" Et on en arrive à la conclusion que Merise, ce n'est peut-être pas aussi dépassé que certains voudraient bien nous le faire croire.

Le marteau du sculpteur et son burin peuvent servir à modeler des oeuvres classiques, mais rien n'empêche l'artiste moderne de les utiliser pour réaliser des sculptures avant-gardistes. Ce n'est pas l'outil qui a changé, mais le regard du créateur sur la matière première. Pour Merise il en va de même. Ce qui doit changer, c'est le regard du concepteur sur l'entreprise. Ce qui doit changer, c'est la façon d'utiliser l'outil, c'est la pondération entre les différentes facettes de la méthode.

Tout le monde sait que, suivant une tendance très forte à l'époque, les concepteurs de Merise ont nettement mieux développé le point de vue des données que le point de vue des traitements. Ils ont privilégié la statique par rapport à la dynamique à tel point que certains réduisent Merise à la modélisation des données. Les temps ont changé; le pendule est en train d'osciller dans la direction opposée et l'attention se porte désormais sur la partie dynamique du système. La question qu'on doit se poser maintenant est la suivante : "Comment les concepteurs de Merise auraient-ils développé la partie dynamique s'ils avaient été confrontés à la situation actuelle ?"

Un autre aspect fondamental, et une grande originalité de la méthode, c'est sa référence à la théorie des systèmes. Bien que rapidement oubliée par la plupart des utilisateurs, qui préfèrent la sécurité des dessins et des modèles à la réflexion systémique, cette dernière semble jouer un rôle de plus en plus important dans les modèles d'entreprises modernes. *L'entreprise-organisme* tend à éclipser *l'entreprise-machine* [Gouillart 95]. Les incertitudes de l'entreprise fractale [Warnecke 93 et 95] prennent le dessus sur la clarté de l'entreprise complètement modélisée et transparente.

Par ailleurs la théorie des systèmes elle-même a fortement progressé au cours des vingt dernières années, poussée en cela par les avancées spectaculaires des sciences de la vie [Maturana 87], mais aussi par l'ouverture progressive des sciences économiques et des sciences de gestion à cette théorie des systèmes [Bartoli 96]. Et de nouveau peut-on se demander comment ces nouvelles connaissances auraient été intégrées dans la méthode, si elle avait été développée aujourd'hui.

Ce qu'il faut donc, c'est un dépoussiérage, un polissage et une remise en valeur des fondements sains de Merise. Et tout d'abord il faudra faire l'examen de tout ce qui est utilisable dans une optique de BPR.

Les modèles Merise au service du BPR

L'approche systémique a amené Merise à étudier globalement des systèmes, en s'intéressant aux interactions du système avec son environnement. L'identification des sollicitations de l'entreprise par l'environnement est le point de départ de la description des processus, considérés comme la réaction de l'entreprise à un événement déclencheur externe. C'est ainsi que dès le départ Merise a préconisé une approche orientée *processus*, plutôt qu'une approche orientée *fonctions*; pour cette raison les modèles de description des processus (MCT) et des procédures (MOT) sont tout à fait adaptés à l'analyse préalable qui est à la base de tout projet BPR. L'objectif est le même dans l'analyse Merise que dans l'analyse BPR : comprendre les processus essentiels de l'entreprise.

Les méthodes BPR proposent généralement un mélange entre MCT et MOT (voir par exemple la "Business Activity Map" (BAM) et le "Relational Diagram" (RD) [Morris 93]).

Les apports du MOT

Le MOT permet de documenter clairement l'organisation de l'entreprise en vue de réagir à un événement déclencheur. Dans une optique BPR il permet d'atteindre deux buts :

- impliquer directement les intervenants internes dans la réflexion sur l'organisation, qui est rarement documentée et généralement peu connue des personnes concernées. Chacun connaît à fond les tâches qui lui sont confiées personnellement, sans nécessairement comprendre l'utilité de ces tâches dans l'ensemble du processus. Or l'utilité d'un travail intéresse fortement celui qui le réalise et il est généralement motivé pour prendre connaissance du modèle complet qui s'étend devant lui et qu'il a contribué à construire.
- montrer clairement les faiblesses de l'organisation actuelle. Le MOT laisse souvent une impression forte auprès des personnes concernées, parce qu'il dévoile impitoyablement la complexité et la lourdeur de l'organisation actuelle. Les acteurs concernés, et plus encore, les responsables, s'étonnent : "Est-ce vraiment comme ça que nous travaillons ?". Cette révélation est indispensable pour déclencher le processus de changement et pour solliciter l'adhésion des personnes concernées à ce processus.

L'étude du MOT permet de détecter rapidement certains dysfonctionnements de l'entreprise :

- tâches qui sont réalisées plusieurs fois par des postes de travail distincts;
- tâches "vides" où l'affaire est simplement transmise à l'intervenant

suivant, après avoir attendu patiemment dans la file d'attente des entrées d'un responsable donné;

- tâches qui ont pour seul but de rendre les données compatibles aux solutions informatiques, alors que les solutions informatiques devraient s'agencer autour des flux naturels des travaux à réaliser par les intervenants.

Des MOT quantifiés en durée déterminent les temps d'attente, les temps de transmission et les temps productifs et permettent de calculer certains paramètres importants :

- la durée totale de traitement, comme étant la somme de toutes les durées d'une occurrence de procédure;
- la durée de travail productif, comme étant la somme de toutes les durées productives.

On peut ainsi calculer des ratios critiques des procédures :

$$\text{taux de traitement productif} = \frac{\text{durée du travail productif}}{\text{durée totale de traitement}}$$

Un taux inférieur à 80 % indique qu'il y a des améliorations à réaliser, un taux inférieur à 50 % (et c'est souvent le cas) montre qu'il y a des problèmes d'organisation à traiter d'urgence.

Aujourd'hui le client ne veut plus attendre, s'il sait que la concurrence réalise un travail de même qualité dans des délais plus satisfaisants. La réduction des délais devient un avantage concurrentiel. L'entreprise dynamique aura tendance à se restructurer jusqu'à réaliser des délais minimaux, en d'autres termes, elle essaiera d'améliorer son taux de traitement productif.

Certaines entreprises essaient d'améliorer le taux de traitement productif en automatisant les procédures actuelles par des systèmes de gestion de flux de travaux informatisés ("*workflow systems*"). Cette façon naïve de résoudre le problème peut se révéler dangereuse. C'est une solution qui essaie d'agir sur les symptômes, plutôt que de curer la maladie elle-même, et une automatisation des flux de travaux actuels, sans remise en cause des processus, n'est pas à considérer comme du BPR. Au contraire, l'automatisation de processus mal conçus peut constituer un désavantage concurrentiel, car elle améliore essentiellement les durées de transmission, sans influencer les durées d'attente et les durées productives. Par contre ces systèmes augmentent le pouvoir des organes de contrôle et de supervision, en fournissant des statistiques détaillées sur le comportement des intervenants, qui se sentiront surveillés et qui auront tendance à s'opposer et à bloquer le système, ou bien à le contourner en réduisant la qualité de la production au profit de la quantité.

En BPR par contre, on essaie de réduire les contrôles et d'accroître la qualité de la production en créant un climat de confiance mutuelle, de motivation

pour l'amélioration continue des procédures et de responsabilisation des intervenants. Pour atteindre ce but, il faut comprendre les processus de façon plus approfondie et à cette fin le MOT s'avère insuffisant.

La quantification des durées productives peut rendre apparents d'autres problèmes : souvent il est difficile de quantifier la durée productive, car elle varie fortement en fonction de la complexité du cas à traiter. On a intérêt à réaliser un histogramme faisant apparaître la répartition des durées productives. L'histogramme ci-dessous montre que 80% des cas de cet exemple sont des cas simples, 15% des cas complexes et 5% seulement des cas très complexes. Ceci met en évidence qu'on soumet au même intervenant des cas simples et des cas complexes, ce qui est un symptôme typique d'organisation par spécialisation fonctionnelle. Elle a le désavantage d'engendrer des chaînes de traitement longues et de multiplier des transmissions entre postes de travail d'experts fonctionnels.

En BPR on propose de réagir aux cas simples et aux cas complexes par des procédures séparées, ce qui permet de réduire nettement le nombre de postes de travail qui interviennent dans le traitement d'un cas simple, l'idéal étant de confier l'ensemble du cas à un seul intervenant polyvalent. Par contre pour les cas complexes il faudra faire intervenir des techniques de coopération assistée si possible par ordinateur.

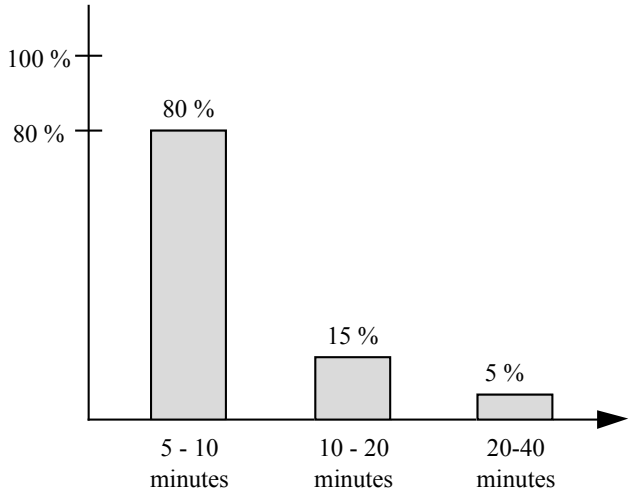


Figure 19.1 : Répartition des durées productives

Le MOT permet d'un seul coup d'oeil de caractériser la structure organisationnelle et ses faiblesses. Souvent la procédure a une structure en cascade. Les intervenants travaillent en chaînes de traitement, réalisent chacun une tâche déterminée et ne se sentent pas responsables de la tâche du prédécesseur ou du successeur. Le désavantage de ce type d'organisation, c'est que personne ne se sent concerné par l'ensemble de la procédure, que des contrôles de qualité sont nécessaires, que des délais de transmission et d'attente

apparaissent en fonction de la disponibilité des intervenants et que personne ne connaît l'état d'avancement de l'affaire. De plus le système est incapable de réagir rapidement à des imprévus et risque d'être écarté à terme des marchés intéressants.

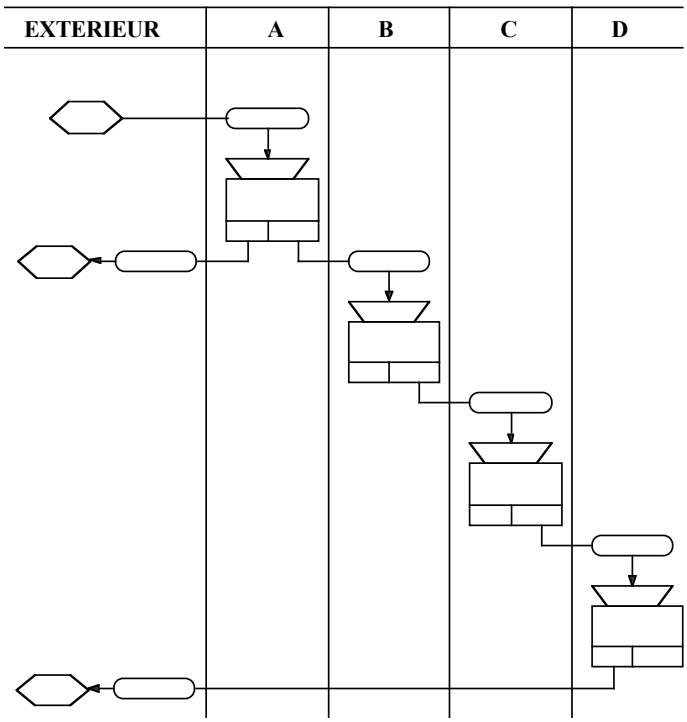


Figure 19.2 : Une procédure en cascade

Les systèmes de gestion de flux de travaux permettent de remédier à cet inconvénient en localisant en temps réel l'état de chaque affaire. Par contre ils ne permettent guère d'accélérer ou de modifier le cours des traitements.

Une organisation améliorée consiste à prévoir un responsable de procédure qui s'occupe complètement des relations avec le client et surveille l'avancement de l'affaire dans la procédure.

Tout se passe comme si le responsable de procédure devenait à son tour client des spécialistes fonctionnels et qu'une relation client/fournisseur s'établissait entre eux. L'entreprise se structure en "front office" et en "back office".

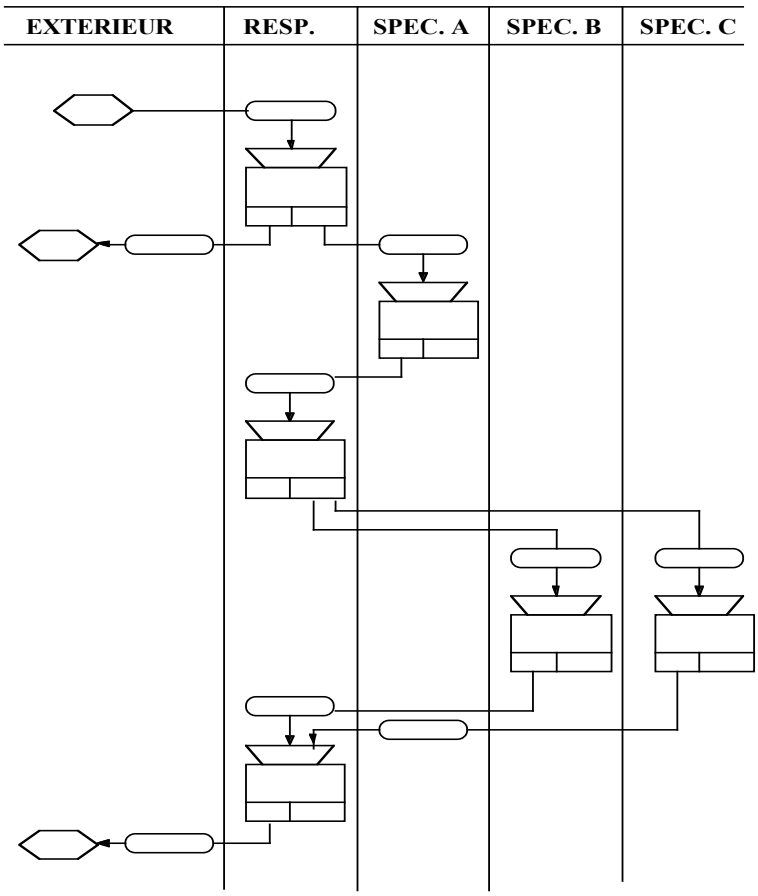


Figure 19.3 : Un responsable gère la procédure

Les apports du MCT

Faire du BPR en se basant uniquement sur des MOT ne peut amener que des améliorations ponctuelles de type organisationnel. Pour aller plus loin il faut être prêt à se détacher complètement de l'organisation actuelle et se poser des questions plus fondamentales sur les processus managériaux. Le MCT Merise est tout à fait approprié pour documenter les réflexions sur ce qu'on fait réellement lors d'une prestation pour le client, indépendamment de la façon dont les choses sont organisées actuellement.

L'entreprise est considérée comme un interlocuteur unique qui essaie d'offrir un service optimal au client. Réaliser un MCT est plus difficile qu'on ne le croit généralement, car l'organisation actuelle a tendance à imposer sa présence au modélisateur, qui souvent ne s'en rend même pas compte. Pour faire un bon MCT on doit considérer l'entreprise comme système et se libérer de toute l'organisation du stockage de l'information. La tentative même de s'engager dans cette voie présuppose un état d'esprit qui conduit directement à des questions plus fondamentales encore. A la recherche du "quoi ?" se substitue

rapidement la quête du "pourquoi ?" et poser la question revient souvent à remettre en cause l'état actuel.

Merise à l'origine se contentait d'explorer le "quoi ?" et faisait de la reconfiguration de processus en tenant compte uniquement de contraintes ou d'opportunités nouvelles.

Le passage du modèle conceptuel du système existant vers le modèle conceptuel du système futur se fait au zénith de la "courbe du soleil" souvent citée dans la démarche Merise. Pourtant peu d'ouvrages Merise expliquent ce qui se passe à ce niveau et généralement on suppose que les choix de gestion à réaliser soient déjà connus (souvent on les considère comme étant les déclencheurs de l'étude Merise).

Ce point de vue peut se justifier dans un contexte d'entreprise statique, qui évolue par mutations périodiques et où les changements de règles sont rares. Dans une entreprise en changement permanent les règles doivent être adaptées dynamiquement en fonction de l'évolution de l'environnement. Lorsque l'entreprise co-évolue avec son environnement, le processus de changement doit se réaliser de façon plus systématique et trouver sa place normale dans le cycle de vie. Le BPR est le mécanisme qui assume le changement et doit trouver sa place normale au sommet de la courbe du soleil.

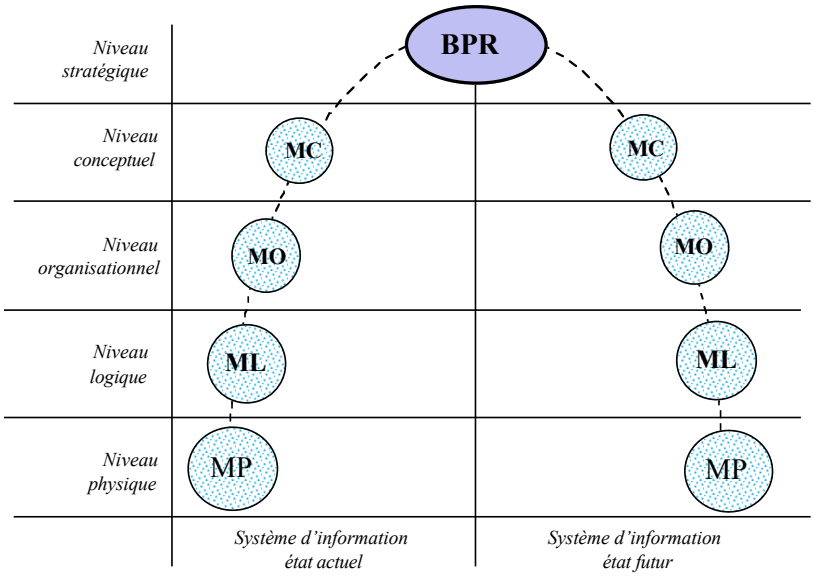


Figure 19.4 : Le BPR au sommet de la courbe du soleil

Le BPR devient ainsi le moteur de l'évolution de l'entreprise. Il se base sur les modèles conceptuels existants et les adapte dynamiquement pour fournir régulièrement des directions de changement. La mise en oeuvre rapide de ces directives est essentielle pour le succès du BPR, car le temps qui s'écoule entre l'émission de la directive et son activation au niveau physique constitue le délai de réaction de l'entreprise face aux changements structurels de

l'environnement. Ce délai dépend largement d'outils qui accélèrent le passage du niveau conceptuel vers le niveau physique. Les directives produites par le BPR ne déterminent pas seulement les choix de gestion, mais peuvent aussi se répercuter sur des choix organisationnels, voire même sur des choix technologiques.

Les mécanismes du BPR

Les préoccupations du BPR se situent à un niveau stratégique du système étudié. Bien que les modèles conceptuels Merise constituent un bon point d'entrée au BPR, ils ne suffisent pas pour soutenir les raisonnements qui se font à ce niveau.

La démarche du BPR est souvent présentée en trois phases successives :

- la phase de positionnement,
- la phase de repositionnement et
- la phase de reconfiguration.

La phase de positionnement

Le but de cette phase est de comprendre le système actuel et de mettre en évidence ses dysfonctionnements. Les modèles organisationnels et conceptuels Merise sont des outils d'analyse fort appropriés qui permettent de répondre à la question : "Que faisons-nous et pourquoi le faisons-nous ?" Les réponses à ces questions permettent de comprendre le fonctionnement du système actuel. Il s'agit ensuite de procéder à une sorte d'auto-évaluation : "Quelles sont nos points forts et nos points faibles ? Quels sont les procédés que nous maîtrisons le mieux ?" La réponse à ces questions ne peut pas se faire par modélisation des processus de l'entreprise, mais nécessite une confrontation aux activités de la concurrence. Il s'agit en réalité de se positionner par rapport à cette concurrence et d'évaluer les chances de survie sur les marchés en changement rapide. Les techniques de positionnement sont hors du domaine d'applicabilité de Merise et font appel à des études de marché et à des raisonnements stratégiques. Le résultat de la phase de positionnement consiste en une vue réaliste de la situation de l'entreprise dans l'environnement socio-économique.

La phase de repositionnement

Lorsque la position de l'entreprise est devenue plus claire, il s'agit alors de produire une vision du futur. "Où voulons-nous être positionnés à l'avenir en tenant compte de nos compétences de base ?" Les réflexions qui sont menées dans cette phase mènent à une redéfinition des objectifs de l'entreprise. Souvent elles sont accompagnées d'un recentrage sur les compétences essentielles et de l'abandon d'activités accessoires dans lesquelles on s'était engagé sur la base d'une tendance générale à la diversification économique. On

en arrive à la conclusion qu'on ne peut pas tout faire et qu'on a uniquement des chances de survie dans les domaines où l'on brille par son excellence et par l'avance sur la concurrence. Progressivement une vision nouvelle de l'entreprise se dégage et il s'agit dès lors de transformer la vision en réalité.

La phase de repositionnement est une activité hautement stratégique et Merise n'y intervient guère jusqu'à présent. Pourtant rien n'empêcherait de le faire. En effet l'approche systémique de Merise se prête de façon naturelle à l'étude des finalités d'un système, lui-même décomposé en sous-systèmes, auxquels correspondent des sous-finalités. La modélisation Merise devrait donc décrire la hiérarchie des finalités, partant de la vision d'entreprise, en passant par des missions, jusqu'à aboutir à des objectifs opérationnels quantifiables. Chaque finalité devrait se décrire comme un couple de valeurs : l'objectif à atteindre et le résultat effectivement atteint par le système. Des initiatives pour compléter la modélisation des traitements et des données par une modélisation des finalités ont déjà été proposées [Marshall 2000].

Dans la phase de repositionnement *("reframing")* on est parfois amené à redéfinir les frontières du système. L'abandon de certaines activités non essentielles, des accords d'association avec d'autres entreprises, des solutions de délocalisation et de sous-traitance font que l'ancien système ne peut plus rendre compte de l'entreprise nouvelle. Les frontières entre entreprises tendent à s'estomper et on se dirige vers l'entreprise étendue ou l'entreprise en réseau. Il en résulte que l'ancien schéma directeur basé sur des frontières rigides n'est plus applicable.

Là encore Merise montre sa force. L'approche systémique se prête aussi bien à décrire une entreprise monolithique qu'un ensemble dynamique de sous-systèmes couplés et associés dans une évolution commune. Pour cela il faudra élargir la vue et décrire globalement le système étendu ainsi défini. Les processus sont maintenant décrits au niveau du réseau et non plus au niveau de l'entreprise isolée. Le BPR exige donc un nouveau schéma directeur sur base de frontières de système nettement étendues et en faisant intervenir des décideurs et des concepteurs répartis géographiquement. La réalisation d'un tel schéma directeur multi-sites, multi-sociétés exige la mise en oeuvre de techniques coopératives nettement plus performantes que celles utilisées aujourd'hui. Les AGL futurs devront intégrer des fonctions coopératives et permettre aussi de modéliser cette coopération.

La phase de reconfiguration

Elle consiste à mettre en place une organisation pour l'entreprise repositionnée. Les lignes directrices qui déterminent la nouvelle organisation sont le service rendu au client et la production de valeurs. Le client n'est plus considéré comme un facteur externe qui de temps en temps lance une commande, mais plutôt comme un partenaire de l'entreprise en réseau. Il s'agit de le comprendre et de tenir compte de ses besoins particuliers. Dans cette

optique chaque commande est différente de l'autre et doit faire l'objet d'une négociation entre client et fournisseur. Il ne suffit plus de modéliser une procédure en représentant l'enchaînement des tâches à l'intérieur de l'entreprise, mais il faudra modéliser correctement l'interaction avec le client.

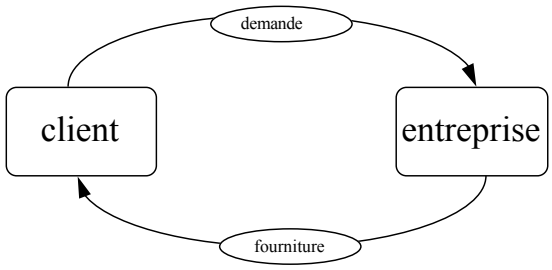


Figure 19.5 : L'interaction entre le client et l'entreprise

En réalité cette interaction est complexe et la boucle demande-fourniture peut faire l'objet de nombreuses itérations et de longues tractations. De fait il s'agit d'un processus de négociation, où il faudra se mettre d'accord sur les termes de l'échange et sur les conditions à satisfaire avant que la demande du client ne soit acceptée. Similairement il s'agira de négocier la réception de la prestation et de prouver, après fourniture, que les termes de la transaction ont bien été respectés.

Les modèles de procédure et de processus actuels ne sont guère appropriés pour représenter des situations de transactions négociées. Medina-Mora, Flores et Winograd proposent un modèle d'interaction client/fournisseur en 4 phases : la préparation, la négociation, la prestation et l'acceptation, comme l'illustre la Figure 19.6 [Medina-Mora 92] :

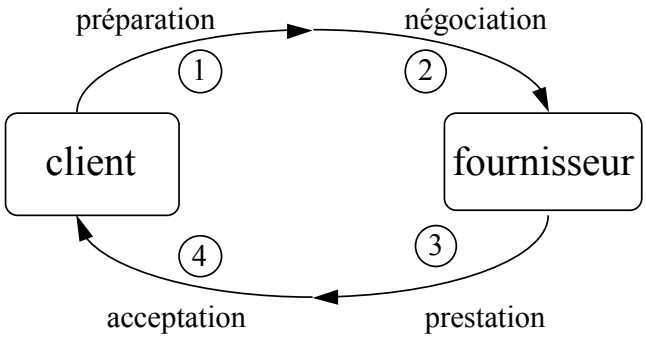


Figure 19.6 : Les phases du modèle d'interaction

Dans la phase de *préparation*, le client définit sa demande ; durant la phase de *négociation*, il se met d'accord avec le fournisseur sur les termes de la transaction. Cette phase peut être complexe et passer par plusieurs états intermédiaires avec propositions et contrepropositions, jusqu'à ce que la phase de *négociation* aboutisse ou cesse. La phase de *prestation* correspond à toutes les tâches nécessaires à la satisfaction du marché conclu ; elle correspond à ce qui est

normalement représenté dans une procédure. La phase *d'acceptation* a pour but de clôturer la transaction : ceci ne sera le cas que lorsque le client a obtenu satisfaction et que les termes négociés ont été respectés.

Le modèle est intéressant à plus d'un égard :

- tout d'abord, en mettant l'interaction avec le client au centre des considérations, il oblige le modélisateur à définir des procédures orientées-client et à définir clairement les responsabilités de chacun ainsi que l'objectif et les conditions de l'interaction.
- ensuite, le fournisseur comme le client pourra recourir aux services de tiers pour accomplir l'une ou l'autre phase de la transaction. En particulier le fournisseur lui-même aura souvent besoin de sous-traiter certains éléments de la négociation ou de la prestation à d'autres entités organisationnelles de l'entreprise fournisseur ou du réseau d'entreprises. Cette sous-traitance fera l'objet d'une boucle d'interaction entre le fournisseur et son sous-traitant.

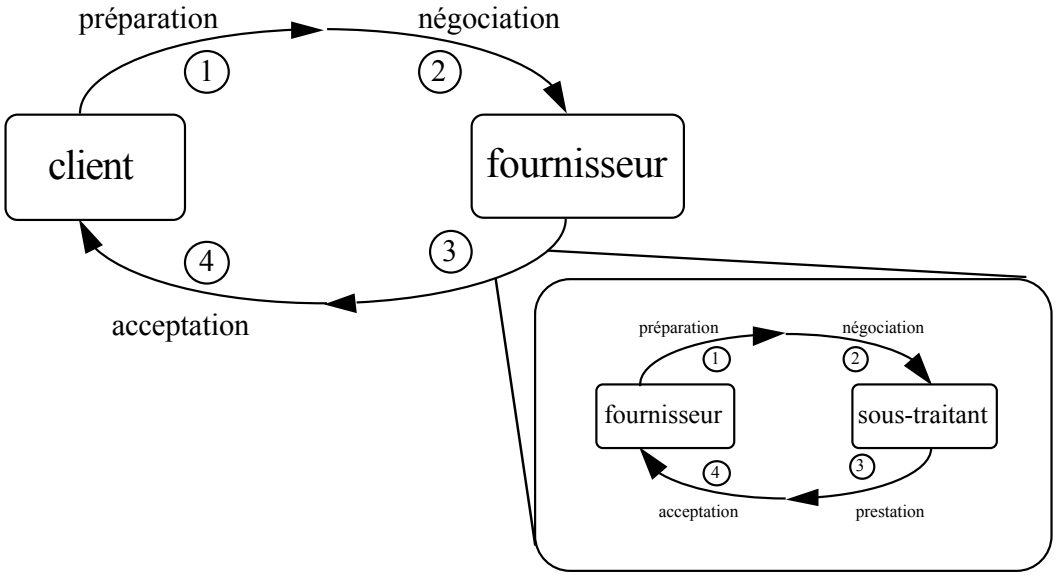


Figure 19.7 : Une boucle d'interaction secondaire

- De cette façon toutes les interactions entre éléments de l'entreprise-réseau sont décrites par des modèles de transaction entre acteurs jouant le rôle de clients et d'autres jouant le rôle de fournisseurs, chaque fournisseur produisant une certaine valeur-ajoutée acquise par le client en fonction des termes d'un contrat (réel ou virtuel). Une telle boucle d'interaction constitue un processus élémentaire et tout processus d'entreprise pourra se décrire comme réseau de boucles d'interaction [Marshall 2000]. Cette façon de modéliser les interactions permet de décrire des situations très variées, où il suffit de remplacer une

transaction par une autre pour que le comportement du système change fondamentalement.

- Certaines transactions, surtout celles avec les clients nouveaux, feront l'objet de négociations serrées, d'autres se dérouleront en fonction de contrats-cadres négociés entre deux partenaires et chaque occurrence de transaction se déroulera suivant un schéma prédéterminé, faisant largement appel à des tâches automatisées. Il est intéressant de noter qu'une transaction peut à la limite représenter une interaction entre un être humain et un ordinateur, ou bien une interaction entre deux ordinateurs, permettant ainsi de faire appel à la même métaphore de représentation pour les transactions commerciales que pour les transactions informatiques du type client-serveur.
- Le modèle des diagrammes d'interaction permet de tenir compte de l'évolution des relations au cours du temps. Une boucle client/fournisseur se transformera progressivement en boucle fournisseur/sous-traitant, dans la mesure où les relations commerciales se stabilisent et qu'un climat de confiance mutuelle, basé sur la qualité des interactions passées, s'établit entre partenaires.
- Enfin, au fur et à mesure que le comportement entre un client et un fournisseur déterminés se standardise, il en sera de même des messages échangés entre eux et on pourra alors effectivement appliquer la métaphore "objets de métier" : tout se passe comme si l'échange entre client et fournisseur était devenu un échange entre *l'objet-client* et *l'objet-fournisseur* communiquant par des messages standards définis d'un commun accord. Mais il faut bien comprendre que c'est l'aboutissement d'un long processus d'évolution et non pas la situation habituelle. Le modèle objets ne devient applicable que lorsque les interactions humaines sont devenues superflues ou se déroulent suivant un code de comportement strictement réglementé; c'est probablement le cas dans les couches basses du modèle de communication.

Dans un monde en changement rien n'est définitif et toute relation pourra se transformer rapidement, voire même cesser complètement. Voilà pourquoi le modèle d'interaction sera en mouvement permanent. La technologie évoluera, les produits changeront, la concurrence s'améliorera et il faudra donc que l'entreprise-réseau soit capable de se remettre en cause régulièrement et ait mis en place un mécanisme d'évolution.

Les mécanismes d'adaptation

L'évolution d'un système d'information se fait par mutations espacées dans le temps au point qu'on a pu utiliser le terme de "cycle de vie" pour désigner ce mécanisme d'adaptation. A la fin d'un tel cycle le système d'information est remodelé afin de tenir compte de changements importants de l'environnement.

Mais dans la mesure où l'environnement change plus rapidement, la durée de vie des produits logiciels devient de plus en plus courte et la vitesse de production doit s'accroître pour rester en accord avec les besoins des utilisateurs. La notion de cycle de vie tend à perdre sa signification et l'évolution du système d'information devient une activité continue.

Ceci ne signifie pas pour autant qu'une telle évolution soit chaotique. Au contraire, l'entreprise en changement ne survivra que lorsqu'elle maîtrisera son évolution et deviendra capable d'assimilier rapidement les opportunités technologiques dans sa structure organisationnelle. Nous présentons brièvement la démarche de l'alignement stratégique développée à la Sloan School of Management du MIT [MacDonald 95].

La démarche de l'alignement stratégique

Le BPR a permis à l'entreprise de se repositionner et de reconfigurer ses processus. Le nouvel état ne sera pas en équilibre statique, mais devra plutôt être considéré comme un état d'équilibre dynamique, un peu comme le randonneur qui, dès qu'il a posé un pied sur terre, y prend appui pour lever l'autre pied en vue d'atteindre un nouvel état d'équilibre. Le BPR doit devenir un état d'esprit, une acceptation du mouvement. Il faudra donc se repositionner régulièrement et par la suite reconfigurer l'organisation et adapter le système d'information.

La démarche de l'alignement stratégique se fait suivant quatre axes différents :

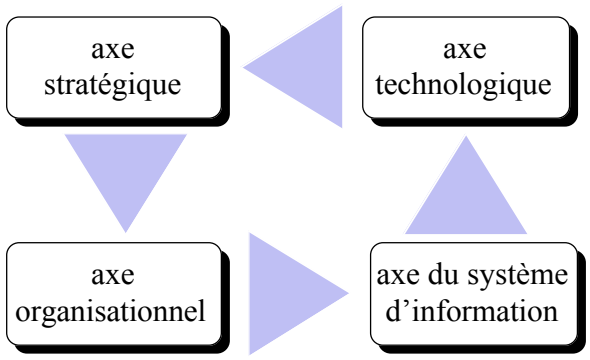


Figure 19.8 : Les quatre axes de l'alignement stratégique

Ces quatre axes s'influencent mutuellement et il faut les examiner un à un :

- *l'axe stratégique* procède à un repositionnement permanent en tenant compte des opportunités technologiques et des contraintes organisationnelles.
- *l'axe organisationnel* procède à une reconfiguration régulière des processus en tenant compte des objectifs stratégiques et des restrictions du système d'information.
- *l'axe du système d'information* développe une structure de communication,

de coopération et de partage de l'information répondant aux besoins des structures organisationnelles et en accord avec les possibilités de la technologie.

- *l'axe technologique* surveille en permanence l'évolution des solutions technologiques en explorant plus particulièrement les solutions adaptées aux besoins du système d'information de l'entreprise, tout en identifiant des voies nouvelles qui pourraient avoir un impact stratégique pour l'entreprise.

En pratique l'équipe de re-engineering s'organise en quatre groupes de travail qui se réunissent cycliquement.

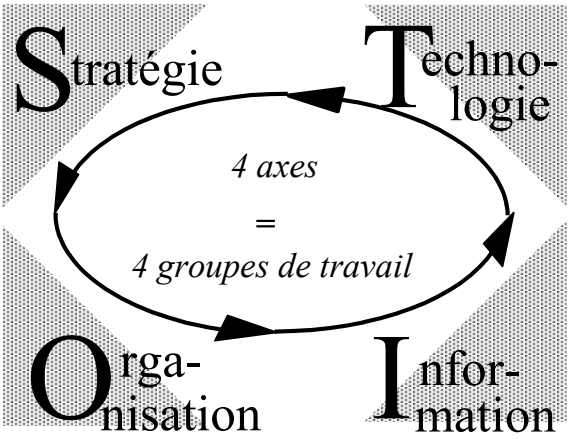


Figure 26.9 : Les quatre groupes de travail de re-engineering

L'équipe "Stratégie" est composée d'experts de la stratégie de l'entreprise, mais comprend aussi au moins un expert de l'axe technologique et un expert de l'axe, mais comprend aussi au moins un expert de l'axe technologique et un expert de l'axe organisationnel. D'une façon générale, chaque groupe de travail comprend essentiellement des responsables de l'axe correspondant, mais comprend aussi au moins un représentant de l'axe précédent et un représentant de l'axe suivant. De cette façon on garantit une continuité dans les raisonnements et une meilleure communication entre les groupes de travail. Chaque groupe essaie de progresser le long de son axe en tenant compte des besoins formulés par le groupe de travail précédent et en exprimant des besoins au groupe de travail suivant.

La démarche n'a pas de début, ni de fin, mais constitue un processus cyclique d'adaptation progressive de l'entreprise aux changements de l'environnement. Le cycle de l'alignement stratégique regroupe à la fois le cycle de décision et le cycle de vie de Merise. La périodicité du cycle peut être adaptée aux besoins de l'entreprise et aux changements qui s'opèrent dans l'environnement, mais d'une façon générale le cycle se déroule assez rapidement : plusieurs passages peuvent raisonnablement avoir lieu chaque année. Il ne faut pas qu'il y ait

rupture du processus. Ceci implique que les changements du système d'information doivent s'opérer au même rythme que la fréquence du cycle d'alignement stratégique et il faut recourir à des outils puissants et souples pour réussir ce tour de force.

Les technologies de l'information au service du BPR

L'alignement stratégique est conditionné en grande partie par les opportunités technologiques du moment et par leur évolution future. Les technologies actuelles rendent possible l'entreprise en réseau et sa restructuration rapide [Tapscott & Caston 94].

Il est indispensable de pouvoir interconnecter des systèmes d'information existants et autonomes, ce qui engendre un certain nombre de conséquences :

- les systèmes d'information doivent être ouverts, c'est-à-dire doivent être capables de communiquer avec d'autres systèmes indépendamment de l'infrastructure matérielle utilisée de part et d'autre.
- les solutions utilisées doivent se conformer à des standards, aussi bien au niveau de la technologie qu'au niveau des messages échangés (interfaces).
- les solutions doivent être conviviales et s'adapter au maximum aux besoins et aux modes de travail de l'utilisateur humain.

En appliquant ces principes à tous les niveaux (inter-entreprise, à l'intérieur de l'entreprise, au niveau du poste de travail), on est en mesure d'agréger, de séparer et de recombiner des systèmes existants, afin de répondre aux besoins de reconfiguration permanents résultant du BPR.

Ces dernières années ont vu apparaître plusieurs approches particulièrement importantes :

- Internet et le commerce électronique ;
- XML et l'échange standardisé de données ;
- l'informatique coopérative ;
- la gestion des flux de travaux.

Internet et le commerce électronique

L'accès des entreprises à Internet constitue un changement technologique qui remet en cause leur façon même de fonctionner. En permettant des échanges de données structurées et non structurées, ainsi que de composants logiciels portables, Internet constitue la colle qui permet de combiner des systèmes d'information indépendants. Ceci mène irrémédiablement à des relations différentes avec les consommateurs, se répercutant techniquement par le

commerce électronique BtoC (*business to consumer*), que par la mise en place de structures coopératives inter-entreprises, se répercutant techniquement par le commerce électronique BtoB (*business to business*). Les marchés se globalisent et des agrégats de plus en plus grands d'entreprises font leur apparition. Les percées dans le domaine de la sécurité juridique, notamment par une reconnaissance quasi-universelle de la signature électronique, vont mener à un développement encore plus rapide de ces restructurations.

Le commerce électronique constitue une nouvelle vague de changements qui déferlent sur les entreprises et va engendrer les besoins d'un BPR de deuxième génération. Désormais ce n'est plus une seule entreprise qu'il faudra repositionner et restructurer, mais des groupes sectoriels entiers et des agrégats d'entreprises partenaires opérant en symbiose.

Pour l'instant il n'existe guère de méthodes de modélisation adaptées à cette évolution, mais la méthode Merise repose sur des bases générales ayant le potentiel d'engendrer des outils valables. En considérant chaque entreprise comme un système poursuivant des finalités, on pourra décrire tout agrégat d'entreprises comme un nouveau système, poursuivant lui-aussi des finalités. L'évolution de l'environnement socio-économique ainsi qu'un processus continu de négociation entre les partenaires conditionnent les finalités de chaque système individuel.

Il sera donc essentiel d'enrichir Merise par des modélisations de finalités et par des modèles d'évolution issus de l'approche systémique, en mettant d'avantage en évidence ce qui représente l'originalité de cette méthode par rapport à des approches purement orientées composants logiciels. Pour jouer un rôle dans la modélisation de systèmes d'information implémentant le commerce électronique Merise devra évoluer vers un outil BPR de deuxième génération.

XML et l'échange standardisé de données

L'évolution d'Internet a progressivement affiné les langages d'échanges de messages structurés en engendrant des standards à tous les niveaux. Si la première génération de langages de marquage de documents se contentait de moyens d'expression assez limités et rudimentaires, ce n'est plus le cas avec la deuxième génération de ces langages. XML (*Extensible Mark-up Language*) constitue l'outil capable de définir une variété illimitée de langages spécialisés pouvant être combinés les uns aux autres.

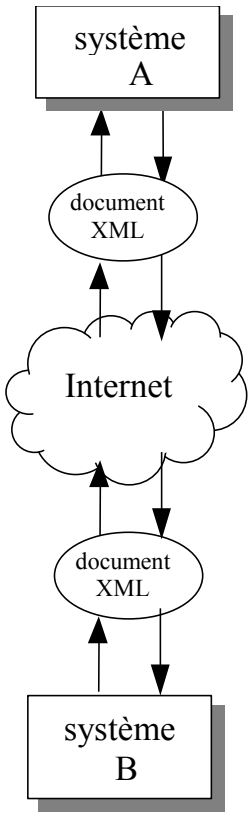


Figure 19.10 : Une communication universelle par XML

Désormais tous les formats d'échange pourront être exprimés sous forme de documents XML. L'intérêt, c'est que le document transporte la description de sa propre structure, ou du moins fait référence à une structure commune connue par les partenaires de l'échange. Ainsi par exemple, le résultat d'une requête à une base de données pourra être exprimé selon un format XML et migrer entre systèmes d'information sous forme d'un document XML, indépendant de la technologie, mais pouvant être interprété par le système destinataire.

Ceci permet une compartimentation plus claire des fonctionnalités logicielles et le développement de composants standards, qui pourront être combinés de multiples façons en fonction des besoins. On pourra par exemple imaginer une application de calcul de salaires qui combinera un composant de retenue d'impôt à la source, fourni par le fisc, un composant de calcul des cotisations sociales, fourni par la sécurité sociale, un composant tarifaire, fourni par une organisation syndicale sur base d'une convention collective sectorielle. Lorsque les règles changeront, les modules pourront être remplacés, sans toucher à l'interface. L'application sera auto-adaptative, sans que l'entreprise ait besoin de développer des composants externes à ses finalités propres.

XML permet de définir des langages spécialisés dans un domaine déterminé et

de combiner les vocabulaires de plusieurs langages spécialisés dans un même document d'échange. Ainsi il y a des langages de description de livres, de formules mathématiques, de pièces de rechange, de signatures électroniques et tous les jours on voit apparaître de nouveaux langages sectoriels. Les anciennes normes EDIFACT sont transformées progressivement en structures XML.

Ceci aura des conséquences importantes pour la modélisation des données. A l'avenir le modélisateur ne pourra plus inventer librement ses propres structures de données, mais devra de plus en plus tenir compte de structures prédéfinies, issues d'efforts de standardisation sectoriels. Un modèle des données sera construit à partir de composants, implémentant des vocabulaires spécialisés (*XML name space*).

Cette évolution est fort intéressante et tout à fait dans l'esprit de Merise : en effet on s'éloigne d'avantage de structures de données plus ou moins limitées par des considérations techniques, pour passer à un niveau d'abstraction plus général, celui du langage. Or le langage est une caractéristique fondamentale des systèmes intelligents, tels que les êtres humains ou les organisations. Merise devrait mettre d'avantage le poids sur la modélisation du langage plutôt que sur la représentation de données, un MCD n'étant rien d'autre qu'un énoncé faisant intervenir des concepts de langages de communication importants pour le système étudié. Ceci irait un peu dans la voie tracée il y a longtemps par l'informaticien-anthropologue C. Vogel dans ses travaux sur le génie cognitif [Vogel 88] et il faudra se poser la question si la méthode Merise à l'avenir ne devrait pas s'inspirer de l'ethno-science, tout comme à ses débuts elle s'est inspirée de la science des systèmes.

L'informatique coopérative ("groupware")

Du point de vue Merise l'échange informatisé de données peut être modélisé au niveau organisationnel, à condition d'intégrer les procédures des deux partenaires, ce qui est est recommandé par le BPR [Bergman 91]. Ceci pose néanmoins le problème de la coopération sur d'autres plans : négociation de conventions d'échange, élaboration en commun des nouvelles procédures, co-évolution des stratégies commerciales. Il s'agit cette fois d'échanger des messages peu structurés et d'entretenir une relation continue, et non pas d'exécuter une transaction commerciale plus ou moins standardisée. On voit apparaître la nécessité de technologies améliorant ce genre de coopération.

Les technologies disponibles sont basées sur l'échange libre de messages non structurés (courrier électronique, vidéoconférences, multimédia, téléprésence) et sur le partage de l'espace de travail (co-rédaction de documents, partage de bases de documents, visualisation commune de fenêtres et intervention commune dans ces fenêtres).

Les outils comprennent des canaux de communication à large bande passante adaptés au multimédia (RNIS, ATM) et des environnements de partage de

documents (p.ex. Lotus Notes). Au niveau logique, ces techniques font intervenir des échanges entre serveurs (réplication de bases de données).

Au niveau organisationnel il s'agit de modéliser des tâches coopératives faisant intervenir simultanément plusieurs postes de travail. Le MOT devrait permettre d'exprimer que certaines tâches exécutées par des acteurs différents ne constituent qu'une seule tâche coopérative, où chacun des acteurs joue le cas échéant un rôle particulier.

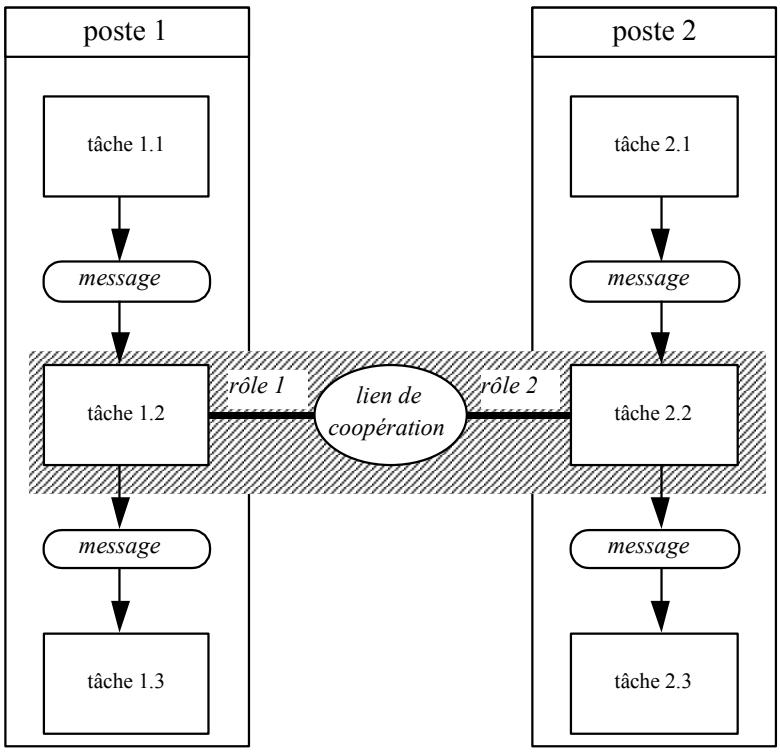


Figure 19.11 : Modélisation d'une tâche coopérative

Pour cela il faudrait introduire la notion de relation de coopération entre tâches, notion qui n'est pas prévue actuellement dans le méta-modèle de Merise. De cette façon deux tâches peuvent être reliées, soit par des liens dirigés de séquencement de tâches (messages entre tâches consécutives), soit par des liens non dirigés de coopération (tâches synchrones avec partage de données).

La gestion des flux de travaux ("workflow")

Il existe une forme plus structurée de coopération qui se situe entre l'échange de données informatisé et l'informatique coopérative. Il s'agit d'une implémentation des procédures internes basée sur un échange automatique de messages électroniques.

La généralisation des réseaux informatiques à l'intérieur de l'entreprise a

rapidement engendré l'apparition de serveurs spécialisés dans le routage des flux de travaux à travers une procédure. Nous avons vu que l'automatisation pure et simple des procédures existantes pouvait se révéler en opposition avec les finalités du BPR, lorsque cette automatisation ne s'accompagne pas d'une remise en cause et d'une restructuration fondamentale de la procédure, avec une redéfinition du rôle joué par les différents intervenants de la procédure. Par contre, lorsque l'automatisation des flux de travaux s'insère dans le cadre d'un BPR bien mené, elle peut apporter des gains de productivité remarquables.

L'intérêt des outils de gestion de flux de travaux, c'est qu'ils obligent à modéliser les procédures, et la plupart de ces outils permettent à présent de faire des modélisations graphiques d'une façon très aisée; ils permettent en outre de quantifier les procédures et de les simuler. Ceci favorise la prise de conscience des défauts de la procédure actuelle par les personnes concernées et amène progressivement une amélioration de la procédure. Par ailleurs il n'est plus nécessaire de programmer le routage, le modèle pouvant être transformé automatiquement en script de routage. L'idéal serait que l'AGL utilisé pour modéliser les procédures puisse être interfacé directement avec l'outil de routage. Ceci présuppose une normalisation de la description des procédures, à l'instar de ce qui est proposé par la "Management-Workflow Coalition".

Les modèles de Merise sont tout à fait suffisants pour modéliser les flux de travaux. En général un diagramme de flux entre acteurs est suffisant ; ce diagramme se limite à représenter les différentes phases, sans préciser les tâches qui composent la phase. Pour des flux plus complexes un MOT complet peut apporter des compléments utiles. Au cas où la transposition de la procédure ne peut pas être générée automatiquement, il s'avère très utile d'utiliser un diagramme représentant le cycle de vie de l'objet qui parcourt la procédure. Par exemple dans une application de rédaction d'un document celui-ci parcourt un certain nombre d'états, qui déterminent le routage du document : il peut être "en cours de rédaction", "en cours de vérification", "en cours d'approbation", "approuvé", "en cours de diffusion", "annulé".

La combinaison des possibilités du routage de flux de travaux et des possibilités plus ouvertes de l'informatique coopérative, rend possible des solutions très intéressantes, car elle permet d'insérer dans une procédure des tâches coopératives réalisées en commun par plusieurs intervenants (comme par exemple une prise de décision par un comité). Ces possibilités sont particulièrement utiles dans le contexte de l'alignement stratégique, où une coopération structurée de beaucoup d'intervenants est indispensable pour adapter en permanence les stratégies, les structures organisationnelles, les moyens technologiques de l'entreprise.

Merise et le BPR : quelques conclusions

Nous avons vu que le BPR est une démarche qui est elle-même en train d'évoluer en fonction du développement des technologies et de l'environnement socio-économique. Après un BPR de première génération, visant à repositionner et à reconfigurer une entreprise déterminée, nous passons à une deuxième génération de BPR, visant à restructurer des agrégats d'entreprises, voire même des secteurs économiques entiers.

Merise et le BPR se ressemblent beaucoup par leurs préoccupations et par leurs démarches. Loin d'être une approche dépassée par le progrès technique, Merise est tout à fait capable d'aborder les problèmes de modélisation de l'entreprise en réseau.

Tout comme le BPR doit évoluer, il en est de même de Merise. Au lieu d'éviter les niveaux d'abstraction les plus élevés, comme le font trop souvent les informaticiens et les méthodes qu'ils appliquent, Merise devrait au contraire s'enrichir pour tenir compte des besoins du niveau stratégique, tout en conservant l'originalité de ses racines systémiques.

Cela entraîne le développement de méthodes permettant de modéliser des hiérarchies de finalités. Il faudra aussi envisager d'étendre la modélisation des données par une modélisation plus générale de langages de métiers. Il faudrait trouver des moyens pour modéliser le comportement d'acteurs économiques, que ce soit l'utilisateur, le client, une organisation partenaire ou simplement un agent logiciel parcourant Internet. La coopération entre acteurs devrait pouvoir être exprimée, tout comme les mécanismes d'évolution des systèmes d'information.

Voilà un vaste programme de travail pour la grande communauté des "Merisiens". Tous ceux qui se sentent concernés par ces développements devraient coopérer pour développer la vision d'un Merise pour le 21e siècle, Merise 21 !

20

Conclusion

Durant près de vingt ans la méthode Merise s'est confrontée aux réalités d'une mise en oeuvre dans une grande variété d'organisations. Elle est largement diffusée en France, souvent pratiquée en Europe du Sud (parfois avec des adaptations mineures) et dans certains pays d'Europe du Nord comme la Belgique, la Suisse et même l'Allemagne par laquelle un projet du programme européen Force a financé sa diffusion dans des landers du nord. Elle a aussi inspiré des méthodes nord américaines, comme par exemple la méthode P+ proposée par la société canadienne DMR.

On peut ainsi dire que Merise constitue un standard de fait en conception de système d'information.

Dans cette conclusion nous allons essayer, dans un premier temps, de faire un bilan de ces vingt ans de mise en œuvre de cette méthode. Dans un deuxième temps, nous positionnerons Merise par rapport aux méthodes et notations orientées objet (OMT et UML). Enfin, nous tenterons de projeter l'évolution possible de Merise que nous percevons vers une ingénierie à base de composants qui nous apparaît adaptée à la conception de systèmes d'information de ce début du vingt et unième siècle.

Bilan sur la méthode Merise

Faire un tel bilan n'est pas chose aisée. Rappelons tout d'abord que l'ingénierie des systèmes d'information a pour objectif de formaliser les besoins et attentes de l'organisation et d'en générer les spécifications de futures applications informatiques. Ainsi, l'ingénierie des systèmes d'information et Merise en particulier se situent à la rencontre de deux domaines qui sont l'organisation - le métier - et l'informatique. Dès son origine Merise s'était fixé l'objectif de réconcilier ces deux domaines en permettant l'établissement d'un dialogue permettant aux concepteurs de comprendre le métier de l'organisation afin de pouvoir formaliser ses besoins et attentes.

Un bilan de vingt ans de mise en œuvre de Merise est lié à l'évolution de l'ingénierie des systèmes d'information sur cette période, évolution à son tour liée aux évolutions respectives de ces deux domaines sur cette période. D'une

façon générale, au cours de cette période il est arrivé que l'évolution du métier entraîne l'informatique, parfois fut le contraire, l'informatique qui, de par ses nouvelles potentialités, fit évoluer le métier. Bien sûr ces évolutions de l'ingénierie des systèmes d'information et en conséquence des méthodes de conception se font dans un contexte économique lui-même en évolution qui impose alors le développement de stratégies.

Les années 80 : Merise première génération et la prise en compte du métier

Sur le plan de l'organisation et donc des métiers, les années 80 se caractérisent par une prise de conscience d'une nécessaire reconception de l'architecture générale de l'informatique au sein des entreprises. Cette reconception doit assurer la cohérence générale des informations, préserver l'évolutivité des modes de gestion et d'organisation, permettre l'introduction de nouvelles technologies sans compromettre l'acquis et enfin associer, dans leurs responsabilités respectives, décideurs, utilisateurs et informaticiens.

C'est dans ce contexte qu'ont émergé la notion de système d'information et la nécessité de méthode complète de conception et de spécification permettant l'informatisation des systèmes d'information. L'approche systèmique a largement contribué à ce renouveau et la méthode Merise y puise ses racines.

Il s'agit alors d'appréhender le métier de l'organisation afin d'une part de cerner la mémoire centralisée de l'organisation et d'autre part de permettre au travers de cette dernière de coordonner les activités de l'organisation. Une modélisation des données et des activités permet d'y arriver en ayant cependant recours à une modélisation à plusieurs niveaux d'abstraction cristallisant des préoccupations managériales et organisationnelles associées à la compréhension du métier et des préoccupations techniques (informatiques).

Sur le plan informatique, les bases de données commencent a être opérationnelles et permettent de supporter cette mémoire centralisée nécessaire à cette coordination de l'organisation. La modélisation des données, proposant des formalismes et des outils pour décrire les données indépendamment de leurs utilisations dans des traitements favorise ainsi la mise en œuvre des bases de données. Les SGBD relationnels s'imposent à la fin de cette décennie. Des environnements de développement d'applications autour de ces bases de données font leur apparition. Les réseaux locaux apparaissent et facilitent la coordination de l'organisation.

Merise, de par son cadre de modélisation et sa démarche spécifiques adaptés à la prise en compte du métier apparaît alors comme la réponse adaptée à la conception de systèmes d'information organisationnels.

Les années 90 : Merise deuxième génération et l'évolution des métiers

Les années 90 sont caractérisées par un accroissement de la concurrence, une recherche de productivité qui conduira sur le plan de l'organisation et du métier

à une remise en cause des modèles et pratiques organisationnels qui est l'objet du BPR. Le cadre de modélisation proposé par Merise, de par ses modèles d'activité positionnés à des niveaux d'abstraction (conceptuel - managérial, organisationnel) permet une modélisation et une analyse pertinente de ces modèles et pratiques organisationnels, comme le développe le chapitre 19 de l'ouvrage.

Ces différentes évolutions du métier conduisent à reconsidérer le caractère centralisé du système d'information. L'organisation se structure en centres de profit plus ou moins autonomes, disposant de leurs propres informations voir de leurs propres systèmes d'information. Il s'agit alors notamment de gérer les modalités de partages des informations de ces systèmes. De plus les systèmes d'information sont de plus en plus complexes. Les besoins des organisations en information pour l'aide à la décision s'élargissent notamment à des entrepôts de données (Data-Warehouses). historisées sur plusieurs années.

Sur le plan technique, le mariage des bases de données et des réseaux est suffisamment solide pour pouvoir supporter de telles évolutions organisationnelles. C'est l'apparition des architectures client-serveur, des bases de données distribuées et réparties. Les performances des SGBD, notamment obtenues par le parallélisme, permettent maintenant de gérer efficacement des bases de données très importantes notamment celles associées à des entrepôts de données

Merise évolue pour répondre aux spécificités de ces nouveaux systèmes d'information dont les organisations ont besoin. Pour cela il est nécessaire d'en réviser le cadre de modélisation (passage de 3 à 4 niveaux d'abstraction) et d'étendre les formalismes de données et de traitements. Ces extensions s'inspirent principalement des méthodes et notations orientées objet proposées dans le domaine du génie logiciel, en conception logicielle. Ces évolutions conduisent à la deuxième génération de Merise. Merise permet alors d'appréhender de façon satisfaisante ces nouveaux besoins et de mettre en œuvre ces nouvelles technologies. On peut rajouter que cette nouvelle génération la renforce encore dans sa spécificité en distinguant bien le SIO associé au métier de l'organisation, du SII associé à la mise en oeuvre de l'outil informatique.

La fin des années 90 voit la confirmation des méthodes de conception logicielle orientées objet comme OMT et l'apparition de la notation objet unifiée UML. Un positionnement de Merise par rapport à ces méthodes ou notation est nécessaire, ce que nous allons tenter de faire.

Merise et les méthodes et notation orientées objet : positionnement

Pour effectuer un tel positionnement, il nous est nécessaire de rappeler tout l'intérêt de l'approche objet dans le génie logiciel, mais aussi pour l'ingénierie des systèmes d'information.

L'approche objet en génie logiciel

Largement pratiquée dans de nombreux secteurs de l'informatique temps réel et de l'ingénierie logicielle, l'approche objet offre une manière claire de concevoir une architecture modulaire, encapsulant la complexité et facilitant l'implémentation multi-plateformes. Elle permet une flexibilité technique accrue et une meilleure ouverture aux nouvelles technologies telles que le multimédia. Elle touche aujourd'hui les systèmes d'exploitation, le middleware des architectures client-serveur et de plus en plus l'informatique de gestion. Grâce aux efforts de standardisation et de normalisation conduits notamment sous l'égide de l'Object Management Group (OMG), on peut s'attendre à ce que les progrès de l'approche Objet s'accélèrent.

A la fin des années 90, l'approche objet est réellement opérationnelle dans le génie logiciel, tant au niveau de la conception logicielle (conception orientée objet), qu'au niveau de la réalisation (programmation orientée objet). L'approche objet s'est particulièrement distinguée dans le domaine des interfaces homme-machine. Dans la conception et la réalisation de ces interfaces, basées sur le paradigme objet-action dans lequel l'utilisateur désigne un objet puis choisit le service qu'il veut en obtenir, l'objet encapsulant ses services, est parfaitement adapté. L'approche objet a ainsi permis d'offrir à l'utilisateur une interface ergonomique et intuitive conforme à son contexte d'activité.

Toutefois nous n'en sommes cependant pas encore au tout objet. En effet, bien que très séduisants, les systèmes de bases de données orientés objets sont aujourd'hui encore au stade préindustriel. Ils ne peuvent actuellement apporter une réponse satisfaisante à la gestion d'un volume important de données structurées et partagées sur lesquelles reposent les systèmes d'information des entreprises actuelles. Dans cette évolution des bases de données vers l'objet, les praticiens qui ont ces dernières années majoritairement opté pour des SGBD relationnels, mettent actuellement plus d'espoir dans une évolution en cours des SGBD Relationnels vers l'objet que dans des SGBD révolutionnairement objet.

Ingénierie des systèmes d'information et approche objet

L'approche objet dans les méthodes de conception de système d'information nous semble avoir deux intérêts essentiels : la réutilisation d'objets tant au

niveau du SII que du SIO et la continuité méthodologique (démarche et formalismes) centrée sur la notion d'objet entre la conception du SIO et celle du SII. Une telle évolution serait une réponse à l'actuelle préoccupation majeure des responsables systèmes d'information à savoir la migration vers l'objet, le choix des applications et des «charpentes» d'objets à développer, ceci en adéquation avec les besoins de l'organisation et le Business Process Re-engineering (BPR).

Actuellement deux voies d'évolution possibles de Merise et des méthodes de conception de systèmes d'information traditionnelles vers l'objet nous paraissent possibles :

- La première voie, que nous nommerons "transition vers l'objet", se veut pragmatique, opérationnelle à court terme et préserve une grande partie des investissements méthodologiques antérieurs. Elle consiste à définir de façon claire comment passer de la spécification d'un SIO réalisé selon le cadre méthodologique merisien (celui-ci pouvant être éventuellement aménagé) à une spécification du SII orientée objet. Dans cette voie, il n'y a certes pas de continuité méthodologique objet, mais plutôt une transition méthodologique vers une architecture logicielle objet.
- La seconde voie, que l'on peut appeler "évolution vers l'objet" est plus radicale et conduit à une remise en cause profonde de Merise, permettant de traiter complètement la réutilisation et la continuité méthodologique entre le SIO et le SII. La réutilisation comme la continuité méthodologique reposerait dans cette voie sur l'introduction du concept d'objet au niveau SIO que l'on pourrait nommer "objet métier" et sa dérivation vers le concept d'objet du génie logiciel associé au SII, ou "objet logiciel ou technique".

La première voie, bien que modeste, nous apparaît la plus réaliste aujourd'hui. En effet, dans le contexte actuel et pour les raisons que nous avons précédemment évoquées, une remise en cause radicale nous apparaît difficilement envisageable pour la plupart des entreprises. Une évolution méthodologique vers l'approche objet qui s'effectuerait par transition, en s'appuyant sur des bases méthodologiques acquises telles que Merise, rencontrerait certainement l'adhésion des praticiens.

Un tel couplage est d'autant plus facilité pour la méthode OMT (et UML) qui a retenu pour l'expression des diagrammes de classes (modèle objet) un formalisme graphique constituant une extension du formalisme Entité-Relation proposé par Merise. Pour plus de détail sur ce couplage de la méthode Merise avec la méthode OMT, nous renvoyons le lecteur à un article détaillant les règles et conseils de passage des modèles organisationnels Merise (MOT, MOD) aux trois modèles de OMT (Modèles Objet, Dynamique et Fonctionnel) [Espinasse & Nanci 97].

Dans la seconde voie, bien des aspects sont actuellement imprécis et relèvent

encore de la recherche. Cependant, comme l'a développé le chapitre 9 de cette quatrième édition, "Cycle de vie des objets et des objets métier", l'identification, la formalisation et bientôt la réutilisation d'objets métier au niveau du SIO se font jour, premiers pas vers une ingénierie à base de composants.

Positionnement de Merise et de UML

Dans la foulée de la méthode OMT, la notation UML semble s'imposer. Sans conteste cette notation constitue une avancée significative pour les méthodes de conception logicielle en leur proposant un standard de notation qui leur manquait. Ce standard de notation se présente comme une boîte à outils de modèles et de diagrammes permettant la spécification d'une conception objet.

Notons que cette notation ambitionne de contribuer à la modélisation objet de tout système complexe logiciel ou non. En ce qui concerne le domaine du logiciel, la richesse de cette boite à outils permet de cerner les aspects structurel, dynamique et fonctionnel d'un logiciel, tant à un niveau analyse (analyse des besoins, spécification fonctionnelle), qu'à un niveau conception (architecture logicielle, conception générale et détaillée), qu'enfin à un niveau implémentation. Nous ne rentrerons pas dans une discussion plutôt stérile sur la puissance de tel ou tel formalisme pour cerner tel ou tel aspect.

Cette richesse de notation conduit certain à tenter de l'utiliser pour la conception de systèmes d'information, c'est-à-dire pour modéliser le SIO. Le risque nous apparaît grand car il peut conduire à remettre en cause l'approche systémique originale de Merise propice à appréhender le métier de l'organisation. Ainsi dans la recherche d'une continuité de modélisation centrée sur le concept d'objet, le cadre de modélisation associé à UML se voulant universel, il n'est pas adapté à la conception de systèmes d'information organisationnels.

Ainsi la nécessité de niveaux d'abstraction que préconise Merise et qui regroupent des préoccupations homogènes, spécifiques à la problématique de ces systèmes, notamment relatives au métier, à savoir préoccupations managériales et préoccupations organisationnelles, est remise en cause. Peut-on raisonnablement faire l'économie de ces niveaux d'abstraction spécifiques lorsqu'ils apparaissent de plus en plus pertinents et d'actualité comme nous l'avons vu dans le chapitre 9 pour accompagner le BPR?

Certes non. Leur disparition conduirait au divorce de l'informatique, de l'organisation et du management, ce que nous avions avant Merise. Nous ne concevons pas des systèmes d'information pour mettre en œuvre des techniques informatiques, concevoir et réaliser des logiciels, mais pour supporter des organisations en constante évolution, tant dans leurs pratiques organisationnelles que managériales associées à l'exercice de leur métier, afin de les aider dans leur adaptation à un environnement concurrentiel.

Dans cette adaptation, au delà d'un problème de notation, il nous apparaît bien

plus important de pouvoir, de façon rationnelle, accroître la qualité des logiciels impliqués dans ces systèmes d'information ainsi que la performance des processus utilisés pour les concevoir et les réaliser. Cette qualité et cette performance ne peuvent être améliorées que par une approche par réutilisation de composants, comme cela a déjà été le cas dans bien d'autres domaines d'ingénierie.

Ainsi, le problème central pour l'ingénierie des systèmes d'information est bien celui de la réutilisation et non celui d'une continuité de formalisme - de notation -, permettant de modéliser à la fois le SIO et le SII. Si l'approche objet actuellement associée au génie logiciel apparaît adaptée à une réutilisation en conception et réalisation logicielle, elle ne l'est pas encore pour la conception de systèmes d'information organisationnels en ne prenant pas en compte toute la spécificité de leur complexité, notamment celle associée au métier.

En conséquence, l'adoption d'une notation adaptée au génie logiciel, comme UML, pour la conception de systèmes d'information organisationnels, ne resouds en rien notre problème. Par contre une ingénierie des systèmes d'information à base de composants nous semble la voie à suivre.

Evolution future de Merise : l'ingénierie à base de composants

Le contexte des années 2000

Ce début de siècle est caractérisé par la globalisation de l'économie. La performance des entreprises des pays industrialisés est liée à la puissance de leur organisation interne et par là même à celle des systèmes d'information qui la supporte. Le contexte concurrentiel de plus en plus vif conduit les entreprises à une recherche constante de productivité à tous les niveaux, mais aussi à un accroissement de flexibilité, d'adaptabilité et de réactivité. Les entreprises sont pour cela conduites à une permanente remise en cause de leurs modes de gestion et d'organisation (BPR).

Les nouveaux modes de gestion et d'organisation privilégiés par les entreprises sont principalement caractérisés par des processus décisionnels de complexité croissante, due notamment au raccourcissement constant du cycle de vie des produits. Un tel raccourcissement nécessite la coordination de processus décisionnels distribués, parfois le temps d'un projet, entre de nombreuses équipes de conception, de fabrication, de logistique, de marketing, de finances... internes à l'entreprise ou partenaires de celle-ci. De nouveaux types d'organisation apparaissent; citons notamment l'entreprise réseau, l'entreprise étendue ou encore l'entreprise virtuelle.

Les systèmes d'information supportant ces nouveaux types d'organisation, en permanente évolution, sont nécessairement distribués pour prendre en compte

dans l'entreprise tout fait nouveau jusqu'à complète répercussion de l'ensemble de ses effets. Cette prise en compte nécessite l'usage d'environnements informatisés eux-mêmes distribués et coopératifs dans lesquels les réseaux (d'Intranet à Internet) tiennent une place centrale au même titre que les données mémorisées, augmentant d'autant la complexité de ces systèmes d'information. Dans la compréhension de la complexité de ces systèmes d'informations à concevoir, "formées par l'interaction entre leurs propres parties" [Morin 1977], le problème essentiel est celui de l'autonomie de leurs parties et de leur coopération. Cette coopération est essentielle dans les architectures distribuées. Comment faire coopérer des sous-systèmes, des unités, des composants conçus et maintenus par des équipes indépendantes ?

Le marché des composants

Comme nous l'avons déjà évoqué, l'accroissement de la qualité des systèmes d'information ainsi que la performance des processus utilisés pour les concevoir et les réaliser passe par la réutilisation de composants. Par composant, nous entendons bien sûr les composants logiciels mais aussi les composants de spécification de conception. Il est clair que ces grands types de composants sont de plus en plus en étroite relation.

D'une façon générale, les composants devraient jouer un rôle de premier plan dans l'industrie du logiciel. On en attend les mêmes bénéfices que ceux promis par l'objet : réutilisabilité, interopérabilité, productivité, qualité, maintenabilité... Mais le marché du développement à base de composants en est qu'à ces débuts et la pratique actuelle de l'ingénierie à base de composants est difficile à estimer. Une enquête citée par [Sodhi & al.98] sur un ensemble d'organisations ayant mis en place des programmes de réutilisation fait apparaître pour ces dernières des bénéfices importants (50% de réduction en temps d'intégration, 20 à 30% d'amélioration de la qualité).

De même, le Gartner Group prévoit que le développement à base de composants devrait s'imposer dans quelques années comme la forme d'ingénierie majeure : en 2003, 50% des développements devraient se faire à partir de composants et que les revenus générés par les ventes de logiciels à base de composants et par les services associés devraient augmenter de manière très significative [kiely 98]. Cette enquête montre aussi que la réutilisation est pratiquée avec des méthodes et des outils empiriques et qu'il existe aujourd'hui un réel besoin de rendre la réutilisation plus systématique dans l'ingénierie des systèmes d'information.

Le cabinet IDC distingue pour sa part un segment particulièrement dynamique pour le développement des composants, celui des outils de développement d'applications incluant des produits tels que les L3G, les L4G, les SGBD, les composants logiciels et les outils de construction et d'assemblage de composants, suit ensuite le segment des "middlewares".

L'ingénierie à base de composants

On peut distinguer différents types de composants se différenciant par leur granularité, par leur niveau d'abstraction (composants logiciels, composants de conception, composants de métier) ou encore par la nature de la connaissance qu'ils permettent de réutiliser. Les composants peuvent ainsi prendre la forme de classes d'objets individuelles ou de structures de connaissance très complexes. Certains composants fournissent des fragments de produit (logiciel, architecture, schéma conceptuel, …), d'autres décrivent des fragments de démarche [Cauvet & al. 00].

Dans cette ingénierie à base de composants, on distingue deux problématiques complémentaires : celle associée au développement d'un système de réutilisation ("design for reuse") et celle relative au développement d'un système particulier par réutilisation ("design by reuse") [kang & al. 90]:

- Le développement d'un système de réutilisation met en œuvre des tâches d'identification de ressources réutilisables, de spécification et d'organisation de composants, mais aussi d'implantation d'un ensemble de composants au sein d'un système de réutilisation. Les recherches concernent la définition de modèles et de langages pour la spécification de composants réutilisables permettant la représentation de connaissances génériques. Des méthodes et des outils logiciels assistant l'identification de ressources réutilisables et la conception / réalisation de systèmes de réutilisation doivent également être développés.

- Le développement d'un système, par exemple un SI, par réutilisation de composants nécessite une re-formulation des processus usuels d'ingénierie utilisés pour le développement de ce système ainsi que des outils de développement intégrant de manière systématique la réutilisation de composants. Un tel développement consiste à identifier et organiser les composants en fonction d'une stratégie de réutilisation, de contraintes de déploiement et de composants disponibles pouvant avoir différentes granularités comme un composant métier élémentaire ou une application complète.

Divers modèles (et langages) de composants sont dès aujourd'hui proposés, citons notamment les abstractions de domaine [Maiden 93], les patrons ("design pattern") [Gamma &al. 95] et les canevas ("frameworks") [Johnson 93]. D'une façon générale et comme le souligne [Cauvet & al. 00], ces modèles tout d'abord s'orientent de plus en plus vers des formes de composants permettant la réutilisation de produits de spécification et de produits de conception, en conséquence vers le métier associé à l'applicatif à développer. Ensuite ils intègrent, dans la spécification des composants, des propriétés permettant de guider leur réutilisation, facilitant par exemple leur recherche, leur adaptation et leur assemblage.

Composants et objets métier

Stimulée par le succès de Java, la technologie objet se répand et jette les bases d'une industrie du logiciel à base de composants et tournée vers le métier des utilisateurs. Les composants métier ou objets métier représentent la forme la plus élaborée de la réutilisation pour laquelle les enjeux techniques et surtout commerciaux sont considérables. Notons que l'OMG (Object Management Group) et l'OAG (Open Application Group) travaillent chacun de leur côté à la standardisation de composants métier.

De même que les "widgets" sont devenus incontournables pour le développement d'interface graphique, les objets métier sont les briques applicatives de base pour l'assemblage d'applications qui touche à un métier ou à un processus de l'organisation. Si les premiers s'adressent aux programmeurs, les objets métier s'adressent aux analystes, aux utilisateurs avancés et aux éditeurs de logiciels ou de progiciels, c'est-à-dire à l'organisation et son métier (l'entreprise) plutôt qu'uniquement aux informaticiens.

L'offre en composants métier en est à ces débuts. Parmi les acteurs pionniers on trouve des sociétés comme Soleri-Cigel, Nat Systems, Synon et Lyon Consultants (IMR). De même citons l'investissement considérable d'IBM sur le projet San Francisco [Carey & al. 00] qui regroupe plus de cinquante éditeurs majeurs autour de la construction de bibliothèques de composants métier réutilisables pour le développement de progiciels Client/Serveur en Java. Ces composants métiers définis en tant que "design patterns" ont conduit au développement de bibliothèques de composants métier réutilisables en Java et organisés en "framework". La première mouture (300 composants Java), sortie en 1997, mettait à disposition l'infrastructure technique nommée "Foundation", les services applicatifs généraux appelés "Common Business Objects" ainsi que la première brique, consacrée à la comptabilité générale de la couche métier (une centaine de composants). Cette brique appelée "Core Business Processes" est destinée à recevoir des processus regroupés en sous-systèmes extensibles ou "frameworks" applicatifs. La deuxième mouture sortie en 1998 rajoute deux "framework" l'un consacré aux commandes et l'autre à la gestion des stocks. La troisième rajoute le traitement des effets à recevoir et des effets à payer. Les composants Java issus du projet commencent à être utilisés (plusieurs centaines de licences vendues) et ont dès à présent permis le développement de nombreuses applications.

Vers une ingénierie des systèmes d'information Merise à base d'objets métiers

Rappelons que (cf. chapitre 9) la notion d'objet métier est progressivement apparue dans la méthode Merise au début de la décennie 90 comme une extension de la modélisation conceptuelle des données. Plus de vingt années de pratique nous conduisent à penser que Merise nous offre un cadre de modélisation propice à une ingénierie par composants et plus particulièrement par composants métier.

En effet dès son origine, Merise s'est appliquée à concilier l'organisation, c'est-à-dire le métier, avec l'informatique. Le cadre de modélisation structuré en niveaux d'abstraction ou de préoccupation et plus particulièrement les niveaux conceptuel et organisationnel tels que définis dans merise deuxième génération, nous apparaissent vraiment adaptés à la modélisation du métier et par la même à l'identification d'invariants métier, de composants ou objets métier. Cette conviction est d'ailleurs confortée par l'usage de Merise pour le BPR dont la finalité est essentiellement de modéliser les pratiques organisationnelles du métier (cf. chapitre 19).

Dans l'ingénierie à base de composants, nous avons vu que l'on pouvait distinguer deux problématiques complémentaires : celle du développement d'un système de réutilisation et celle du développement d'un système particulier par réutilisation. Dans le cadre de cet ouvrage, nous nous plaçons essentiellement dans la première problématique. Le développement d'un système de réutilisation nécessite tout d'abord l'identification des ressources réutilisables - les composants métier -, leur spécification et leur organisation.

Un composant métier de conception en ingénierie des systèmes d'information spécifie un ensemble de services, mais sans être lié à une application particulière, par exemple, un objet métier facture fournissant un service permettant de saisir un encaissement. Un même composant peut être réutilisé pour concevoir différents systèmes particuliers par exemple un composant métier facture gérant des factures peut être réutilisé pour concevoir différentes applications de facturation.

C'est à l'identification, la spécification et l'organisation de composants métiers de conception que nous pensons dés à présent avoir contribué dans le chapitre 9. Certes cette reflexion n'en est qu'à ses débuts. Un composant n'est pas forcément un objet même si les concepts objets s'appliquent assez logiquement aux composants : encapsulation, classification (héritage), polymorphisme.

Nous avons dans le cadre de cette quatrième édition, présenté tout d'abord deux différentes perspectives méthodologiques associées à l'objet métier : la macro-modélisation et la modélisation intégrative. La macro-modélisation se limite à la modélisation des données (structure) de l'objet métier. La modélisation intégrative ajoute à cet aspect structurel l'ensemble des autres aspects de sa modélisation. L'objet métier apparaît alors comme l'intégration de diverses facettes d'un concept du métier qu'explicitent les différentes modélisations formalisées : son comportement dynamique (CVO), son utilisation fonctionnelle (MCT - MOT) et ses présentations (maquettes d'IHM).

La perspective de la macro-modélisation nous a semblé, dans l'état actuel de nos réflexions, la plus praticable et nous avons proposé, dans le cadre de Merise, des éléments méthodologiques permettant tout d'abord la modélisation d'objets métiers (nouvelles extensions du formalisme Entité-relation) et ensuite d'accompagner le concepteur dans l'élaboration de ces modèles en lui

proposant une démarche spécifique (élaboration par composition et par décomposition).

Dès à présent ces nouveaux éléments méthodologiques devraient permettre au concepteur merisien une certaine rationalisation de la conception de modèle de données (MCD) au travers d'une meilleure intégration (interne et externe) de ces objets métier conduisant à une réduction des coûts de développement et d'évolution des systèmes d'information.

La définition complète d'un système de réutilisation ("design for reuse") nécessiterait de passer de l'approche macro-modélisation à l'approche intégratrice prenant en compte les autres composantes de l'objet métier (comportement, fonctionnement, présentation) y compris pour les autres niveaux (organisationnel et logique). Ensuite il faudrait envisager l'implantation d'un tel système mettant à disposition du concepteur un ensemble (bibliothèques) de composants métiers. Une telle implantation pourrait s'appuyer sur l'atelier Merise Win'design.

Un tel système étant développé, l'étape suivante consisterait alors à reformuler les processus d'ingénierie associés au développement d'un système d'information particulier à partir des bibliothèques de composants métiers disponibles ("design by reuse"). Il s'agirait entre autres d'apporter une aide méthodologique au concepteur dans le choix des composants, dans leur adaptation et leur assemblage, ceci selon des stratégies de réutilisation encore à définir.

Ainsi l'approche objets métier peut désormais s'inscrire comme un complément et une évolution prometteuse de l'approche « merisienne » classique en ingénierie de systèmes d'information et comme diraient certain "sur le métier remettez votre objet…"

21

Références bibliographiques

Paul Valéry,
"Introduction à la méthode de Léonard de Vinci",
Editions Gallimard, Paris, 1957.

Références directement liées à la méthode Merise

Alakl N., Lalanne J.-Ch., *la Méthode TACT*, éditions d'Organisation, 1989.

AFCET-CERAM, *Autour et à l'entour de Merise : les méthodes de conception en perspective*, Actes du congrès AFCET-CERAM, 17-18-19 avril 1991, Sophia-Antipolis, publication AFCET, 1991.

AFCET, *Merise et les autres : quels systèmes d'information pour un monde qui change*, Actes du congrès AFCET, 5-7 octobre 94, Versailles, publication AFCET, 1994.

Bergman M., Cucchi A., Espinasse B., Lorenzo F., *Merise et l'EDI : contribution à une méthodologie de conception de systèmes d'information d'échange communautaires*, in *Autour et à l'entour de Merise : les méthodes de conception en perspective*, Actes du congrès AFCET-CERAM, 17-18-19 avril 1991, Sophia-Antipolis, publication AFCET, 1991.

Brams G.W., *Réseaux de Petri : théorie et pratique*, tomes I et II, éditions Masson, 1983.

Bredeweg B., Brunet E., De Greef P., Schreiber G., Wielinga B., Wallyn A., *A KADS Approach to KBS Design*, Esprit Project, Deliverable D6, 1989.

Brunet E., Dorbes G. *KADS et Merise : vers une unification du génie cognitif et du génie logiciel*, in *Génie logiciel & systèmes experts*, n° 19, juin 1990, pages 10 à 27, publication EC2, 1990.

CCE, *Aspects de l'EDI*, EUR11883FR, Office des publications officielles de la Communauté européenne, 1989.

CCE, *Rapport d'activité du programme TEDIS* 1988-1989, COM(90) 361, juillet 1990.

CGI, *Projet PAC : programmation automatique CORIG*, brochure technique, CGI, 1972.

Centre technique informatique, *Méthode de définition d'un système d'information*, mission informatique, ministère de l'Industrie, 1979.

Courbon J.C., *Systèmes d'Information: structuration, modélisation et communication*, InterEditions, 1993.

Espinasse B., Lai M., Nanci D., *Merise+ : Une extension de la méthode MERISE à l'approche objet par un apport de la méthode HOOD,* Revue Ingénierie des Systèmes d'Information, Hermès Editeur, Volume 3 - n°2-3, 1995.

Espinasse B., Nanci N., *Merise et l'approche orientée objet : du couplage avec OMT à une troisième génération*, Revue Ingénierie des Systèmes d'Information, Hermès Editeur, Vol.5, n°4, oct. 97.

Franckson M., Peugeot C., *Eurométhode*, in *Autour et à l'entour de Merise : les méthodes de conception en perspective*, Actes du congrès AFCET-CERAM, 17-18-19 avril 1991, Sophia-Antipolis, publication AFCET,1991.

Heckenroth H., Tardieu H., Espinasse B., *Modèles et outils pour la conception de la cinématique d'un système d'information,* Rapport de recherche IRIA 310, CETE d'Aix-GRASCE/CNRS, université d'Aix-Marseille III, 1980.

Jacques P., Mercier B., Mollard D., Nguyen V.H., *De Merise à l'approche objet*, Rapport d'étude, CNAM-Informatique d'entreprise, 1996.

Longworth G., Nicholls D., *SSADM Manual*, vol. 1 et 2, NCC Publications, 1986.

Levine P., Pomerol J.-C., *Systèmes interactifs d'aide à la décision et systèmes experts,* Hermes, Paris, 1989.

Morejon, J., *Merise : vers une modélisation objet,* éditions d'Organisation, 1994.

Panet G., Letouche R., Peugeot C., *De Merise à Merise/2 : techniques de "refinement" et nouvelles architectures d'application*, in *Autour et à l'entour de Merise : les méthodes de conception en perspective,* Actes du congrès AFCET-CERAM, 17-18-19 avril 1991, Sophia-Antipolis, publication AFCET, 1991.

Panet G., Letouche R., *Merise/2 : Modèles et techniques Merise avancés,* éditions d'Organisation, 1994.

Pham Thu Quang, *Merise et Yourdon, in Génie logiciel et systèmes experts,* n° 15, septembre 1989, pages 22 à 39, publication EC2, 1989.

Pham Thu Quang, Chartier-Kastler C., *Merise appliquée,* Eyrolles, Paris, 1990.

Rochfeld A., Morejon, J., *la Méthode Merise,* tome III : *Gamme opératoire,* éditions d'Organisation, 1989.

Rochfeld A., Bouzeghoub M., *From MERISE to OOM*, in Ingénierie des systèmes d'information, vo.1 N°2, pp 151-176, 1993.

Rolland C., Foucaut O., Benci G., *Conception des systèmes d'information : la méthode Remora,* Eyrolles, Paris, 1988.

Sandoval V., (1), *L'échange de données informatisé pour l'entreprise*, éditions Hermès, Paris, 1990.

Sandoval V., (2), *Technologie de l'EDI,* éditions Hermès, Paris, 1990.

Saphin G., Delaroche S., Lebouvier S., André J., *Le formalisme de données de Merise : extensions du pouvoir d'expression*, compte rendu de la journée d'étude du 15 novembre 90, organisée par le groupe 135 de l'AFCET, AFCET/Interfaces n° 101, mars 1991.

SLIGOS, *Manuel de référence de la méthode Minos*.

Stoven B., Deturche M., *Conduite de projet EDIFACT*, Simprofrance, Paris, 1990.

Stoven B., *EDIFACT*, Simprofrance, Paris, 2e édition, 1990.

Tabourier Y., Nanci D., *The Occurrence Sructure Concept : an Approach to Structural Integrity Constraints in the Entity Relationship Model*, in *Proceddings of Entity Relationship Conference*, 1981.

Tabourier Y., *De l'autre côté de Merise*, éditions d'Organisation, 1989.

Tabourier Y., *Les extensions au formalisme de données Merise*, Document de cours présenté au congrès "Autour et à l'entour de Merise", AFCET/CERAM, le 17 avril 1991.

Tardieu H., Nanci D., Heckenroth H., *Méthode, modèle et outils pour la conception de la base de données d'un système d'information*, Rapport interne du centre d'études techniques de l'équipement, Aix-en-Provence, 1975.

Tardieu H., Nanci D., Heckenroth H., *Méthode, modèles et outils pour la conception de la base de données d'un système d'information*, Rapports de recherche IRIA 74180-77187, CETE D'AIX-GRASCE/CNRS, université d'Aix-Marseille III, 1975, 1976, 1977, 1978.

Tardieu H., Nanci D., Pascot D, *Conception d'un système d'information. Construction de la base de données,* éditions d'Organisation, 1979.

Tardieu H., Pascot D., Nanci D., Heckenroth H., *A Method, a Formalism and Tools for Data Base Design,* in *Proceedings of Entity Relationship Conference,* 1979.

Tardieu H., *De l'analyse des applications à la conception des systèmes d'information,* Informatique et gestion n° 117, sept. 1980.

Tardieu H., Rochfeld A., Coletti R., *La méthode Merise,* tome 1 : *Principes et outils,* éditions d'Organisation, 1983.

Tardieu H., Rochfeld A., Coletti R., Panet G., Vahee G., *la Méthode Merise,* tome 2 *: Démarche et pratiques,* éditions d'Organisation, 1985.

Techniques et Sciences Informatiques, numéro spécial *Réseaux de Petri*, volume 4, n° 1, janvier-février 1985, Dunod-AFCET Informatique, 1985.

TRANSPAC, *Communication et échanges électroniques interentreprises*, Collection Transpac, A jour éditeur, compte rendu du forum Transpac EDI des 25 et 26 octobre 1988, Paris.

Vogel C., *Génie cognitif : la méthode KOD*, Masson, 1988.

Références sur la théorie des systèmes, l'organisation, le management et divers apports conceptuels

Actes des congrès sur l'approche Entity Relationship, 1979 publié en 1980 par North Holland.

ANSI:X3:SPARC, *Study Group ou Data Base Management Sys*tems : Interim report 785-02-08, ACM SIGMOD Newsletter, vol 7, n° 2, 1975.

Anthony R.N., *Planning and Control Sys*tems : a Framework for Analysis, Cambridge, Mass, Harvard University Press, 1966.

Bartoli J.-A., Le Moigne J.-L., éditeurs, Organisation intelligente et système d'information stratégique, éditions Economica, Paris, 1996.

Bouchy S., *L'ingénierie des systèmes d'information évolutifs*, Eyrolles, Paris, 1994.

Boulanger P., *Organiser l'entreprise en réseau*, éditions Nathan, Paris, 1995.

Boulding K.E., *General System Th*eory : the Skeletton of Science, Management Science, 1956.

Browne J., Sackett J.C., Wortmann, Industry Requirements and Associated Research Issue in the Extend Entreprise, European Workshop on Integrated Manufacturing Systems Engineering, IMSE 94, Grenoble, France, December 12-14, 1994.

Brunet S., Gardin H., *Pratiques du reengineering : redessine-moi l'entreprise*, ESF éditeur, Paris, 1995.

CECAR-CTI, ministère de l'Industrie.

Cohen J., *Décupler sa productivité avec le BPR*, 01 Réseaux, n° 17, sept. 1995.

Davenport T., *Process Innovation : reengineering work through information technology*, Harvard Business School Press, Boston (US), 1993.

De Terssac G., Dubois P., Les nouvelles rationnalisations de la production, Cépadues-Editions, 1992.

Deen S.M., Cooperation Issues in Holonic Manufactiring Systems, Information Infrastructure Systems for Manufacturing (B-14), in Yoshikawa H. and Goossenaerts J. Editors, Elsevier Science B.V., North Holland, 1993.

Dent H., *Job choc*, traduction française, éditions générales F1RST, Paris, 1995.

Ettighoffer D., *L'entreprise virtuelle, ou les nouveaux modes de travail*, éditions Odile Jacob, Paris, 1992.

Flores F., Graves M., Hartfield B., Winograd T., *Computer systems and the design of organizational interaction*, ACM Transactions on Office Information Systems, avril 1988, pp. 153 - 172.

Gorry A., Scott Morton M., *A Framework for Management Information System, Sloan Management Review, 1971.*

Gouillart F., Kelly J., Du mécanique au vivant, L'entreprise en transformation, éditions Village Mondial, Paris, 1995.

Grouard B., Meston F., *L'entreprise en mouvement*, Dunod, Paris, 1993.

*Guide R*acines*, La Documentation Française, 1982.*

Hammer M., Champy J., *Le reengineering*, traduction française, Dunod, Paris, 1993.

Herath A., Towards a CKBS Model for Holonic Manufacturing Environments, Report N° DAKE/-/TR-94002, DAKE Centre, University of Keele, United Kingdom, 1994.

Jacob G., *Le reengineering : l'entreprise reconfigurée*, Hermès, Paris, 1994.

Jacobson I., Ericsson M., Jacobson A., *The object advantage, Business Process Reengineering with object technology*, ACM Press Book, Addison-Wesley, 1994-1995.

Johansson H., McHugh P., Pendlebury A., Wheeler W., *Business process reengineering : breakpoint strategies for market dominance*, John Wiley & Sons Ltd, Chichester (UK), 1993.

Le Moigne J.L, *La modélisation des systèmes complexes,* éditions Dunod, Paris, 1990.

Le Moigne J.L, *les Systèmes de décision dans l'organisation*, PUF, Paris, 1974.

Le Moigne J.L, *Théorie du système général, théorie de la modélisation,* PUF, Paris, 1977.

MacDonald H., *Développement, mise à niveau et réétude de la stratégie de l'entreprise*, in Scott Morton M., *L'entreprise compétitive au futur : technologies de l'information et transformation de l'organisation*, traduction française, éditions d'Organisation, Paris, 1995.

Marshall Ch., Enterprise Modeling with UML, Designing Successful Software through Business Analysis, Addison-Wesley, 2000.

Maturana H., Varela F., *The tree of knowledge*, New Science Library, Boston (US), 1987

Medina-Mora R., Winograd T., Flores R., Flores F., *The action workflow approach to workflow management technology*, in *CSCW 92 Proceedings*, nov. 1992.

Mélèse J., *Approches systémique des organisations,* 1979, éditions Hommes et Techniques, Paris.

Mélèse J., *L'analyse modulaire des systèmes AMS,* éditions Hommes et Techniques, Paris, 1972.

Molina A., Al-Ashaab, Elis T.I.A., Young R.I.M., Bell R., A Review of computer Aided Simultaneous Engineering Systems in Computer Aided Simultaneous Engineering Systems, Manufacturing Engineering Department, Loughbrough University of Technology, England, 1994.

Morin E., *La méthode, tome 1, 2, 3,* Editions du Seuil, 1977, 1980, 1986.

Morris D., Brandon J., *Re-engineering your business*, McGraw-Hill, New York (US), 1993.

Oesterle H., *Business Engineering : Prozess- und Systementwicklung*, Springer Verlag, Berlin (DE), 1995.

Olivier J.L., *Planification des systèmes d'information,* in *Systèmes d'information stratégique ; pratique d'entreprises et pédagogie,* pages 58 à 100, FNEGE, ICHEC Bruxelles, HEC Liège, 1984.

Paché G., Paraponaris C., *L'entreprise en réseau*, Presses Universitaires de France, Paris, 1993.

Porter M.E., *Competitive Advantage, The Free Press, Macmillan Pub.*, 1985, traduction française : *L'avantage concurrentiel : comment devancer ses concurrents et maintenir son avance,* InterEditions, 1990.

Porter M.E., *Competitive Strategy, The Free Press, Macmillan Pub., 1980,* traduction française : *Choix stratégique et concurrences : techniques d'analyse des secteurs de la concurrence dans l'industrie,* Economica, 1990.

Probst G., Mercier J.-Y., Bruggimann O., Rakotobarison A., *Gérer le changement organisationnel*, in *Organisation et management*, tome 2, éditions d'Organisation, Paris, 1992.

Probst G., Ulrich H., *Pensée globale et management : résoudre les problèmes complexes*, éditions d'Organisation, Paris, 1989.

Roberts B., Flight G., *The enabling role of EDI in business process re-engineering*, in *Electronic Commerce - Electronic Partnership*, the 7 th Int. Conf. *Electronic Data Interchange & Interorganizational Systems*, Bled (Slovenia), 1994.

Rockart J., Bullin C., *A Primer on Critical Success Factor*, CISR working paper Sloan School of Management, MIT, 1981.

Rockart J., Treacy M., *The CEO goes on live*, Havard business review, janvier-février, 1982.

Rolstadås A., Beyond Year 2000: Production Management in the Virtual Company, IFIP WG5.7, Working Conference on Evaluation of Production Methods, Gramado, Brazil, march 21-29, 1994.

Sandoval V., *Les techniques du reengineering*, Hermès, Paris, 1994.

Scott Morton M., *L'entreprise compétitive au futur : technologies de l'information et transformation de l'organisation*, traduction française, éditions d'Organisation, Paris, 1995.

Simon H.A., *Sciences des systèmes ; sciences de l'artificiel*, traduction française, Nouvelle édition, Dunod, Paris, 1991.

Skjellaug B., Hämmäinen H., Saathoff H., Enterprises and Distributed CIM : Inter-organizational Communication, Computers in Industry , Vol. 9, 1990.

Tabourier Y., *Le "business reengineering" et ses modèles*, cours 4, Congrès "Merise et les autres", AFCET, 1994.

Tapscott D., Caston A., *L'entreprise de la deuxième ère*, traduction française, Dunod, Paris, 1994.

Tardieu H., Guthmann B., *Le triangle stratégique : stratégie, structure et technologie de l'information*, éditions d'Organisation, 1991.

Tardieu H., Guthmann B., *Le triangle stratégique : stratégie, structure et technologie de l'information*, éditions d'Organisation, Paris, 1991.

Tardieu H., *Système d'information-stratégiques et systèmes-d'information stratégiques*, in *Autour et à l'entour de Merise : les méthodes de conception en perspective*, Actes du congrès AFCET-CERAM, 17-18-19 avril 1991, Sophia-Antipolis, publication AFCET, 1991.

Van Griethuysen J.J., ISO TC97 SC5 WG3, *Concepts and Terminology for the Conceptual Schema and the Information Base*, 1982.

Vernadat F., Future R&D Directions for CIM Deployment, European Workshop on Integrated Manufacuring Systems Engineering, pp. 3-6, Grenoble, France, December, 1994.

Vogel C., Génie cognitif, Masson, Paris, 1988.

Warnecke H.-J., *Aufbruch zum fraktalen Unternehmen : Praxisbeispiele für neues Denken und Handeln*, Springer Verlag, Berlin (DE), 1995.

Warnecke H.-J., *Revolution der Unternehmenskultur : das fraktale Unternehmen*, Springer Verlag, Berlin (DE), 1993.

Weawer W., Shannon C.E., *Théorie mathématique de la communication*, traduction française, RETZ-CEPL, Paris, 1975.

Wilenski R., *Planning and Understanding a Computational Approach to Human Reasoning*, Addison Wesley Publishing compagny, Readings USA, 1983.

Wiseman Ch., *Strategic Information System*, Irwin Inc, Illinois, 1988.

Wiseman Ch., *Strategy and Computer*, Dow Jones-Irwin Inc, 1985, traduction française : *L'informatique stratégique, un nouvel atout de la compétitivité*, éditions d'organisation, Paris, 1987.

Références sur le génie logiciel et les bases de données

AFCET-Interfaces n° 96 octobre 90, enquête génie logiciel.

Bachman C.W., *Data Structures Diagrams*, Data Base Journal of ACM SIGBDP, 1969.

Barthet M.F., *Logiciels interactifs et ergonomie*, Dunod, Paris, 1988.

Booch G. , *Conception orientée objet et application*, Addison Wesley, 1992

Booch G., *Ingénierie du logiciel avec Ada*, InterEditions, 1988.

Booch G., Jacobson I., Rumbaugh J., The Unified Modeling Language for Object Oriented Development, version 0.91, Rational Software Corporation, 1996.

Booch G., *Object-Oriented Analysis and Design with Applications* , second edition, The Benjamin Cummings Publishing Company, 1993.

Bouché M., *La démarche objet: concepts et outils* , AFNOR, 1994.

Bourgeois J., *Ateliers de génie logiciel : état de l'art et perspectives*, in Revue de génie logiciel, *Ateliers logiciels*, n° 1, Agence de l'informatique, numéro 1, 1985.

Bouzeghoub M., Gardarin G., Valduriez P., *Du C++ à Merise objet: Objets*, Eyrolles, 1994.

Bouzeghoub M., Jouve M., Pucheral P., *Systèmes de bases de données : des techniques à la conception de schémas*, éditions Eyrolles, Paris, 1990.

Carey J., Carlson B., Graser T., *San Francisco Design Patterns : Blueprints for Business Software*, Addison Wesley, 2000.

Castellani X., *Méthodologie générale d'analyse et de conception des systèmes d'objets ; tome 1 : L'ingénierie des besoins*, Masson, 1993.

Cauvet C., Rieu D., Front-Conte A., Ramadour Ph., *Réutilisation dans l'ingénierie des systèmes d'information*, in Ingénierie des systèmes d'information, Coordinateurs : C. Rosental-Sabroux et Corine Cauvet, Hermés Editeur, à paraître en 2001.

Cauvet C., Rolland C., *Conception des systèmes d'information : une approche orientée objet*, in Autour et à l'entour de MERISE : les méthodes de conception en perspective, Actes du congrès AFCET-CERAM, 17-18-19 avril 1991, Sophia-Antipolis, publication AFCET, 1991.

Chen P., *The Entity-Relationship Model Toward a Unified View of Data*, ACM TODS, vol. 1, n° 1, 1976.

Coad P., Yourdon Y., traduction française de A. Boughlan, *Analyse Orientée Objets*, Masson, 1992.

Codd E.F., *A Relational Model for Large Shared Data Banks*, Comunication of ACM, V13, n° 6, june 1970.

Codd E.F., *Further Normaliation of the Relational Model*, Proc. Computer Science Symposium, R.Rustin Ed., Printice-Hall, 1971.

Coleman D., Arnold P., Bodoff S., Dollin C., Gilchrist H., Hayes J., Jeremaes P., *Fusion : la méthode orientée objet de 2e génération*, Masson, 1996.

Date C., *Introduction au standard SQL*, Addison-Wesley Europe, InterEditions, 1989.

De Marco T., *Structured Analysis and System Specification*, Yourdon Inc., New York, 1978.

Delobel C., Adibas M., *Bases de données et systèmes relationnels*, éditions Dunod, 1982.

Demontmollin M., *L'ergonomie*, éditions de la Découverte, Paris, 1986.

Espinasse B., Lai M., Nanci D., *Merise+ : Une extension de la méthode MERISE à l'approche objet par un apport de la méthode HOOD,* Revue Ingénierie des Systèmes d'Information, Hermès Editeur, Volume 3 - n°2-3, 1995.

Espinasse B., Nanci D., *Merise et l'approche orientée objet : du couplage avec OMT à une troisième génération*, Revue Ingénierie des Systèmes d'Information, Hermès Editeur, Vol. 5, n°4, oct. 97.

Everest G.C., *Characteristics of Inter-Entity Relationschips and a Graphical Notation*, MISRC-WP-77-04, MIS Research Center, University of MN, 1977.

Flory A., *Bases de données : conception et réalisation*, éditions Economica, Paris, 1982.

Foucaut O., Thiery O., Smaïl K., *Conception de systèmes d'information et programmation événementielle*, InterEditions, 1996.

Gamma E, Helm R., Johnson R., Vlissides J., Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing Company, 1995.

Gane C., Sarson T., *Structured Analysis : Tools and Techniques*, New York, Improved System Technologies, 1979, traduction française : *Analyse structurée des systèmes : outils et techniques*, IST, Gland, Suisse.

Gardarin G. & O., *Le client-serveur,* Eyrolles, 1996.

Gardarin G., *Bases de données : les systèmes et leurs langages*, éditions Eyrolles, Paris, 1983.

Gardarin G., Valduriez P., *Les systèmes de gestion de bases de données relationnels*, édition Eyrolles, 1985.

Gardarin G., Valduriez P., *SGBD avancés : bases de données objets, déductives, réparties*, éditions Eyrolles, Paris, 1990.

Gendre P., Bitteur H., *HOOD, version 3.0 : l''âge de raison*, Génie Logiciel & Systèmes Experts, n° 19, juin 1990.

Groupe AFECT/LBD4G, *Langages de quatrième génération*, éditions Dunod Informatique/AFCET, Paris, 1988.

Guelaud F., Beauchesne M.N., Gautrat J., Roustang G., *Pour une analyse des conditions du travail ouvrier dans l'entreprise*, recherche du Laboratoire d'économie et de sociologie du travail (LEST), CNRS, Armand Colin, 1975.

Heitz M., (CISI Ingénierie Toulouse), *HOOD, une méthode de conception hiérarchisée orientée objets pour le développement des gros logiciels techniques et temps réel*, Bigre n° 57, Journées Ada France, déc. 87.

HOOD *Reference Manual*, version 3.1.1, HOOD User Group, june 92.

IBM, AD CYCLE, 1989, *Composantes de l'offre*.

Jacobson I., Christerson M., Jonsson P., Övergaard G., *Object-Orented Software Engineering - A use Case Driven Approach*, Reading MA: Addison-Wesley, 1992.

Jacobson I., Ericsson M., Jacobson A., *The object advantage, Business Process Reengineering with object technology*, ACM Press Book, Addison-Wesley, 1994-1995.

Johnson R., "How to design frameworks", Tutorial Notes, OOPSLA'93, 1993.

Kang K., Cohen S., Hess J., Novak W., Peterson S., Feature-oriented domain analysis (FODA) feasibility study CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1990.

Kiely D., "Art Components: the Future of Software ?", Computer, février 1998.

Krakoviak S., *Systèmes intégrés de production de logiciel : concepts et réalisations*, TSI, AFCET-Bordas, 1982.

Laboratoire MASI, Université de Versailles, *Bases de Données Actives*, Paris, Journée du 8 décembre 93, publication AFCET.

Labreuille B., Farail P., *Application d'Ada aux systèmes temps réel distribués*, Bigre n° 57, Journées Ada France, déc. 1987.

Lai M., *Conception Orientée Objet : Pratique de la méthode HOOD*, Editions Dunod Informatique, 1991, 320 pages.

Laville A., *L'ergonomie*, collection Que sais-je ?, PUF, Paris, 1981.

Lefebvre A., *L'architecture client-serveur,* Armand Colin, 1993.

Marée C., Ledant G., *SQL 2: initiation Programmation,* Armand Colin, 1995.

Miranda S., Ruols A., *Client-serveur: concepts, moteurs SQL et architectures parrallèles,* Eyrolles, 1994.

Muller, P-A., *Modélisation objet avec UML*, Eyrolles, 1997.

Parnas D.L., *On the Criteria to be Using in Decomposing System into Modules*, CACM, vol. 5, #12, décembre. 72, *p. 1053.*

PCTE, 1988, *A Basis for a Portable Common Tool Environment, Functional Specification,* 15 novembre 1988, Commission of the European Communities, Brussels.

Perry T.J., Lateer J.G., *Oracle par la pratique*, éditions Sybex, Paris, 1990.

Richard D., *Une méthodologie de conception physique de base de données pour le système expert SECSI*, thèse de l'université Paris VI, 1989.

Ridjanovic J., *Notes de cours en MBA*, département systèmes d'information organisationnels, faculté des sciences de l'administration, université Laval, Québec, Canada, 1983.

Rolland C., Foucaut O., Benci G., *Conception des systèmes d'information, la méthode Remora*, Eyrolles, 1987.

Ross D., *Structured Analysis (SA) : A Language for Communicating Ideas,* IEEE Transactions on Software Engineering, volume SE-3, pp. 16-34, 1977.

Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

Senko M.E., *DIAM as a Detailed Example of the ANSI SPARC Architecture*, in G.M. Nijssen, Ed., *Modeling in Data Base Management*, Noth-Holland, 1976.

Shlaer S., Mellor S.J., *Object life cycles : Modeling the world in states*, Prentice Hall, 1990.

Simon E., Valduriez O., *Design and Implementation of an Extensible Integrity Subsystem*, Proc. of ACM-SIGMOD Conference, Boston in SIGMOD RECORDS, volume 14, n° 2, juin 1984.

Sodhi J., Sodhi P., Software Reuse, Domain Analysis and Design Process, Software Development, Mc Graw Hill, 1998.

Sperandio J.C., *L'ergonomie du travail mental*, édition Masson, Paris, 1984.

Stevens W.P., Myers G.J., Constantine U., *Structured Design,* IBM System Journal, vol. 13, #12, pp. 115-139.

Valentin A., Lucongsang R., *L'ergonomie des logiciels*, éditions de l'ANACT, 1987.

Warnier J.D., *Construction et transformation de programme, L.P.C.* éditions d'Organisation, Paris.

Yourdon E., Constantine L., *Structured Design,* Prentice Hall, 1979.

Yourdon Ed., Whitehead K., Thomann J., Oppel K., Nevermann P., *Main Stream Object : an Analysis and Design Approach for Business*, Yourdon Press, Prentice Hall, 1995.

—