Revue Ingénierie des Systèmes d'Information, Hermès Editeur, à paraître 1997.

Merise et l'approche orientée objet :
du couplage avec OMT à une troisième génération

Bernard Espinasse * - Dominique Nanci **

** DIAM-IUSPIM, Université Aix-Marseille, Domaine Universitaire de St Jérôme, Avenue Normandie Niemen, 13397 Marseille Cedex 13, France -*
Email : espinas@unix05.u-3mrs.fr
*** CECIMA, 24 avenue de la Molle, 13100 Aix-en-Provence, France.*

RESUME. Dans un contexte concurrentiel de plus en plus vif, les entreprises recherchent un accroissement de flexibilité, d'adaptabilité et de réactivité. De nouveaux types d'organisation apparaissent (entreprise réseau, entreprise étendue, entreprise virtuelle) tous caractérisés par une distribution accrue et supportés par des systèmes d'information aussi distribués. Dans un premier temps nous rappelons tout l'intérêt de l'approche objet dans l'ingénierie de tels systèmes d'information. Ensuite nous étudions comment des méthodes traditionnelles comme Merise, peuvent évoluer vers l'objet pour mieux s'adapter à la conception de ces systèmes. Nous développons deux grandes voies d'évolution possibles de Merise vers l'objet : la première "transition vers l'objet", consiste à son couplage avec OMT, méthode orientée objet de génie logiciel, la seconde voie consiste à une remise en cause profonde conduisant à une troisième génération de Merise.

ABSTRACT. In an increasingly competitive context, enterprises search for an increase of flexibility, adaptability and reactivity. New types of organization appear (network enterprise, extended enterprise, virtual enterprise), all characterized by an increased distribution and supported by equally distributed management information systems. First, we develop the interest of the object oriented approach for the design and the realization of these distributed information systems. Then we study how traditional methods like Merise, can evolve towards the object to be better adapted to the design of these systems. We present two main ways of possible evolution for Merise towards the object : the first one, "transition towards the object", which consist in its coupling with OMT, an object oriented method of software engineering and the other one consists in a profound evolution and leads to a third generation of Merise.

MOTS-CLES: méthode de conception, système d'information, distribution, génie logiciel, Merise, OMT, approche orientée objet, objet.

KEYWORDS : design method, management information system, distribution, software engineering, Merise, OMT, object oriented approach, object.

1. Introduction

Dans cette fin de siècle caractérisée 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, à laquelle le mouvement Business Process Re-engineering (BPR) s'intéresse.

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 ?

La complexification des systèmes d'information n'est pas nouvelle. Depuis plus de quinze ans, Merise a du y faire face et le passage de la première à la deuxième génération en est le témoignage. Ainsi l'extension de trois à quatre niveaux d'abstraction correspond à une évolution profonde des principes fondamentaux de Merise. Elle a permis de mieux distinguer les deux missions distinctes et complémentaires de l'ingénierie des systèmes d'information : la conception du Système d'Information Organisationnel (SIO) d'une part, et celle du Système d'Information Informatisé (SII) d'autre part. Egalement, l'extension des formalismes tant des données que des traitements a permis l'élaboration de modèles de plus en plus aptes à aborder des situations complexes. Pour faire face à ce nouvel accroissement de complexité des systèmes d'information vers plus de distribution et de réactivité, pour tirer parti des nouvelles technologies comme le client-serveur, le multimédia et l'orienté objet, Merise ainsi que les autres méthodes traditionnelles de conception de systèmes d'information doivent encore évoluer.

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 façon encore balbutiante, 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.

Dans un premier temps nous allons montrer pourquoi l'approche objet est adaptée à l'ingénierie des systèmes d'information distribués. Ensuite nous étudions comment des méthodes de conception traditionnelles comme Merise, peuvent évoluer vers l'objet pour mieux s'adapter à la conception de tels systèmes d'informations. Enfin nous présentons en détail deux grandes voies d'évolution possibles : l'une consistant au couplage de Merise avec OMT, méthode orientée objet de génie logiciel et l'autre consistant à une remise en cause profonde de Merise pouvant conduire à une troisième génération.

2. Ingénierie des systèmes d'information et approche objet

L'approche objet est d'abord apparue en génie logiciel pour ensuite tenter de s'imposer au domaine des systèmes d'information et remettre en cause des méthodes traditionnelles comme Merise.

2.1. L'approche objet en génie logiciel

Le développement d'applications informatiques de plus en plus complexes a conduit à revoir la façon de programmer avec l'émergence des langages de programmation orientés objet. L'approche objet constitue ainsi pour les informaticiens une révolution interne remettant en cause leurs méthodes et outils et permettant d'accroître la modularité, la fiabilité et la réutilisabilité des composants logiciels, ceci sans impact pour l'utilisateur final.

En cette 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.

Pour aider l'informaticien à concevoir ces applications informatisées et mettre en oeuvre ces langages, des méthodes orientées objet ont vu le jour. Ces méthodes ont pour objectif essentiel d'aider l'informaticien à concevoir l'architecture logicielle de ses applications informatiques qui seront ensuite programmées en langages objets (de plus en plus de façon assistée par des outils de génie logiciel) et exploiteront des bases de données.

De telles méthodes commencent à être utilisées dans le développement d'applications en informatique de gestion. Notons qu'à cette programmation par objets au travers de langages comme C++, SmallTalk ou d'autres plus spécifiques à des environnements de quatrième génération associés à des SGBD, se rajoute la programmation événementielle qui nous semble avoir aussi considérablement fait évoluer la façon de développer des applications et nécessiter du support méthodologique (cf. notamment [Foucault &al.96], [Rolland &al.87], [Cauvet &al.91]).

Toujours d'un point de vue méthodologique, les concepts de classe et d'héritage supportés par les langages Objets donnent un visage nouveau au prototypage. Des itérations successives rapprochées permettent un prototypage évolutif. Celui-ci facilite la participation des utilisateurs et accroissent la pertinence des spécifications, tout en diminuant le volume pendant les phases d'analyse et de conception.

Dans cette orientation objet des méthodes du génie logiciel (conception et réalisation orientées objet), il faut reconnaître l'influence considérable et sans cesse croissante des méthodes d'origine nord-américaine. Citons notamment OOD de Booch [Booch 93], OOA de Coad & Yourdon [Coad & Yourdon 92], OOA/OOD de Shlaer & Mellor [Shlaer & Mellor 90], Fusion [Coleman &al.94], OOSE de Jacobson [Jacobson &al.92] et depuis quelques années celle de plus en plus dominante d'OMT de Rumbaugh [Rumbaugh & al. 91]. Il est important de préciser que toutes ces méthodes (excepté peut être OOSE) sont des méthodes de génie logiciel, conçues pour résoudre des problèmes de génie logiciel et non pas des méthodes de conception de systèmes d'information comme peut l'être Merise.

Ainsi l'approche objet en génie logiciel est porteuse d'espoirs commençant à ce confirmer en matière de productivité, d'adéquation aux technologies nouvelles mais aussi en terme de démarche participative.

2.2. L'approche objet en systèmes d'information

Ces évolutions vers l'objet des technologies de l'information, ainsi que celle des méthodes permettant de les mettre en oeuvre, sont fondamentales pour l'ingénierie des systèmes d'information. Elles permettent de concevoir et réaliser, souvent plus rapidement, des applications de gestion de plus en plus conviviales, sophistiquées et la plupart du temps plus faciles à maintenir et à faire évoluer. Ce succès informatique associé à l'objet est pour l'instant prometteur essentiellement au niveau du SII. Aussi il faut se demander si cette approche objet peut également présenter le même intérêt au niveau du SIO, et ainsi conduire à une évolution vers l'objet des méthodes de conception de systèmes d'information comme Merise.

Merise a déjà, par le passé, assimilé d'importantes évolutions technologiques, comme le passage des bases données navigationnelles aux bases de données relationnelles et plus récemment la prise en compte des architectures client-serveur. En ce qui concerne l'objet et dans une moindre mesure, on constate sur le terrain que de nombreuses études menées avec Merise conduisent déjà à des réalisations informatiques mettant en oeuvre des langages et méthodes objets du génie logiciel. La plupart du temps, la conception du SIO est conduite de façon classique en utilisant les modélisations conceptuelles et organisationnelles proposées par Merise et seules les modélisations de niveaux logiques et physiques, associées au SII, sont effectuées avec de formalismes spécifiques aux méthodes objets.

La question qui se pose alors est bien celle de l'évolution de Merise vers l'objet. Cette évolution vers l'objet ne peut sérieusement être justifiée par des faiblesses ou lacunes de la méthode Merise au niveau de la conception du SIO. Quinze années d'expérience témoignent que cette méthode, au travers des modèles conceptuels et organisationnels, a relativement bien réussi dans la spécification de systèmes d'information organisationnels, concourant à concilier informatique et organisation. Ce n'est donc pas la dichotomie Données-Traitements (que nous préférons d'ailleurs appeler Données-Activités), actuellement parfaitement maîtrisée par les praticiens, qui justifierait l'abandon de Merise. Mentionnons à ce propos les problèmes que rencontrent actuellement toutes les méthodes objets de génie logiciel pour gérer trois modélisations complémentaires (statique, dynamique et fonctionnelle). Il faut toutefois objectivement reconnaître que Merise a mis très longtemps à proposer un cadre, encore limité, à la conception logicielle du SII; mais rappelons que Merise ne s'était pas donné cet objectif principal, et le laissait à d'autres méthodes. On rappellera, qu'à sa création, le nom de Merise évoque, par allégorie au merisier, "un tronc commun méthodologique sur lequel viendront se greffer d'autres méthodes...".

L'approche objet dans les méthodes de conception de système d'information, et donc une éventuelle évolution de Merise vers l'objet, nous semble avoir deux intérêts essentiels : la réutilisation d'objets et la continuité méthodologique (démarche et formalismes) centrée sur la notion d'objet. 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).

L'évolution vers l'objet d'une méthode de conception de systèmes d'information comme Merise est probablement souhaitable. Cependant, ayons bien conscience que l'adoption et la mise en œuvre d'un ensemble méthodologique comme Merise a constitué pour un très grand nombre d'entreprises et de sociétés de services en informatique (S.S.I.I.) un investissement considérable, tant financier qu'humain, qui ne peut être remis en cause du jour au lendemain de façon radicale sans qu'il y ait d'alternative crédible proposée.

Sans entrer dans une polémique qui pourrait apparaître à certains d'arrière garde, nous voudrions toutefois souligner une possible conséquence à moyen et long termes de l'abandon de Merise au profit du "tout objet". Comme nous avons essayé à maintes reprises de le mettre en évidence, la méthode Merise s'est essentiellement située sur le terrain de la conception du SIO., à travers la modélisation des activités (processus et organisation) et de la sémantique des informations, facilitant ainsi le passage vers la conception logicielle, en particulier les bases de données. Dans cette "ingénierie des besoins" tournée vers les utilisateurs, de très nombreux concepteurs (dans les entreprises, administrations et SSII) ont acquis en ce domaine une expérience certaine et unique Par contre, les méthodes objets actuellement proposées se focalisent indéniablement sur la partie SII. En changeant de savoir faire (de Merise au tout objet), ces concepteurs vont progressivement abandonner toute la partie SIO, dont l'importance est par ailleurs soulignée de plus en plus. On peut aisément imaginer qu'ensuite, des méthodes de même origine que les méthodes objet, viendront facilement proposer de "nouvelles" solutions à une problématique laissée en jachère.

2.3. Evolutions possibles de Merise vers l'objet

Nous avons précédemment évoqué deux raisons justifiant l'évolution des méthodes de conception de systèmes d'information vers l'objet : d'une part la réutilisation d'objets au niveau de la conception du SIO, d'autre part une continuité méthodologique centrée sur l'objet entre la conception du SIO et celle du SII.

A ce jour, nous estimons que les réflexions et propositions des spécialistes du domaine n'ont pas atteint, sur ce sujet, la convergence nécessaire à une solution opérationnelle. Aussi nous proposons deux voies d'évolution possibles de Merise et des méthodes de conception de systèmes d'information traditionnelles vers l'objet :

- 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, plus radicale, 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. Aussi, nous développerons, dans les paragraphes suivants, les grandes lignes d'une telle alternative qui associerait Merise à la méthode de génie logiciel OMT.

Dans la seconde voie, bien des aspects sont actuellement imprécis et relèvent encore de la recherche. Aussi, nous nous contenterons de ne donner ici que quelques pistes de réflexion nous permettant d'introduire les prémisses d'une méthode Merise d'une troisième génération.

3. Une transition vers l'objet : le couplage Merise-OMT

Cette solution consiste à coupler Merise avec une méthode de génie logiciel orientée objet. Une première tentative a été faite il y a quelques années entre la méthode Merise et la méthode HOOD (Hierarchical Object Oriented Design) [HOOD 92]. Ce projet, subventionné par le Ministère de l'Industrie dans le cadre du programme "Informatique 92" a associé plusieurs partenaires industriels (les sociétés Cecima, Ingénia et Systémia) et un partenaire universitaire (l'Université Aix-Marseille III). Les limitations de la méthode HOOD (entre autres absence de la notion d'héritage) n'ont pas permis d'effectuer un couplage satisfaisant [Espinasse &al.93]. Néanmoins, ce projet de recherche nous a offert l'occasion d'étudier les modalités d'un tel couplage et nous permet aujourd'hui de proposer une approche équivalente entre la méthode Merise et la méthode OMT, qui s'est avérée mieux adaptée que HOOD. Après une succincte mais indispensable présentation d'OMT, nous développerons les modalités d'un tel couplage.

3.1. Présentation générale d'OMT

La méthode OMT (Object Modelling Technique) a été à l'origine développée au sein de la Société General Electric à la fin des années 80, par Rumbaugh, Blaha, Premerlani, Eddy et Lorensen. Un premier ouvrage présentant cette méthode a été publié en 1991 [Rumbaugh &al.91]. Cette méthode est en constante évolution avec un premier rapprochement avec la méthode OOD (Object Oriented Design) de G.Booch [Booch 93] et plus récemment de la méthode suédoise OOSE (Object Oriented Software Engineering), initialement connue sous le nom de Objectory, proposée par Jacobson et son équipe [Jacobson 92]. La méthode OMT connaît actuellement un grand succès tant en Amérique du Nord qu'en Europe.

Nous nous limiterons à un simple rappel des notions principales de cette méthode; pour plus de détail, nous renvoyons le lecteur à des ouvrages qui lui y sont intégralement consacrés [Rumbaugh &al.94]. Citons également d'autres ouvrage introduisant OMT et la comparant avec d'autres méthodes de génie logiciel orientées objets, notamment [Bouzeghoub &al.94].

Comme sa "soeur" OOA [Shlaer & Mellor 90], OMT respecte dans sa philosophie générale, les trois dimensions de modélisation statique ou objet, dynamique et fonctionnelle. OMT propose ainsi trois modèles principaux associés à ces dimensions. Nous présenterons de façon succincte les deux aspects principaux de la méthode, à savoir les trois modèles proposés et la démarche générale préconisée.

3.1.1. Le modèle objet d'OMT

Le modèle objet (M.O.) est le modèle le plus important des modèles proposé par OMT. Ce modèle se présente comme une extension du modèle Entité-Association. Notons à ce propos, qu'il est maintenant très voisin de celui déjà proposé par Merise 2ème génération, notamment par ses concepts, sa représentation graphique, et par le fait qu'il introduise les associations n-aires et les associations porteuses d'attributs (ce que la plupart des modèles Entité-Association anglo-saxons refusent). L'extension vers l'objet de l'Entité-Relation proposée par OMT est classique; elle consiste à introduire l'agrégation, la généralisation et surtout la spécification d'opérations et de contraintes au niveau des entités, nommées ici "Classes".

Classes et Instances

Dans OMT, une Classe d'objets modélise un ensemble d'objets ayant des propriétés similaires (Attributs), des comportements similaires (Opérations). Le diagramme ou modèle de classes présente ces classes, leurs hiérarchies et leurs relations entre elles. On peut aussi modéliser des instances de ces classes, instances qui permettront d'élaborer des diagrammes d'instances exprimant des scénarios possibles parmi les multiples possibilités représentées dans le diagramme de classes.

Associations binaires et n-aires

Des associations peuvent relier des classes non nécessairement distinctes. On distingue des associations binaires et des associations n-aires auxquelles peuvent être rattachés des attributs et/ou des opérations propres. Les premières sont représentées par un lien, les secondes sont représentées par un symbole spécifique.

Ces associations sont caractérisées par la notion de multiplicité et suivent des conventions. On notera que ces multiplicités sont, sur les associations binaires, écrites à l'autre bout de la ligne (à l'inverse des cardinalités Merise). De ce fait, la notion de multiplicité ne fonctionne plus avec les associations n-aires.

Généralisation , spécialisation et héritage

La généralisation/spécialisation est une relation entre une classe et une ou plusieurs autres classes, nommées sous-classes, qui sont des raffinements de la première, nommée surclasse. La généralisation est une relation ascendante entre classes alors que la spécialisation est une relation descendante. Ces relations peuvent être basées sur un attribut spécifique, nommé attribut distinctif..

Chaque sous-classe hérite des caractéristiques (attributs et opérations) de la surclasse associée. Toute sous-classe peut modifier et/ou ajouter de nouveaux attributs ou opérations par rapport à ceux hérités. La génération comme l'héritage sont transitifs sur un nombre arbitraire de niveaux, on introduit alors les termes de descendants et d'ancêtre. Notons qu'une instance de sous-classe est simultanément une instance de toutes les classes ancêtres.

On peut aussi spécifier des contraintes sur les hiérarchies de classes, comme des contraintes d'intersection/disjonction. Des sous-classes disjointes ont une intersection vide, des sous-classes intersécantes peuvent en avoir une non vide.

Agrégation

L'agrégation est une relation transitive du type Composé-Composant entre classes. Une classe Véhicule est composée par exemple de deux autres classes, les classes Moteur et Carrosserie.

Contraintes

On définit dans OMT de nombreuses contraintes pouvant porter sur des objets, des classes, des attributs, des liaisons et des associations d'un modèle objet. D'une façon générale, ces contraintes restreignent le champ des valeurs possibles des éléments sur lesquels elles portent. Nous n'évoquons ici que deux types de contraintes: les contraintes d'attributs et les contraintes inter-associations. Les *contraintes d'attribut* restreignent les valeurs pouvant être prises par un attribut d'une classe. Les *contraintes inter-association* enrichissent la sémantiques des modèles objet en permettant par exemple l'expression de contrainte d'inclusion - subset.

Clés candidate

On définira dans OMT la clé candidate comme un ensemble minimum d'attribut qui identifie de manière unique un objet ou une association. Ce concept est indispensable dans le cas d'association n-aire.

Autres possibilités en OMT

Il est possible de représenter dans les modèles objet d'OMT d'autres concepts :

- Méta-classes : ce sont des classes permettant de décrire d'autres classes comme des instances;
- Classes abstraites et classes concrètes : une classe abstraite est une classe sans instance mais dont les descendants, nommés classes concrètes, ont des instances;
- Classes jointe : une classe associée à plusieurs surclasses (héritage multiple);
- Classes dérivées : une classe dérivée est une classe dont les instances peuvent êtres calculées à partir d'autres classes;
- Associations dérivées : une association dérivée est une association dont les instances peuvent êtres calculées à partir d'autres associations;

3.1.2. Le modèle dynamique d'OMT

Le modèle dynamique (M.D.) a pour objet de représenter les aspects de l'application affectés par le temps et de spécifier l'ordonnancement de l'exécution des opérations déjà définies dans le modèle objet. On considère qu'une opération se produit sans prendre en compte ni son action, ni l'objet sur lequel elle agit, ni la manière dont elle est accomplie.

Ce modèle se présente graphiquement comme un *diagramme d'états-transitions* spécifiant l'ordonnancement des *états* et des *événements* autorisés pour une classe d'objets. Sur ces diagrammes d'états-transitions sont mentionnés les événements déclencheurs des transitions et les opérations de transformation associées. On parle aussi de cycle de vie des objets. Les événements de ce diagramme correspondent aux opérations définies au niveau des classes et associations du modèle objet. De même, le lien entre ce modèle dynamique et le modèle fonctionnel repose sur les actions qui représentent les fonctions du modèle fonctionnel, comme nous le verrons plus loin.

L'événement

L'événement défini dans OMT est l'unique porteur d'information d'un objet vers un autre objet. Un objet qui envoie un événement vers un autre objet pourra attendre une réponse qui sera elle-même un autre événement. Notons qu'un événement est sans durée et qu'un attribut *date* le caractérise implicitement. Un événement peut consister en un simple signal ou peut avoir des *attributs d'événement* définissant le message qu'il transporte. On peut regrouper des événements de même structure de message et de comportement commun en *classes d'événements*. La structure des classes d'événements est assez similaire à celle des classes d'objet du modèle objet. Les événements peuvent s'enchaîner par des liens de causalité ou survenir de façon concurrente.

Etat et transition

Un état est la valeur d'un objet durant un intervalle de temps ou durant l'occurrence de deux événements. Durant cet intervalle de temps, l'objet peut réaliser une action qui ne change pas son état de façon significative (on pourra accepter des changements d'états réalisés par une telle action non considérés comme remarquables à mémoriser). Dans ce diagramme d'états-transitions, le passage d'un état à un autre se traduit par la modification de la valeur d'un ou de plusieurs attributs de l'objet concerné.

Une transition est décrite par l'événement qui la déclenche, les attributs qui caractérisent cet événement, la condition à vérifier en plus de l'occurrence d'événement et l'action à exécuter. L'action que doit exécuter un objet dans un certain état sera introduite dans le diagramme d'états par le mot clé *do*.

A chaque état remarquable, on associe une description textuelle du type :

Etat: <nom_d'état>
Description: <string>
Séquence d'événements qui produit l'état:
    Evénement -1
    Evénement -2 ...
Conditions qui caractérisent l'état:
    Condition -1
    Condition -2 ...
Evénements qui sont acceptés par l'objet dans cet état:
    Evénement Action
Prochain état:
    ... ...
...

Pour chaque classe d'objets le nécessitant, on développera un diagramme d'états représentant son cycle de vie. Les diagrammes d'états de l'ensemble des classes constituent le modèle dynamique. La figure suivante présente le diagramme d'état associé à un objet *Document (Di)*. Les principaux événements sont ici des clics de souris réalisés par l'utilisateur. Les actions accompagnants chacun des événements (ouvrir, fermer, sélectionner) sont considérées comme atomiques. Les activités *afficher* et *masquer* introduites par le mot clé *do:* sont des activités permanentes (exemple issu de [Bouzeghoub &al.94]).

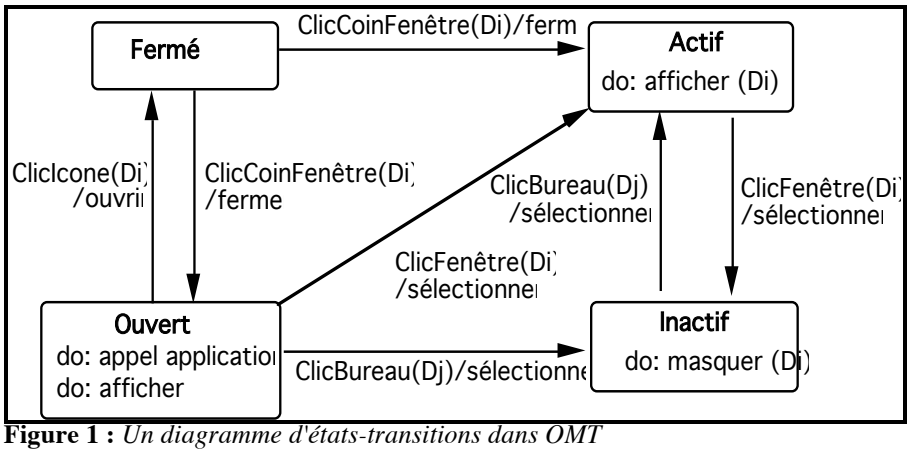


Figure 1 : *Un diagramme d'états-transitions dans OMT*

Notion de scénario

Pour aider le concepteur à identifier les événements et à réaliser les diagrammes d'états, OMT propose une représentation intermédiaire, le *scénario*. Un scénario dans OMT se définie comme une séquence d'événements qui se produit lors d'une exécution du système. Le scénario est au modèle dynamique ce que le modèle d'instance est au modèle objet. Un scénario peut représenter tous les événements d'une application ou ceux qui ont un rapport avec un groupe spécifique d'objets. Un scénario sera représenté par une liste ordonnée de nom d'événements, par exemple :

{événement-1, événement-2, événement-3, événement-4, événement-5, événement-6}

Traceurs d'événement

Un traceur d'événement est un diagramme qui représente la séquence d'événement associée à un scénario mais aussi les objets qui envoient et reçoivent ces événements.

L'ensemble des scénarios décrit l'activité du système. Ces scénarios représentent en quelque sorte les fonctions du système. La synthèse de l'ensemble de ces scénarios permet d'identifier, pour chaque classe d'objets, les événements qui arrivent ou sont émis par cette classe. C'est à partir de ces événements arrivant ou partant que l'on peut élaborer le diagramme d'états d'une classe.

3.1.3. Le modèle fonctionnel d'OMT

Le modèle fonctionnel (M.F.) d'OMT modélise les processus de transformation de l'application, les transformations des valeurs-fonctions, "mapping", contraintes et dépendances fonctionnelles. Ce modèle constitue une formalisation opérationnelle d'un scénario. Il spécifie la signification des opérations et contraintes du modèle objet et les actions du modèle dynamique.

Le MF proposé par OMT se présente comme un diagrammes de flux classique modélisant des *processus* transformant des données, des *flux* de données transportant des données, des objets *acteurs* produisant et consommant des données et enfin des *dépôts* de données qui stockent ces données. Étudions plus en détail chacun de ces concepts :

Processus

D'une façon générale, un processus transforme des données. Un processus peut être contrôlé par une valeur booléenne appelée flot de contrôle, montrant ainsi par exemple la dépendance entre processus. Un processus sera représenté par une ellipse et un flot de contrôle par une flèche en trait pointillé orienté du processus contrôlant vers le processus contrôlé. Pour chaque processus on peut spécifier les opérations mises en oeuvre par celui-ci. On distinguera quatre grands types d'opérations :

- *les opérations d'accès* : elles écrivent ou lisent des attributs ou des liaisons de l'objet auxquelles elles sont rattachées;
- *les opérations requête* : elles sont restreintes aux états visibles d'un objet, c'est à dire sans influence sur les autres objets;
- *les opérations action* : elles produisent des effets sur des autres objets et modifient leurs états. Elles sont normalement dérivées des processus du modèle fonctionnel et peuvent être spécifiées par des formules arithmétiques, des tables/arbres de décision, algorithmes ou par un texte en langage naturel. Elles sont instantanées et donc de durée nulle;
- *les opérations activité* : elles produisent des effets sur des autres objets, modifient leurs états et ont une durée dans le temps;

Flux de données

Un flux de données assure la connexion entre la sortie d'un objet ou d'un processus et l'entrée d'un autre processus. Il représente une valeur intermédiaire d'une données qui peut être une donnée simple ou un agrégat de données. Il sera représenté par une flèche annotée du nom de la donnée ou de l'agrégat de donnés le composant, s'il génère un objet, la flèche se terminera par un triangle blanc.

Acteur

Un acteur est un objet actif qui comporte une opération stimulant un processus ou étant concerné en sortie d'un processus. Il est représenté par un rectangle.

Dépôt de données

Un dépôt de données est un objet passif qui assure le stockage de valeurs pouvant être simples ou agrégées (listes, tables), en vue d'une utilisation ultérieure. Il sera représenté par deux lignes parallèles horizontales, les flèches y entrant représentent les opérations ou les informations modifiant les données mémorisées dans le dépôt, les flèches en sortant représentent les informations demandées au dépôt.

3.2. Démarche méthodologique d'OMT

La démarche méthodologique proposée par OMT repose sur un cycle en cascade composé des phases suivantes : *l'Analyse*, *la Conception système*, *la Conception objet* et enfin *l'Implémentation*. On notera que dans ce cycle en cascade, en accord avec les recommandations de son auteur [Boehm 81], seules les itérations vers les phases immédiatement précédentes sont permises, comme l'illustre la figure suivante :

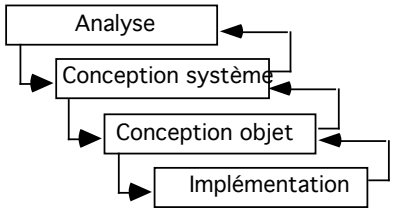


Figure 2 : *Le cycle de développement d'OMT*

3.2.1. La phase d'analyse

Elle a pour objectif de décrire ce que le système à concevoir "doit" faire et non "comment" le faire. Pour cela, on élabore les trois modèles (objet, dynamique et fonctionnel) du système. L'élaboration de ces trois modèles se fait de façon progressive et itérative. Ainsi on commence par élaborer, d'une façon un peu intuitive, le modèle objet en faisant émerger les classes essentielles, puis on élabore les deux autres modèles (dynamique et fonctionnel), pour revenir ensuite sur le modèle objet, afin de l'affiner en introduisant par exemple de nouveaux attributs, opérations et contraintes.

Le Modèle Objet favorise la communication entre les personnes du domaine informatique et les experts du domaine. La phase d'Analyse commence par son élaboration car la structure objet d'un système est normalement la partie la mieux définie. L'élaboration du modèle objet n'est pas un processus linéaire, de nombreux aller-retours avec les deux autres modèles sont nécessaires.

Le Modèle Dynamique spécifie les comportements dans le temps du système et des objets le représentant. Pour élaborer ce modèle on recherche les événements pertinents associés à ces comportements, ainsi que les séquences d'événements valides ou scénarios pour chaque objet que l'on modélisera dans un diagramme d'états.

Le Modèle Fonctionnel spécifie, au travers d'un diagramme de flux, comment les valeurs sont calculées indépendamment de l'ordonnancement, des décisions ou de la structure des objets; Il montre comment les valeurs dépendent les unes des autres, ainsi que les fonctions qui relient ces valeurs. Un processus élémentaire d'un diagramme de flux de données correspond à des activités ou à des actions du diagramme d'états de la modélisation dynamique. De même, un flux correspond à des objets ou à des valeurs d'attributs du modèle d'objets. En conséquence, il est recommandé d'élaborer le modèle fonctionnel après avoir élaboré les deux autres modèles.

Les trois modèles fondamentaux étant construits, il s'agit alors de façon itérative, de vérifier chaque modèle, le faire interagir avec les deux autres. Enfin, une validation avec les experts du domaine d'application est indispensable.

3.2.2. La phase conception système

Cette phase a pour objectif de définir l'architecture globale, les constituants et les ressources du système informatique, "comment" le système doit faire. La phase conception système conduit à décomposer le système en sous-systèmes, identifier des conflits éventuels, de choisir les stratégies de stockages, d'accès aux données et de contrôles à effectuer.

3.2.3. La phase conception objet

La phase de conception objet consiste à spécifier de façon détaillée l'implémentation des objets, compatible avec la technologie retenue pour l'implémentation du système. Les modèles objets, dynamiques et fonctionnels issus de la phase de conception système sont ici encore affinés. Ainsi, l'ensemble des opérations associées aux objets sont complètement spécifiées, notamment leurs algorithmes. Toute opération définie au niveau de l'objet doit apparaître dans le modèle dynamique et tant qu'opération de transformation et en tant que processus élémentaire dans le modèle fonctionnel.

3.3. Couplage des méthodes Merise et OMT

Le couplage de Merise avec des méthodes de génie logiciel n'est pas une idée neuve. Divers couplages, il est vrai plus ou moins heureux, ont déjà vu le jour. Citons entre autres celui de Merise avec la méthode Yourdon [Pham 85]. Rappelons encore qu'à l'origine de Merise, les niveaux logique et physique devaient être supportés par des pratiques ou méthodes déjà utilisées sur le terrain (le merisier...).

L'évolution du cadre méthodologique de Merise de la première à la deuxième génération, à savoir le passage de trois à quatre niveaux d'abstraction, nous semble faciliter ce type de couplage de Merise avec des méthodes de génie logiciel (conception et réalisation logicielles) telle que OMT.

Une comparaison des deux méthodes Merise et OMT est difficile, chaque méthode ayant son propre positionnement et sa propre interprétation des concepts. Merise et OMT bénéficient toutes deux d'une notoriété certaine dans des domaines d'applications différents et sont supportées par des outils logiciels commercialisés. Les deux méthodes comportent une démarche structurée concernant des phases différents du cycle de vie AFNOR. Merise se positionne en amont de OMT. Dans leur mise en œuvre, les deux méthodes ont le soucis de préconiser une répartition du travail et préconisent des procédures d'assurance qualité. Enfin, les deux méthodes n'abordent pas la dimension conduite de projet, pour laquelle elles peuvent s'associer à d'autres méthodes spécialisées.

La présentation succincte d'OMT met en évidence les capacités de structuration d'OMT dans l'élaboration des architectures logicielles. Les aspects "objet" et structuration du logiciel permettent de rénover de façon significative la mise en oeuvre des modèles logiques et physiques de données et de traitements de Merise. Aussi, le couplage des méthodes Merise et OMT que nous préconisons peut se caractériser par un apport d'OMT limité au SII (Système d'Information Informatisé)

et plus particulièrement au niveau logique défini dans Merise. Dans cette perspective,

- pour la conception du SIO, sont conservées les modélisations de la méthode Merise (MCD, MOD, MCT, MOT) ainsi que les étapes de la démarche concernée (étude préalable et étude détaillée);
- pour la conception du SII, la méthode OMT est adoptée en aménageant sa démarche et ses raisonnements pour assurer une interface avec les modélisations issues du SIO.

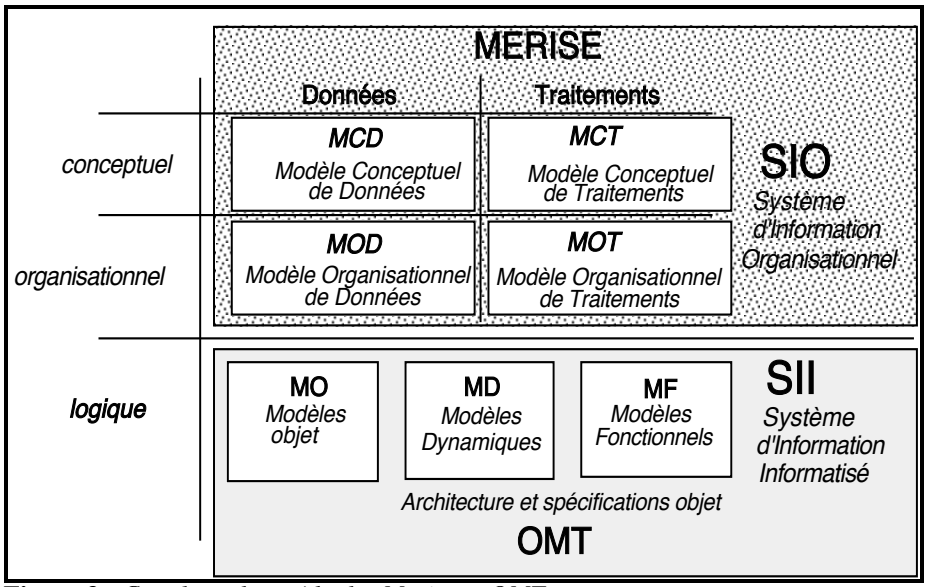


Figure 3 : *Couplage des méthodes Merise et OMT*

3.4. Démarche de passage de Merise à OMT

Ainsi, d'une spécification du SIO selon les modèles proposés par Merise, il s'agit de passer à une spécification du SII orientée objet en OMT. Des éléments de démarche définissant les modalités d'un tel passage peuvent être avancés. Comme nous venons de le voir, les trois grands modèles d'OMT, le Modèle Objet (MO), le Modèle Dynamique (MD) et le Modèle Fonctionnel (MF) sont en forte interaction et doivent être développés de façon progressive et en parallèle.

L'apport d'OMT défini dans une problématique purement génie logiciel, ne peut s'articuler directement avec les spécifications standards du niveau organisationnel de Merise, une transition est nécessaire. Cette transition demande tout d'abord un affinement des modèles organisationnels de Merise et ensuite un développement progressif des modèles OMT. Nous donnons quelques éléments de démarche relatifs à l'affinements des modèles organisationnels Merise et à l'élaboration de chacun des modèles d'OMT à partir de ces modèles affinés.

3.4.1. Affinement des modèles organisationnels de Merise

En ce qui concerne les MOT (Modèles Organisationnels de Traitements), nous proposons au concepteur, que tout en restant à un même niveau de préoccupation (organisationnel de traitement), de développer un MOT suffisamment détaillé

permettant de faire émerger pour chaque tâche, des activités suffisamment fines pouvant être associées à un sous-ensemble de données, appelé dans Merise, Sous-Schéma de Données (S.S.D.). Toujours au niveau de ces tâches élémentaires, on précisera les états et les règles de traitement utilisés. Les CVO (Cycles de Vie d'Objet) élaborés conjointement aux MOT fourniront également les bases permettant le passage vers les modèles dynamiques.

En ce qui concerne le MOD (Modèles Organisationnels de Données), on gagnera aussi en précision, en tirant partie des extensions déjà apportées au formalisme Entité-Relation dans les dernières évolutions de Merise, notamment en matière d'explicitation de contraintes inter et intra relations.

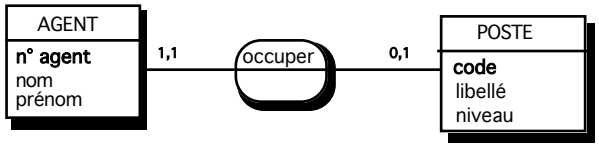
3.4.2. Élaboration des Modèles Objet

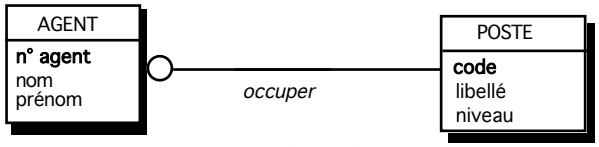
Les modèles objet permettent la modélisation structurelle du SII. Ils permettent la spécification des classes en indiquant leurs comportements. Ils seront dérivés du MCD - MOD (Modèle Conceptuel - Organisationnel de Données) dont ils hériteront de la structure et du MOT et des CVO dont ils hériteront d'aspects comportementaux intrinsèques aux données concernées par les tâches. Ils seront ensuite enrichis par des classes de nature plus logicielle.

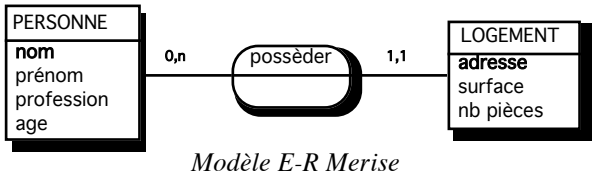
Pour la dérivation des Modèles Objet (MO), nous avons défini un certain nombre de règles de dérivation permettant de passer de modèles organisationnels Merise aux modèles objet OMT dans leur première formulation. Le MCD/MOD exprimé en formalisme Entité-Relation est considéré dans sa totalité. Les classes essentielles du Modèle Objet seront toutes issues des entités de ce MCD/MOD. Il s'agit d'une dérivation systématique suivant des règles définies dont les principales sont présentées dans les tableaux suivants.

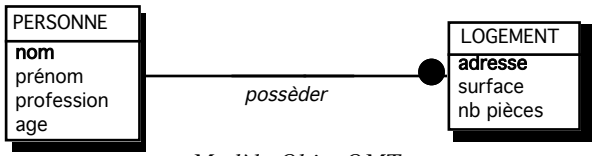
L'exploitation du MOT Merise permettra l'ajout de services aux classes du Modèle Objet OMT ou la création de nouvelles classes. On peut énoncer les principes généraux suivants :

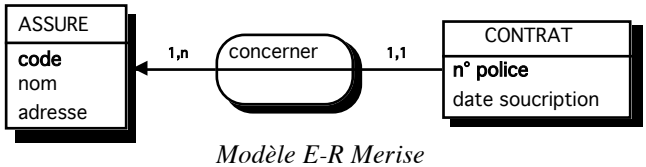
- à chaque acteur correspond un objet/une classe;
- les événements entre tâches et acteurs correspondent à des envois de messages paramétrés par des données associées aux événements
- chaque tâche contenant une seule ligne de description est équivalente à un service /méthode ayant un nom construit sur la désignation de la tâche
- les autres informations, condition de déclenchement, synchronisation, les règles d'émission des résultats sont traités dans les descriptions textuelles (templates) des services concernés des classes.

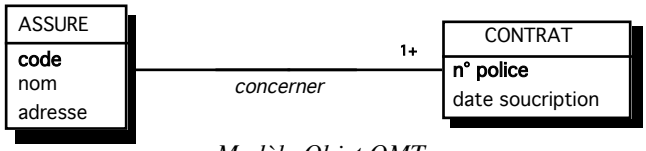
<table>
<tr><th>Modèle E-R Merise - modèle objet OMT</th><th>Commentaires</th></tr>
<tr><td>• les propriétés du modèle E-R deviennent des attributs dans OMT;
• l'identifiant de l'entité devient une clé candidate;
• les entités du modèle E-R deviennent des classes;</td><td></td></tr>
<tr><td>• cardinalités (1,1) - (0,1)
 

Modèle E-R Merise
 

Modèle Objet OMT</td><td></td></tr>
<tr><td>• cardinalités (0,n) - (1,1)
 

Modèle E-R Merise
 

Modèle Objet OMT</td><td></td></tr>
<tr><td>• cardinalités (1,n) - (1,1)
 

Modèle E-R Merise
 

Modèle Objet OMT</td><td></td></tr>
</table>

Tableau 1: *Règles de transformation de modèles E-R Merise en modèle objet OMT*

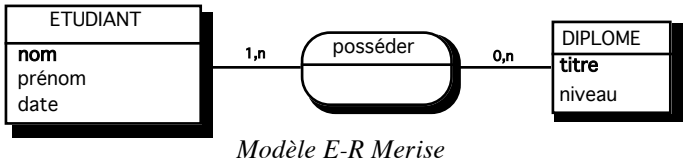
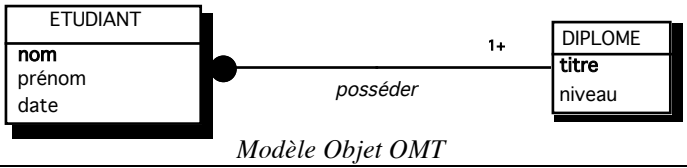
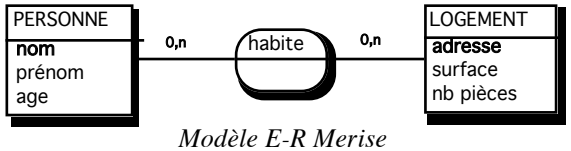
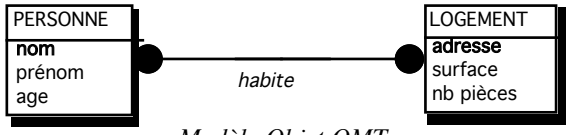
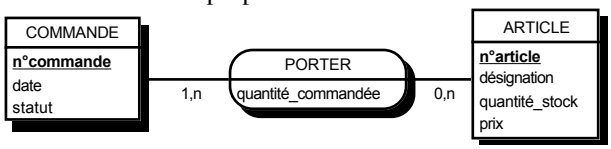
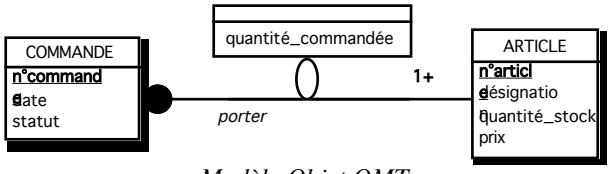
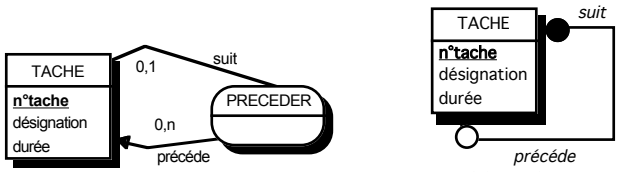
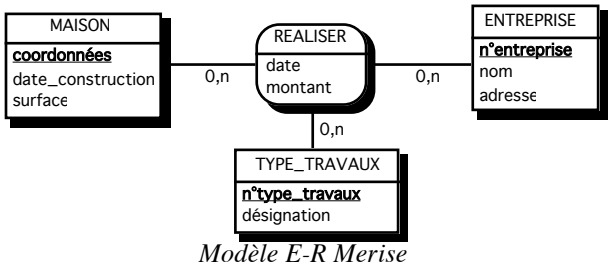
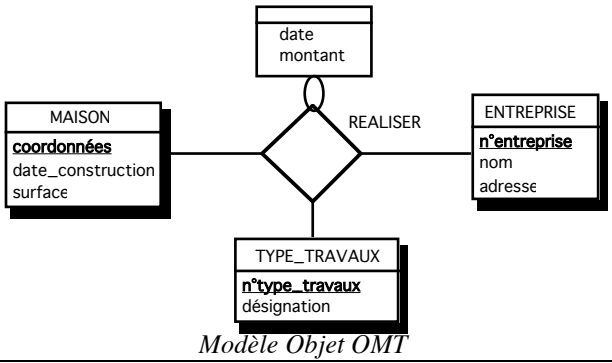
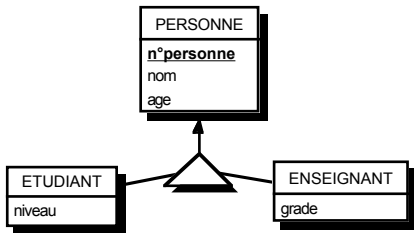
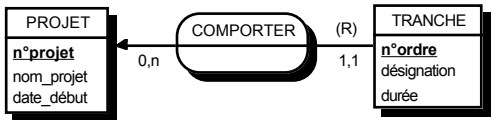
Modèle E-R Merise - modèle objet OMT	***Commentaires***
• cardinalités (1,n) - (0,n)  *Modèle E-R Merise*  *Modèle Objet OMT*	
• cardinalités (0,n) - (0,n)  *Modèle E-R Merise*  *Modèle Objet OMT*	
• relations binaires avec propriétés  *Modèle E-R Merise*  *Modèle Objet OMT*	
• relations réflexives  *Modèle E-R Merise* *Modèle Objet OMT*	

Tableau 2: *Règles de transformation de modèles E-R Merise en modèle objet OMT*

<table>
<tr><th>Modèle E-R Merise - modèle objet OMT</th><th>Commentaires</th></tr>
<tr><td>• relations n-aires avec propriétés
 

Modèle E-R Merise
 

Modèle Objet OMT</td><td>Dans OMT les multiplicités sur une association n-aire ne sont pas mentionnées.</td></tr>
<tr><td>• spécialisations
 

Modèle E-R Merise et Modèle Objet OMT</td><td>Les spécialisations sont représentées dans Merise et OMT avec le même formalisme.</td></tr>
<tr><td>• identifiants relatifs
 

Modèle E-R Merise
 

Modèle Objet OMT</td><td></td></tr>
</table>

Tableau 3: *Règles de transformation de modèles E-R Merise en modèle objet OMT*

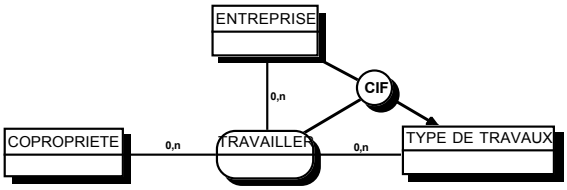
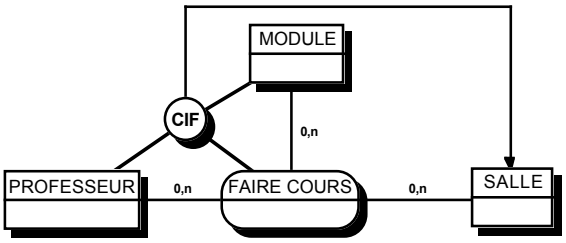
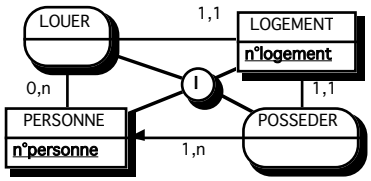
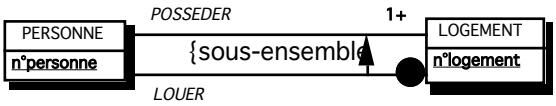
Modèle E-R Merise - modèle objet OMT	***Commentaires***
• Contraintes intra relations *a) dépendance fonctionnelle simple*  *Dans le cadre de la relation TRAVAILLER, chaque ENTREPRISE n'effectue qu'un TYPE DE TRAVAUX.* *Modèle E-R Merise* *b) dépendance fonctionnelle composée*  *Modèle E-R Merise*	*Les contraintes intra relations ne sont pas explicitement prises en compte dans OMT.* *Ces contraintes sont traitées dans OMT par la création de contraintes spécifiques de classes voire par des clé candidates partielles.*
• Contraintes inter relations  *Modèle E-R Merise*  *Modèle Objet OMT*	*Seules les contraintes inter relations de type inclusion sont explicitement prises en compte dans OMT.* *Ces contraintes sont alors traitées par la création de contraintes spécifiques de classes voire par des clé candidates partielles.*

Tableau 4: *Règles de transformation de modèles E-R Merise en modèle objet OMT*

3.4.3. Élaboration des modèles fonctionnels et dynamiques

L'élaboration des modèles fonctionnels et dynamiques reposera sur l'exploitation conjointe des modèles de données (MCD-MOD), les modèles organisationnels de traitements (MOT) et les cycles de vie des objets (CVO).

La modélisation du cycle de vie des objets (CVO) introduite dans la seconde génération de Merise vient judicieusement combler une lacune de la méthode et permet une meilleure articulation avec les méthodes orientées objets [Nanci, Espinasse et al. 96]. Nous avons vu que le cycle de vie des objets (CVO) et la modélisation organisationnelle des traitements (MOT) présentaient une certaine dualité et complémentarité.

Au travers du MOT on modélise, sous la forme d'une procédure organisationnelle, 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. L'événement et l'activité (tâche) y sont les concepts principaux, l'état est un concept secondaire. Avec le MOT, on s'intéresse ainsi essentiellement à l'analyse, à la répartition et aux ressources mises en oeuvre pour effectuer ces activités. Chaque schéma représente une procédure.

Au travers du CVO, on modélise les différents états possibles d'un même objet. Les événements et les transitions expriment les conditions de passage d'un état à un autre. L'état y est un concept principal, l'événement et l'activité (transition) sont des concepts secondaires. Par le CVO, 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.

Le modèle fonctionnel sera développé à partir des modèles organisationnel des traitements (MOT) détaillés, précisant notamment les états et les sous-schémas de données associés aux tâches de ces MOT. Il sera aussi nécessaire de consulter les cycles de vie des objets principaux (CVO) déjà élaborés.

Le modèle dynamique pour sa part sera développé à partir des cycles de vie des objets principaux (CVO) déjà élaborés. Il sera cependant nécessaire, par exemple pour des objets secondaires de se reporter aux modèles organisationnel des traitements (MOT) détaillés précisant au niveau de états et les sous-schémas de données.

3.5. Limites de cette voie

La première voie d'évolution de Merise vers l'objet que nous venons de présenter, nommée "transition vers l'objet" couplait Merise à la méthode objet OMT. Cette transition vers l'objet consistait à définir comment passer de la spécification d'un SIO réalisé selon le cadre méthodologique merisien 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.

Comme nous venons de le voir, cette transition vers l'objet ne pose pas de difficultés majeures. Pragmatique, opérationnelle à court terme, elle préserve une grande partie des investissements méthodologiques antérieurs en s'appuyant sur des bases méthodologiques acquises de Merise. Aussi nous apparaît elle aujourd'hui la voie la plus réaliste. Cependant cette voie présente un certain nombre de limites.

Tout d'abord, elle repose fortement sur la méthode OMT qui n'est pas encore stabilisée. En effet un premier rapprochement s'est déjà opéré avec la méthode OOD

[Booch 93] conduisant à la "méthode unifiée" [Booch & Rumbaugh 95]. Plus récemment un autre rapprochement de cette "méthode unifiée" est en cours avec la méthode Objectory-OOSE [Jacobson &al.94]. Notons que ce dernier rapprochement conduit plus précisément à la proposition d'un "langage unifié" de spécification pour le développement orienté objet [Booch & al.96]. Le principal intérêt de ce langage nous semble être l'intégration des cas d'utilisation déjà proposés par Jacobson dans OOSE. Ces cas d'utilisation devraient permettre à OMT, ou plutôt à cette nouvelle "méthode unifiée" en gestation, de ce placer plus en amont dans l'analyse et ainsi d'aborder le niveau du SIO. Malheureusement nous n'avons pas encore suffisamment de détails sur cette méthode unifiée intégrant l'apport de OOSE pour en évaluer les conséquences sur son éventuel couplage avec Merise.

Ensuite, le cadre de modélisation proposé par OMT ne nous apparaît pas toujours des plus satisfaisants pour aborder le SII avec une approche orientée objet. Ainsi, le modèle objet, modèle central d'OMT, est une extension à l'objet du modèle Entité-Association par adjonction d'opérations et de contraintes aux concepts d'entité et de relation. Il possède une puissance d'expression importante . Excepté la présence d'opérations et de contraintes, il est assez voisin du formalisme Entité-Relation proposé par Merise 2ème génération [Nanci, Espinasse et al. 96].

Certaines notions introduites dans le modèle objet OMT, nous apparaissent d'ailleurs plus avancées et plus clairement formalisées dans Merise; c'est notamment le cas de l'identification des relations, des cardinalités, des contraintes d'intégrité fonctionnelles, des contraintes inter-relations, de la nature des attributs pouvant être simple ou complexe, monovalué ou multivalué, de l'unicité de modélisation des relations qu'elles soient binaire, n-aires ou porteuses de propriétés. Nous invitons d'ailleurs le lecteur familier avec le formalisme Entité - Relation de Merise à découvrir la présentation du modèle de classes OMT faite dans [Rumbaugh &al.94]. Il constatera que les aspects considérés comme innovants par rapport au formalisme Entité-Association américain sont depuis longtemps présents dans Merise. Il pourra également sourire de certaines remarques et conseils sur la modélisation des associations (nom des associations, associations réflexives, associations ternaires, multiplicité, clés candidates, ...) que les praticiens de Merise connaissent depuis plus de dix ans.

Nous ne développerons pas ici les remarques liées aux modèles dynamiques et fonctionnels présentant un certain nombres de faiblesses ou imprécisions tant dans leurs expressions que dans leurs liens entre eux et avec le modèle objet. Enfin, de nombreuses lacunes subsistent encore dans la démarche itérative d'élaboration de ces modèles [Jacques &al.96].

Ainsi, OMT et les méthodes orientées objet en général, nous apparaissent dans une première phase de maturité, dans laquelle elles ont à stabiliser, épurer et préciser diverses notions et principes qu'elles proposent mais aussi se préoccuper du contrôle et de la validation des modèles. Il leur faut aussi acquérir une expérience de terrain sur des projets du monde réel dans des conditions normales, ce qui permettra de les éprouver et les fiabiliser. Cet état de fait engendre de nombreuses difficultés à la mise en oeuvre de cette première voie, leur couplage avec Merise.

4. Évolution vers l'objet : vers une troisième génération de Merise

Au delà de ces remarques sur le cadre de modélisation d'OMT, il est certain que tout couplage entre deux méthodes n'est jamais pleinement satisfaisant, même si les missions de chacune sont relativement circonscrites, comme c'est le cas ici entre Merise et OMT. Ceci tient principalement au fait que les cadres de modélisation

ainsi que les démarches des deux méthodes ne sont que partiellement compatibles. Ceci nous amène à introduire quelques réflexions sur la seconde voie d'évolution de Merise vers l'objet que nous associeront à l'émergence d'une troisième génération de Merise.

Comme nous l'avons déjà évoqué, cette seconde voie, plus radicale, 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 au travers de l'approche objet. La réutilisation comme la continuité méthodologique reposerait dans cette voie sur l'introduction d'une typologie du concept d'objet. En amont on trouverait un concept d'objet couvrant le niveau SIO que l'on pourrait nommer "objet métier". En aval, on retrouverait des concepts d'objet relevant du génie logiciel associé au SII, "objet logiciel" ou "objet technique". Une dérivation des "objets métiers" en "objets logiciels" devant bien évidement être possible et clairement définie.

Dans cette nouvelle voie, bien des aspects sont actuellement imprécis et relèvent encore de la recherche. De nombreux travaux vont en ce sens et déjà plusieurs méthodes de conception objet de SI, la plupart encore au stade de développement, sont proposées. Parmis ces méthodes, une des rares à se prévaloir de l'héritage merisien, et vraisemblablement une des plus avancée, est la méthode OOM (Orientation Objet dans Merise) proposée par Rochfeld et Bouzeghoub [Rochfeld &al.93] [Bouzeghoub &al.94]. Dans cet article, nous ne développerons pas cette méthode et nous renvoyons le lecteur sur des présentations spécifiques.

Comme nous l'avons déjà évoqué, l'approche objet nous semble présenter deux intérêts principaux pour la conception des systèmes d'information : la réutilisation d'objets et la continuité méthodologique (démarche et formalismes) centrées sur la notion d'objet. Nous nous limitons ici à développer une réflexion sur ces deux principes ainsi que sur une typologie du concept d'objet sur laquelle ils pourraient reposer. Ces éléments de reflexion nous apparaîssent indispensables au développement d'une troisième génération de la méthode Merise .

4.1. De la réutilisation en ingénierie des SI

La réutilisation de composant logiciels permet d'améliorer la productivité et la réactivité de l'entreprise en matière de développement d'applications informatiques.

Pour illustrer l'intérêt de la réutilisation pour l'entreprise, considérons le cas de l'évolution d'une architecture client-serveur. Une telle architecture est caractérisée par des parties «client» coopérant avec des parties «serveur». Supposons que l'entreprise doive faire évoluer cette architecture client-serveur en redéfinissant ces parties client et serveur par exemple pour des raisons de performance. Cette évolution pourra conduire à implémenter une même application dans des architectures différentes. Une telle flexibilité n'est possible que s'il existe une organisation du logiciel où des modules réutilisables se comportent eux-mêmes en client-serveurs les uns vis à vis des autres et peuvent être distribués selon les besoins entre les machines interconnectées s'échangeant des services. Dès à présent, certains SGBD (Oracle notamment) permettent de réaliser de façon aisée une telle redistribution.

D'une façon générale, on peut distinguer deux grands types de réutilisation :

- La *réutilisation par intégration* consiste à associer, au sein d'un même système, des pièces qui ont des origines différentes et qui n'ont pas été conçues à cette fin. Il faut pour cela examiner l'intérieur de chacune des pièces pour identifier les modifications qu'il faudrait leur apporter.

- La *réutilisation par assemblage* consiste elle à réunir des pièces ensemble sans avoir à les modifier. Il faut pour cela que les pièces que l'on veut assembler aient été conçues pour cette fin.

Dans ces deux types de réutilisation, la modélisation des composants s'avère indispensable soit au moment de leur intégration soit au moment de leur conception pour un assemblage futur. Dans cette modélisation, l'approche objet s'avère extrêmement efficace en mettant en avant des nouveaux principes permettant la réutilisabilité des composants logiciels.

La réutilisation, déjà extrêmement efficace en génie logiciel, pourrait l'être également au niveau de la conception de systèmes d'information à savoir au niveau du SIO (réutilisation conceptuelle et organisationnelle). Une telle réutilisation permettrait, vraisemblablement à moindre coût, de concevoir et faire évoluer efficacement les systèmes d'information des entreprises.

Elle devrait de ce fait apporter une réponse efficace aux besoins constants de réorganisation de l'entreprise en rendant encore plus opératoire une approche BPR, en permettant le recours à des composants "métier". On peut ainsi imaginer une reconception de la gestion et de l'organisation basée sur l'utilisation "d'éléments préfabriqués". Notons que de nombreux praticiens, en particulier des consultants, pratiquent depuis longtemps et de façon informelle, la réutilisation conceptuelle et organisationnelle en réutilisant des modélisations "standard" qu'ils se sont bien souvent eux mêmes construit au fur et à mesure des projets.

4.2. De la continuité méthodologique en ingénierie des SI

La continuité méthodologique centrée sur le concept d'objet est très séduisante mais délicate à réaliser. En effet, en ingénierie de systèmes d'information, nous avons à concevoir deux systèmes distincts, le SIO et le SII, en s'appuyant sur un même concept, celui d'objet. Ainsi, le concept d'objet défini au niveau du SIO. devrait être en continuité sémantique avec celui déjà défini en génie logiciel au niveau du SII. Cette hypothèse est loin d'être confirmée et mérite quelques commentaires.

Dans une démarche d'ingénierie de systèmes d'information, l'un des premiers travaux abordé est celui de la modélisation du SIO Cette modélisation s'effectue en se concentrant sur les aspects managériaux (informations utilisées, activités exercées, règles appliquées, organisation adoptée, ressources mises en jeu), en faisant provisoirement abstraction des aspects strictement informatiques. Notons que les niveaux conceptuels et organisationnels et leurs modèles associés définis dans Merise s'avèrent, en cette fin des années 90 d'une pertinence extrême pour la prise en compte, la remise en cause et l'évolution des modes de gestion et d'organisation de l'entreprise [Nanci&Espinasse al. 96].

L'approche objet commence à être envisagée pour aborder l'entreprise à des niveaux managérial et organisationnel associés au SIO. Citons notamment les travaux de Jacobson [Jacobson &al.94] ainsi que ceux de Yourdon [Yourdon &al.95]. Dans cette spécification du SIO, les objets à prendre en compte sont d'une nature très différente de ceux qui se retrouveront dans le SII. Dans la spécification du SII, les objets mis en oeuvre ont une existence essentiellement liée à une solution logicielle; ils ne résultent probablement pas d'un choix organisationnel ou managérial. Il est toutefois probable que certains des objets logiciels puissent avoir une filiation avec ceux retenus dans une modélisation objet du SIO; mais la mise en évidence et la justification de ces objets du SIO procède d'autres critères et démarches que ceux associés aux objets du SII.

Ainsi, si nous convenons que la recherche d'une continuité méthodologique est un objectif pertinent, il nous faut cependant être extrêmement prudent. Nous ne pouvons proposer une alternative objet à Merise que si celle-ci est parfaitement adaptée à la mission ambitieuse que s'est fixée cette méthode, à savoir la conception de systèmes d'information supportant l'activité d'organisations socio-économiques.

La continuité méthodologique qui nous préoccupe ici est celle qui permettrait de passer d'une spécification du SIO à une spécification du SII sans rupture méthodologique. En fait cette continuité ainsi assurée n'est que partielle, les problèmes liés à chacune des conceptions étant différents et nécessitant des univers du discours spécifiques auxquels sont associés des niveaux de préoccupation ou d'abstraction.

On peut distinguer deux niveaux de continuité méthodologique :

- La *continuité de concept* consiste à pouvoir, au travers des mêmes concepts, développer des spécifications de niveaux d'abstraction différents. Par exemple, le concept d'objet Contrat serait utilisé à l'identique dans une spécification du SIO puis dans une spécification du SII.
- La *continuité de formalisme* consiste à utiliser un même formalisme de modélisation pour développer des spécifications de niveaux d'abstraction différents. Par exemple, on modélise sous la forme d'objet de graphismes semblables un Contrat au niveau du SIO et une fenêtre F1 au niveau du SII.

Tant pour la continuité de concept que pour la continuité de formalisme le processus de dérivation d'une spécification d'un niveau d'abstraction à une autre de niveau plus bas doit être précisé.

La continuité de formalisme est très séduisante car elle permet une économie de formalisme et peut en partie simplifier le processus de dérivation. La principale critique que l'on peut faire aux méthodologues de l'objet actuels consiste à vouloir réutiliser des formalismes utilisés dans la spécification de bas niveau, la cible (objets logiciels des langages de programmation) pour élaborer des spécifications de plus haut niveau associées au besoin (besoins d'utilisateurs par exemple). Ainsi bien que séduisante, cette continuité de formalisme nous apparaît encore très ambitieuse.

La continuité de concept nous apparaît bien plus urgente à appréhender que la continuité de formalisme. Cette continuité de concept repose nécessairement sur la définition d'une typologie d'objet permettant de différencier la nature et la granularité des objets que l'on utilisera dans les différentes spécifications du système d'information depuis l'expression des besoins relevant du SIO jusqu'au codage associé au SII. Cette typologie d'objets nous permettra aussi d'appréhender le processus de dérivation et d'assurer la traçabilité «verticale» entre ces divers types d'objets.

4.3. Des objets métiers aux objets logiciels

La réutilisation comme la continuité méthodologique repose sur l'introduction du concept "objet métier" au niveau SIO et sa dérivation en "objet logiciel" au niveau du SII.

4.3.1. Les Objets métiers

Bien que que le terme "d'objet métier" commence à être utilisé dans la littérature relative tant au génie logiciel qu'au management, il n'existe pas vraiment à notre connaissance de concensus sur une définition. Le cadre de modélisation proposé par Merise nous permet d'en proposer une à la fois acceptable d'un point de vue génie logiciel et d'un point de vue managérial.

Ainsi, les objets dits "métiers" sont des objets perçus au niveau du SIO, ils relèvent, en se référant au cadre de modélisation de Merise, soit du niveau conceptuel (objets de gestion) soit du niveau organisationnel (objets organisationnels). Ces objets permettent de décrire les informations structurées, les comportements managériaux et organisationnels associés à l'activité d'un domaine.

Les objets métiers conceptuels

Les *objets métiers conceptuels* sont des objets du monde managérial. Ils décrivent les information, les connaissances et les activités que l'organisation a choisi de gérer, voire de capitaliser. Ces objets métiers conceptuels encapsulent des services précisant les comportement managériaux ou règles de gestion que l'organisation a adopté. Notons que ces comportements managériaux sont principalement associés à un fonctionnement dit "normal" de l'organisation mais ils peuvent aussi concerner des situations sortant de la normalité tout en restant prévisibles. La considération de tels comportements permet de rendre l'organisation plus réactive.

Cette notion d'objet métier conceptuel permet de réunir sous un même concept des modélisations qui aujourd'hui dans la méthode Merise sont exprimées sous différentes formes. Par exemple, le concept de Commande peut être modélisé comme une entité dans un MCD-MOD, comme un processus dans un macro MCT, comme un événement dans un MCT, comme un modèle externe associé à un sous-schéma de données dans un MOT.

L'objet métier conceptuel devient ainsi plus unifié et plus proche de la perception globale - bien que multiforme - qu'en a le gestionnaire. Il se situe ainsi à un niveau d'abstraction et de synthèse plus élevé que les modélisations qualifiées de conceptuelles dans Merise qui son déjà une vision analysée par un concepteur de systèmes d'information.

Les objets métiers organisationnels

Les *objets métiers organisationnels* sont des objets, issus des perceptions de l'utilisateur, analysés et modélisés par un concepteur en vue de la conception d'un système d'information. L'utilisateur pourra ultérieurement consulter, modifier et créer ces objets par exemple l'objet client, l'objet commande, l'objet facture. Ces objets métiers encapsulent des services qu'ils doivent pouvoir fournir, par exemple l'objet client doit pouvoir donner son nom et son âge, l'objet commande doit pouvoir fournir entre autre le détail de ces lignes de commande et pouvoir se facturer.

Ces objets métiers organisationnels permettent aussi dans des modèles comportementaux (fonctionnels et dynamiques), de décrire une partie de l'activité de

l'utilisateur sollicité par un événement extérieur. Par exemple l'activité d'un vendeur à la commande d'un client pour des articles spécifiques et dans des conditions (par exemple de facturation et de livraison) tout aussi spécifiques. Ils permettent de décrire l'organisation (la répartition) du travail entre les postes de travail et au sein d'un même poste de travail notamment la répartition du travail entre homme-système ainsi que la logique de dialogue. Ces modèles comportementaux devraient s'inspirer à la fois des MOT déjà proposés par Merise et des cas d'utilisation introduits par Jacobson dans OOSE et qui sembleraient être adopter par la plupart des nouvelles méthodes objets comme la méthode unifiée.

Liens entre objets métiers conceptuels et organisationnels

Pour comprendre le lien entre les objets métiers de niveau conceptuel et ceux de niveau organisationnel considérons la courbe du soleil illustrant la démarche préconisée par Merise qui s'avère pertinente dans le BPR [Nanci&Espinasse al. 96]. Selon que l'on se place dans une approche prototypale de développement rapide (RAD) ou dans une approche classique voire participative, les objets métiers n'auront pas la même importance.

Dans le cas du RAD, les objets métiers organisationnels sont très importants et influencent fortement les objets métiers conceptuels au point de les assimiler. Cette assimilation n'est possible que pour des systèmes d'information extrèmement circonscrits à quelques utilisateurs. Dans le cas de systèmes d'information plus larges dans lesquels de nombreux utilisateurs sont impliqués et pour lesquels on adopte plutôt une démarche classique, les objets de métiers conceptuels permettent une synthèse plus abstraite des nombreux objets métiers organisationnels. Dans ce cas, ces objets métiers conceptuels sont développés en s'appuyant sur les objets métiers organisationnels au travers de processus de synthèse et d'abstraction. Les objets métiers organisationnels se présentent alors comme des recompositions, des vues partielles de ces objets métiers conceptuels.

C'est au niveau de ces objets métiers tant conceptuels qu'organisationnels que pourront être spécifiées des orientations de type BPR ou Workflow. Ces objets doivent permettre aussi le développement de bibliothèque d'objets métiers génériques pouvant être combinés et auxquels seraient associés des "briques logicielles" spécialisées, pouvant être assemblées et paramétrées pour développer rapidement des systèmes d'information opérationnels par réutilisation.

4.3.2. Les objets logiciels

Les objets logiciels sont des objets permettant la spécification du SII. Ils composent l'architecture de composants logiciels du SII et ont une existence essentiellement liée à une solution logicielle; ils ne résultent probablement pas d'un choix organisationnel ou managérial. Certains des objets logiciels ont une filiation avec les objets métiers retenus dans la modélisation objet du SIO.

Ces objets logiciels permettent la spécification des différents modules composant les applications du SII, notamment ceux permettant de présenter des informations structurées sur une interface utilisateur, d'interpréter les actions demandées par un utilisateur sur cette interface, de mettre en oeuvre les règles de gestion attendues, et de consulter ou mettre à jour une ou plusieurs bases de données.

Ces objets logiciels sont de différents types selon leur rôle dans la structuration de l'application, de son interface utilisateur jusqu'aux données stockées qu'elle concerne. Pour chaque type d'objet logiciel, une description et un comportement spécifiques seront définis.

On peut distinguer plusieurs grands types d'objets logiciels pouvant composer une application quelconque : les objets de type mémoire, de type usage et de type interaction.

Les objets logiciels mémoire

Ces objets sont esssentiels car ils assurent le lien entre l'application et les données mémorisées. En fait, ces objets sont les garants de la persistance des informations, voire des connaissances mémorisées.

Pour assurer cette persistance, deux types d'objets sont nécessaires :

- les objets *sémantiques* : ils encapsulent un ensemble d'information concerné par l'application ainsi que les services associés assurant la cohérence des actions pouvant être réalisées sur ces informations. Les objets sémantiques sont principalement dérivés des objets métiers conceptuels et organisationnels, et constituent ainsi des implémentations logiques partielles de ces derniers. Les informations étant stockées la plupart du temps dans une ou plusieurs base(s) de données, ces objets simulent un sous ensemble de ces bases en s'assurant de son intégrité c'est à dire en respectant les contraintes d'intégrités associées.
- les objets *d'accés* : ils ont pour principale fonction d'assurer la construction des objets sémantiques à partir des informations gérées dans la ou les base(s) de données. Ces objets réalisent notamment toutes les requêtes SQL nécessaires pour extraire les données de la base, ils permettrent ainsi la construction des objets sémantiques précedents. Ces objets d'accés assurent ainsi le lien entre les schèmas relationnels des bases de données avec leurs contraintes d'intégrités associées et les objets sémantiques. Par exemple, on définira un objet sémantique "Commande" qui sera construit par un ou plusieurs objets d'accés à partir de tables "Commande" et "Ligne de Commande" d'une base de données relationnelle, ceci grâce à plusieurs requêtes SQL.

Les objets logiciels d'usage

Ces objets permettent, à partir des objets sémantiques, de reconstruire les objets métiers organisationnels perçus par les utilisateurs et leurs services associés. Les objets d'usage sont ainsi des "vues" des objets sémantiques.

Par exemple dans une entreprise d'assurance, les agents commerciaux et les juristes invoquent tous les deux un objet d'usage nommé "Contrat". Lors d'une utilisation particulière du SII, les agents commerciaux utilisent à travers les fenêtres des applications qui leur sont destinées, une vue du "contrat" mettant en évidence ses produits et services pertinents pour leur fonction. De même, les juristes, au travers d'application qui leur sont destinées, utilisent une autre vue du "contrat" mettant en évidence des aspects juridiques à savoir des responsabilités et des garanties. Chacune de ces vues peut être considéré comme un objet d'usage, objet composé à partir d'un ou plusieurs objets sémantiques et encapsulant des méthodes spécifiques.

L'objet d'usage évite ainsi de disséminer les règles de construction d'une vue et les services associés entre plusieurs objets sémantiques, ce qui préserve leur interchangeabilité.

Les objets logiciels d'interaction

Ces objets assurent les interactions utilisateur-système. Pour assurer cette interaction trois types d'objets sont nécessaires : les objets d'interface, les objets d'intéraction et les objets processeur de dialogue :

- les objets *d'interface* assurent dans l'application la présentation à l'utilisateur des informations qu'il saisit ou demande, ainsi que l'identification des actions qu'il veut déclencher. Notons que la spécification de ces objets est fortement liée à la technologie informatique retenue pour développer l'interface.
- les objets *d'interaction* interprétent et contrôlant les messages reçus par un objet d'interface et gère les interactions entre celui-ci et le système. Il a pour rôle de dissocier de l'objet d'interface les comportements qui ne sont pas contraints par la technologie de celui-ci.
- les objets *processeur de dialogue* jouent un rôle comparable à un moniteur d'enchaînement d'écrans. En fait, il pilote l'activation des objets d'interaction qui participent à une utilisation spécifique du système par l'utilisateur et gère le contexte global de cette utilisation.

La figure suivante illustre cette typologie des objets métiers et logiciels.

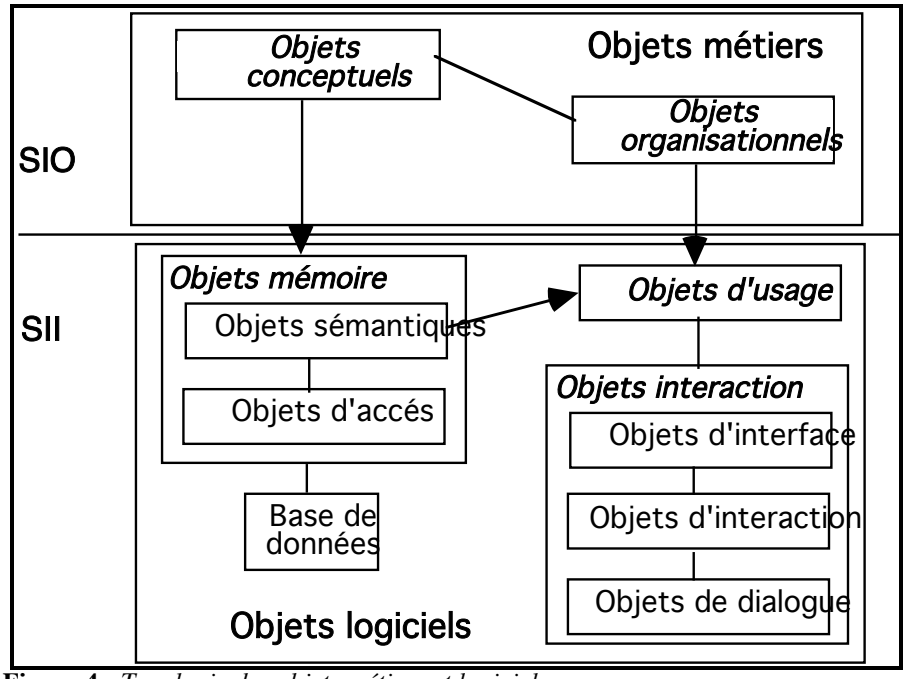


Figure 4 : *Typologie des objets métiers et logiciels.*

5. Conclusion

L'évolution générale de la gestion des entreprises et en particulier des systèmes d'information se caractérise par un accroissement de la complexite du à la multiplication des interactions, et une répartition des activités favorisé par le développement des réseaux et de l'informatique distribuée.

Déjà largement utilisée en génie logiciel, l'approche objet offre une manière claire de concevoir une architecture modulaire, encapsulant la complexité et facilitant l'implémentation multi-plateformes. L'approche objet présente deux intérêts principaux pour la conception des systèmes d'information : la réutilisation d'objets et la continuité méthodologique (démarche et formalismes) centrées sur la

notion d'objet. Elle permet en outre une flexibilité technique accrue et une meilleure ouverture aux nouvelles technologies telles que le client-serveur, le multimédia et s'avère ainsi bien adaptée à la conception et la réalisation de ces systèmes d'information distribués. 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 et touche de façon profonde l'informatique de gestion.

La première voie d'évolution de Merise vers l'objet que nous proposons, nommée "transition vers l'objet" couple Merise à la méthode objet OMT. Cette transition vers l'objet consiste à définir comment passer de la spécification d'un SIO réalisé selon le cadre méthodologique merisien 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. Pragmatique, opérationnelle à court terme, cette voie préserve une grande partie des investissements méthodologiques antérieurs en s'appuyant sur des bases méthodologiques acquises de Merise. Aussi nous apparaît elle aujourd'hui la voie la plus réaliste. Cependant cette voie présente un certain nombre de limites principalement liées au fait que les cadres de modélisation ainsi que les démarches des deux méthodes ne sont que partiellement compatibles.

La seconde voie d'évolution de Merise vers l'objet que nous proposons consiste en une remise en cause profonde de Merise, permettant de traiter complètement la réutilisation et la continuité méthodologique entre le SIO et le SII au travers de l'approche objet. Bien des aspects y demeurent imprécis et relèvent encore de la recherche. Aussi, nous sommes nous limité à développer une réflexion sur ces deux principes ainsi que sur une typologie du concept d'objet. En amont on trouverait un concept d'objet couvrant le niveau SIO que l'on pourrait nommer "objet métier". En aval, on retrouverait des concepts d'objet relevant du génie logiciel associé au SII, "objet logiciel" ou "objet technique". Une dérivation des "objets métiers" en "objets logiciels" devant bien évidement être possible et clairement définie. De nombreux travaux vont en ce sens et déjà plusieurs méthodes de conception objet de SI, la plupart encore au stade de développement, sont proposées [Rochfeld &al.93] [Bouzeghoub &al.94].

Pour conclure, il nous apparaît indispensable de menner plus a fond cette reflexion (à peine esquissée ici) relative à la modélisation du système d'information organisationnel et/ou informatisé, tant d'un point de vue génie logiciel que managérial sur laquelle pourrait s'appuyer une véritable troisième génération de Merise.

6. Bibliographie

[Boehm 81], Boehm B., *Software Engineering Economics*, Prentice-Hall, 1981.

[Booch & al.95], Booch G., Rumbaugh J., *Unified Method,* version 0.8, Rational Software Corporation, 1995.

[Booch & al.96], Booch G., Jacobson I., Rumbaugh J., *The Unified Modeling Language for Object Oriented Development,* version 0.91, Rational Software Corporation, 1996.

[Booch 93], Booch G., *Object-Oriented Analysis and Design with Applications* , second edition, The Benjamin Cummings Publishing Company, 1993.

[Bouzeghoub &al.94], Bouzeghoub M., Gardarin G., Valduriez P., *Du C++ à Merise objet: Objets*, Eyrolles, 1994.

[Cauvet &al.91], 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.

[Coad & Yourdon 92], Coad P., Yourdon Y., traduction française de A. Boughlan, *Analyse Orientée Objets*, Masson, 1992.

[Coleman &al.94], Coleman D., Arnold P., Bodoff S., Dollin C., Gilchrist H., Hayes J., Jeremaes P., *Fusion : la méthode orientée objet de 2e génération*, Masson, 1996.

[Espinasse &al.93], 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.

[Foucault &al.96], Foucaut O., Thiery O., Smaïl K., *Conception de systèmes d'information et programmation événementielle*, InterEditions, 1996.

[HOOD 92], HOOD *Reference Manual*, version 3.1.1, HOOD User Group, june 92.

[Jacobson &al.92], Jacobson I., Christerson M., Jonsson P., Övergaard G., *Object-Orented Software Engineering - A use Case Driven Approach*, Reading MA: Addison-Wesley, 1992.

[Jacobson &al.94], Jacobson I., Ericsson M., Jacobson A., *The object advantage, Business Process Reengineering with object technology*, ACM Press Book, Addison-Wesley, 1994-1995.

[Jacques &al.96], Jacques P., Mercier B., Mollard D., Nguyen V.H., "De Merise à l'approche objet", *Rapport d'étude, CNAM-Informatique d'entreprise*, 1996.

[Morin 1977], Morin E., *La méthode, tome 1,* Editions du Seuil, 1977.

[Nanci, Espinasse et al. 92], Nanci D., Espinasse B., Cohen B., Hechenroth H., *Ingénierie des systèmes d'information avec la méthode MERISE : vers une deuxième génération*, Éditions Sybex, mai 1992, 650 pages, 2° édition.

[Nanci, Espinasse et al. 96], Nanci D., Espinasse B., Cohen B., Hechenroth H., Asselborn J-C., *Ingénierie des Systèmes d'Information: Merise Deuxième Génération* - 3° Edition entièrement revue et augmentée, Éditions Sybex, 1996.

[Pham 85] Pham Thu Quang, *Merise et Yourdon, in Génie logiciel et systèmes experts,* n° 15, septembre 1989, pages 22 à 39, publication EC2, 1989.

[Rochfeld &al.93], Rochfeld A., Bouzeghoub M., *From MERISE to OOM*, in Ingénierie des systèmes d'information, vo.1 N°2, pp 151-176, 1993.

[Rolland &al 87], Rolland C., Foucaut O., Benci G., *Conception des systèmes d'information, la méthode Remora*, Eyrolles, 1987.

[Rumbaugh & al. 91]Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

[Shlaer & Mellor 90] Shlaer S., Mellor S.J., *Object life cycles: Modeling the world in states*, Prentice Hall, 1990.

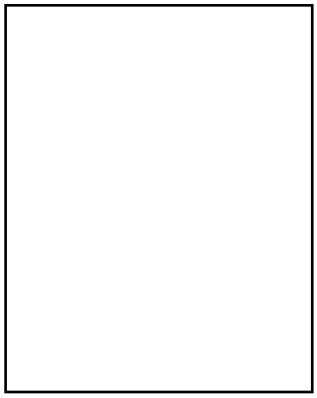
[Tardieu &al.83], Tardieu H., Rochfeld A., Coletti R., *La méthode Merise,* tome 1 : *Principes et outils,* éditions d'Organisation, 1983.

[Tardieu &al.85], Tardieu H., Rochfeld A., Coletti R., Panet G., Vahee G., *la Méthode Merise,* tome 2 *: Démarche et pratiques,* éditions d'Organisation, 1985.

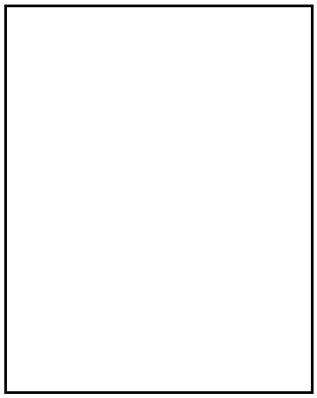
[Yourdon &al.95] Yourdon Ed., Whitehead K., Thomann J., Oppel K., Nevermann P., *Main Stream Object : an Analysis and Design Approach for Business*, Yourdon Press, Prentice Hall, 1995.

———————

Biographie



Bernard ESPINASSE *est Professeur à l'Université d'Aix-Marseille. Ingénieur Arts et Métiers, Docteur en Systèmes d'Information et diplômé de l'IAE d'Aix-en-Provence, il rejoint en 1979 l'équipe "aixoise" et participe aux recherches sur la méthode Merise. Depuis 1987 et après trois années d'enseignement à l'Université Laval (CANADA), il partage son temps entre la recherche, l'industrie et l'enseignement dans divers programmes de deuxième et troisième cycle spécialisés en systèmes d'information (Ecoles d'ingénieurs, MIAGE et Mastères Spécialisés). Il est l'auteur ou le co-auteur d'ouvrages, d'articles et de communications dans le domaine de l'ingénierie des bases de données et des systèmes d'information et d'aide à la décision.*



Dominique NANCI *est Directeur Technique de CECIMA, société de conseil en ingénierie de systèmes d'information, spécialisée dans la méthode Merise et éditrice du logiciel Win'Design. Ingénieur INSA Lyon, diplômé de l'IAE, Docteur en systèmes d'information. De 1973 à 1978, enseignant-chercheur à l'IAE d'Aix, il participe, au sein de l'équipe animée par H.Tardieu, aux travaux de recherche initiaux sur la méthode Merise. En une quinzaine d'années de pratique de cette méthode, il a capitalisé une expérience de tout premier plan dans la mise en oeuvre de Merise. Son pôle d'intérêt reste centré sur l'évolution des méthodes de conception de systèmes d'information et les outils associés. Il est enfin co-auteur de deux ouvrages et de plusieurs articles sur la méthode Merise.*