Le projet MERISE+ :

Une extension de la méthode MERISE à l'approche objet par un apport de la méthode HOOD

Bernard Espinasse * **- Michel Lai** ** **- Dominique Nanci** ***

** Institut SYSTEMIA, Domaine Saint-Hilaire-Pichaury, BP 223 000, 13796 Aix-en-Provence, Cedex 3, France et DIAM-LGIM, Université Aix-Marseille III, Domaine Universitaire de St Jérôme, Avenue Normandie Niemen, 13397 Marseille Cedex 13, France - Email : espina@frmop11.bitnet.*

*** INGENIA, Technôpole de Château Gombert, Europarc, Bat. C, 13013 Marseille, France - Email : Lai.@ingenia.fr*

**** CECIMA, 24 avenue de la Molle, 13100 Aix-en-Provence, France.*

RESUME. Depuis son origine la méthode MERISE a connu des développements et des enrichissements profitables, mais il n'en reste pas moins qu'elle présente actuellement encore un certain nombre de lacunes, la principale concernant le passage de la conception à la réalisation. Pour combler cette lacune liée à l'accroissement de la complexité des logiciels à réaliser, le projet MERISE+ se propose de rapprocher MERISE de la méthode orientée objet HOOD utilisée pour la conception de gros logiciels dans les domaines scientifique et technique. Dans un premier temps on positionne les deux méthodes en génie logiciel, ensuite on les compare selon des axes et critères jugés pertinents. Cette comparaison nous permet ensuite de définir les complémentarités de ces deux méthodes. Enfin, nous présentons les orientations retenues pour leur articulation dans le cadre du projet MERISE +.

ABSTRACT. Since its origin, the MERISE method has known some useful developments and extensions. But today it still presents a certain number of deficiencies, the principal one concerning the passage from the design to the realisation. In order to fill this gap associated to the complexity increase of the software to realise, the MERISE+ project attempts to bring together MERISE and HOOD, an object oriented software engineering method used to design big software in scientific and technical fields. In a first time we define the position of the two methods in the software engineering, then we compare these methods according to some specific axes and criterions. This comparison permits us to define the complementarities of these two methods. Finally we present the main orientations adopted for their coupling in the MERISE+ Project.

MOTS-CLES: méthode de conception, système d'information, génie logiciel, MERISE, Hood, orienté objet, objet.

KEYWORDS : design method, management information system, software engineering, MERISE, Hood, object oriented, object.

Le projet MERISE+ est partiellement subventionnée par une aide du Ministère de l'Industrie dans le cadre du programme Informatique 92.

1. Introduction

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

Dans le processus de conception et de développement des systèmes d'information, les spécialistes s'accordent aujourd'hui à reconnaître deux niveaux majeurs de préoccupation :

1 - la conception du S.I.O. (Système d'Information Organisationnel), également intitulé ingénierie des "besoins" ou "users requirements", correspondant aux préoccupations des utilisateurs,

2 - la conception du S.I.I. (Système d'Information Informatisé), également intitulé ingénierie du logiciel ou "software development" correspondant aux préoccupations des informaticiens.

La double utilisation du terme "conception" entretien malheureusement une ambiguïté préjudiciable à la compréhension mutuelle. En fait, l'objet à concevoir est différent à chaque niveau. Le concepteur de S.I.O définit puis mettra en service le système d'information adapté aux activités d'une entreprise (les besoins); le concepteur de S.I.I. définit et réalise un logiciel, support de ce système d'information (le produit).

Bien qu'ambitionnant à couvrir l'ensemble des niveaux de préoccupation, la plupart des méthodes ne se positionnent explicitement ou implicitement, que sur l'un des niveaux. Ainsi la méthode MERISE se focalise essentiellement sur la conception de S.I.O., tandis que la méthode HOOD se préoccupe exclusivement de la conception de S.I.I.

Afin d'apporter aux praticiens de la conception et du développement de systèmes d'information à informatiser, un ensemble méthodologique complet couvrant les deux niveaux majeurs et leur permettant de faire face à la complexité des logiciels à réaliser ainsi qu'aux nouvelles technologies, le projet MERISE + se propose de combler les insuffisances de la méthode MERISE au niveau du SII par un enrichissement avec la méthode orientée objet HOOD. Le résultat sera accompagné par un outil logiciel réalisé par l'extension de l'AGL de conception MESSAGE [Message 87-93] de la société CECIMA.

L'objectif de cette communication est de présenter l'intérêt de cette "greffe". Après avoir rappelé l'origine, le positionnement et les caractères principaux de chaque méthode, on comparera les deux méthodes afin de mettre en évidence les complémentarités. Enfin, nous présenterons les orientations retenues pour l'articulation des méthodes MERISE et HOOD dans le cadre du projet MERISE +.

2. MERISE et l'ingénierie des systèmes d'information

2.1. *Origine de MERISE*

La méthode MERISE a été concrétisée entre 1977 et 1979, avec la collaboration des principales sociétés de service et du CETE d'Aix-en-Provence, sous l'égide du Ministère de l'Industrie. [CTI79]. Depuis une dizaine d'années, la diffusion de cette méthode est un succès, tant en France où elle fait office de standard que dans une partie de l'Europe, où elle est adoptée avec parfois quelques aménagements mineurs. Ce succès s'est accompagné par des développements et enrichissements profitables :

- extension du formalisme entité-relation avec notamment l'introduction des sous-typages, des contraintes d'intégrité inter-relations,
- clarification de la modélisation des traitements, compte tenu des différents niveaux de préoccupation et degrés de détails,
- extension des niveaux d'abstraction et des modèles avec l'émergence du modèle logique de traitements (MLT) et du modèle organisationnel de données (MOD) [Panet, Letouche, Peugeot 91],...
- couplage avec des méthodes de conduite de projet,
- mise sur le marché d'ateliers de génie logiciel (AGL) de conception spécifiques à la méthode.

Indiscutablement, la méthode MERISE constitue une étape importante en ingénierie de systèmes d'information.

Il n'en reste pas moins que la méthode MERISE présente actuellement des lacunes ; la principale concernant le passage de la conception du S.I. à la conception du logiciel. Rappelons à cette occasion qu'à l'origine de MERISE, la conception du logiciel (correspondant aux niveaux logiques et physiques) devait être supportée par des pratiques ou méthodes déjà utilisées sur le terrain.

2.2. *Positionnement de MERISE*

La méthode MERISE se présente comme une méthode de conception de systèmes d'information à informatiser. La notion de système d'information relève essentiellement du domaine de la gestion des organisations. La définition "merisienne" d'un système d'information est la représentation des activités d'un domaine (système opérant ou système de pilotage) et de ses échanges avec l'environnement, conçu à l'initiative du système de pilotage en fonction des objectifs à atteindre et de l'organisation choisie. Le système d'information comprend :

- la mémorisation des informations représentation des activités du domaine,
- les activités permettant de constituer et utiliser ces informations (acquisition, transformation, diffusion).

Cette définition d'un système d'information est indépendante des moyens nécessaires à sa mise en oeuvre. La conception et la formalisation d'un système d'information est d'abord un travail de "gestionnaire", autrefois assuré exclusivement par les "organisateurs". Le recours à l'informatique pour automatiser les fonctions de mémorisation, traitement et communication d'un système d'information a introduit une composante génie logiciel. Pour informatiser un système d'information, il faut d'abord concevoir le système d'information puis concevoir le logiciel. C'est cette double conception qui constitue une ambiguïté dans la conception des systèmes d'information informatisés, la conception du logiciel n'étant pas directement déductible de celle du système d'information.

La méthode MERISE a tenté de jeter un pont permettant aux deux disciplines (gestion et génie logiciel) d'assurer une continuité des démarches et raisonnements pour l'informatisation des systèmes d'information. Selon la méthode MERISE, l'informatisation d'un système d'information conduit à distinguer deux niveaux d'étude différents [Nanci, Espinasse et al.92]:

- le niveau du système d'information organisationnel (SIO) qui exprime l'activité organisée, associée au fonctionnement du système d'information perçu par les gestionnaires (signification des informations, fonctionnement des activités du domaine, échanges d'informations, organisation des tâches humaines et informatisées),
- le niveau du système d'information informatisé (SII) qui ne concerne que le contenu informatisé perçu par les informaticiens (logiciel, fichiers, bases de données, interfaces homme-machine (IHM)).

Cette distinction se traduit par des préoccupations et des raisonnements différents dans le processus de conception. Le SIO est essentiellement tourné vers les préoccupations des utilisateurs et fait appel à des disciplines des sciences de la gestion. Le SII est l'affaire exclusive des informaticiens et fait appel aux disciplines du génie logiciel. Cependant, le SII doit s'inscrire dans le SIO ; la conception du SII s'appuie donc sur celle du SIO.

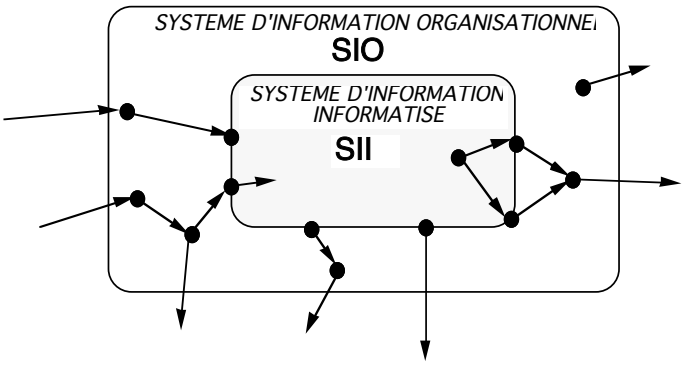


Figure 1 : *Les deux niveaux d'étude d'un système d'information.*

Pour assurer une continuité méthodologique, MERISE a, jusqu'à présent, proposé une démarche (cycle de vie) et des raisonnements (modèles et cycle d'abstraction) couvrant la conception du SIO puis du SII. Indéniablement, la contribution de MERISE a été plus originale, complète et appréciée dans la partie SIO; pour la conception du SII, elle s'est appuyée sur l'état de l'art en conception de logiciel (méthodologique et technologique) du début des années 80. Aujourd'hui, le développement du génie logiciel permet de réviser complètement démarche et les raisonnements de la partie SII.

2.3. *Niveaux d'abstraction et modèles*

Pour la spécification des données et du fonctionnement du SIO et du SII, la méthode MERISE utilise des modèles. Ces modèles s'articulent en niveaux d'abstraction qui expriment la prise en compte de préoccupations successives :

- Le *niveau conceptuel* exprime les choix fondamentaux de gestion en terme de signification des informations utilisées et d'activités exercées (recherche des éléments stables indépendamment des moyens à mettre en oeuvre, de leurs contraintes et de leurs choix organisationnels).
- Le *niveau organisationnel* exprime les choix d'organisation des ressources humaines et matérielles, au travers notamment de la définition de sites, de postes et de procédures.
- Le *niveau logique* exprime le choix des moyens et ressources informatiques en faisant abstraction de leurs caractéristiques techniques précises, au travers notamment des architectures logicielles.
- Le *niveau physique* traduit les choix techniques et la prise en compte de leurs spécificités.

Le croisement de ces 4 niveaux avec la dichotomie données-traitements conduit à la définition de 8 modèles spécifiques :

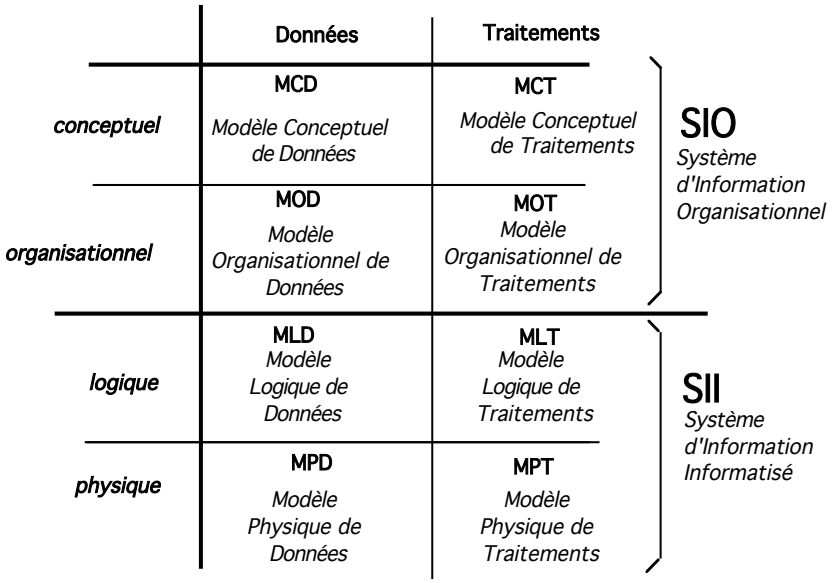


Figure 2 : *Les 8 modèles de MERISE 2° génération*

Ces modèles s'expriment suivant des formalismes définis dans la méthode MERISE, formalismes pouvant être commun à plusieurs modèles mais prennant alors en compte des préoccupations différentes.

2.4. *Démarche de MERISE*

La méthode MERISE propose, pour le déroulement du cycle de vie, un découpage en étapes relevant du modèle de la cascade :

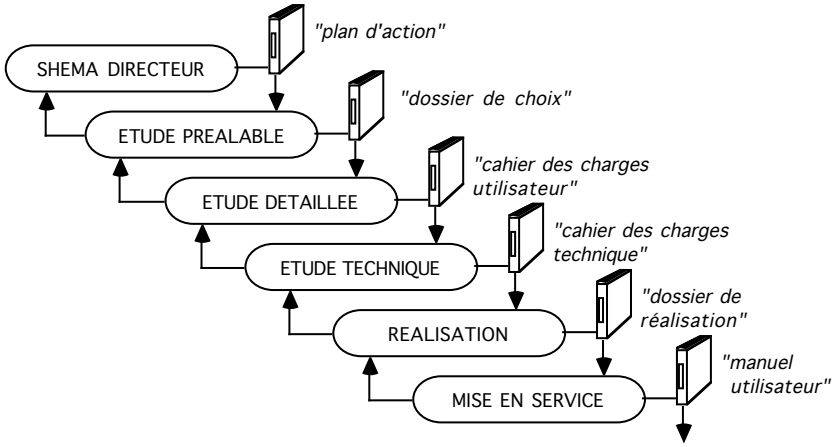


Figure 3 : *La démarche proposée par MERISE*

On peut assimiler les étapes d'Etude Préalable et Etude Détaillée aux phases d'Analyse et Définition des Besoins du Système" du génie logiciel. En effet, c'est à l'issue de l'Etude Détaillée qu'est produit le cahier des charges utilisateur.

3. HOOD et le génie logiciel

3.1. *Origine de HOOD*

La méthode HOOD fait suite à un appel d'offre international de l'Agence Spatiale Européenne (ESA). La méthode HOOD (Hierarchical Object Oriented Design), développée par un consortium franco-danois composé des sociétés Cisi-Ingénierie, Matra-Espace et CRI, a vu le jour en 1987 [Heitz 87], [Labreuille 87], [Gendre et al. 90], [Lai 91]. La méthode HOOD se présente comme une synthèse de la méthode OOD de G.Booch [Booch 87], des machines abstraites et des techniques de conception pour les systèmes temps réel. Elle fournit une démarche pour préciser les éléments d'une conception logicielle orientée objet, appliquée à la conception de systèmes importants et complexes du domaine scientifique et technique, principalement développés en langage ADA. La méthode HOOD évolue aussi, sous l'impulsion d'un groupe d'utilisateurs

(HUG) et du groupe de travail de l'ESA (HWG). La plupart des pays membres de la CEE ont fait ou font usage de la méthode HOOD. HOOD est ainsi la méthode retenue pour les projets HERMES et COLOMBUS ainsi que pour le programme EUROFIGHTER.

3.2. *Positionnement de HOOD*

La terminologie relative aux phases de développement d'un cycle de vie d'un logiciel n'est malheureusement pas unique. Il nous semble important de préciser que lorsque nous parlons de conception logicielle, il s'agit essentiellement de la phase de conception définie par l'AFNOR dans le cycle de vie en cascade. Or, une grande majorité des méthodes orientées objet s'intitulent Analyse et Conception, comme si les mots d'analyse et de conception n'en faisaient qu'un. Concrètement, les méthodes se préoccupent plus de spécifier en termes de classes et d'héritage un modèle d'information, plutôt que de se préoccuper d'un modèle logique à base de modules ayant effectivement une correspondance physique avec les unités de compilation du ou des langages cibles.

La spécificité de la méthode HOOD est d'avoir pour objectif la définition d'un modèle logiciel à partir duquel un modèle physique satisfaisant puisse être facilement déduit. Ainsi, la majorité des méthodes orientées objet sont plus proches de la phase de spécification, tandis que des méthodes telles que HOOD sont plus proches de la phase de code. Ceci ne remet pas en cause l'utilité des méthodes d'analyse et de spécification orientées objet. Nous constatons seulement qu'il est actuellement nécessaire de préciser une application en terme d'architecture logicielle afin de pouvoir établir, en début de la phase de codage, le plan de développement, le plan de tests, le plan d'intégration et de validation.

La méthode HOOD, directement issue de l'application des techniques récentes du génie logiciel, se base sur une démarche constructive d'une solution logicielle en termes d'architecture. De la même manière qu'en phase de spécification, il est nécessaire de procéder par étapes d'abstractions successives, la méthode HOOD permet d'appréhender graduellement comment les unités de programmation seront organisées entre elles. Une conception HOOD est l'équivalent d'une plan de masse pour un architecte.

Bien que de plus en plus les ateliers de développement ou de programmation permettent de gérer des applications importantes, il n'en reste pas moins qu'il faut avoir une démarche cohérente pour introduire de nouveaux modules de code. De plus, pour permettre un développement incrémental optimal, il faut également pouvoir déterminer, a priori, les relations de précédence introduites par les relations de compilation. Enfin, lorsqu'une démarche de conception est claire, il est possible d'en déduire facilement un plan de tests et d'intégration. C'est ce que permet l'application de la méthode HOOD. HOOD est donc une méthode essentielle pour de grosses applications. Il est néanmoins possible d'utiliser la méthode HOOD sur de petits projets en simplifiant certaines étapes et la documentation à produire.

3.3. *Modularité, abstraction et encapsulation dans HOOD*

De nombreux travaux relatent, expériences à l'appui, quels sont les principes essentiels que devraient suivre un logiciel pour avoir des chances d'atteindre les objectifs visés. Les principes importants du génie logiciel en termes d'architecture logicielle sont : la modularité, l'abstraction des données et l'encapsulation.

Un module est une partie de logiciel considérée comme formant un tout. Il existe plusieurs manières de décrire un module : logique, fonctionnelle, opérationnelle, structurelle et même physique. La méthode HOOD voit un module comme un objet. Les critères de complexité et de qualité d'une architecture HOOD sont basés sur des concepts de dépendance et de couplage entre objets (Cf. Les métriques HOOD dans [Lai 91]). Une "bonne" conception devrait introduire le moins de dépendances possibles. Ce principe pousse le concepteur d'une application à minimiser la taille des interfaces et à utiliser au maximum l'encapsulation.

L'abstraction des données consiste à définir les propriétés d'un objet en fonction du nom de l'objet et de l'utilisation que l'on désire en faire. Ainsi pour les services de la préfecture une voiture sera isomorphe à une carte grise et en fait aux informations que la carte doit contenir. Par abstraction, il est possible de parler de voitures alors qu'il s'agit seulement de parler de leur numéro d'immatriculation. Le concept de moteur et de chassis sont ici absents.

L'encapsulation va de pair avec la conception par contrat. Les objets utilisateurs doivent voir certaines opérations qui leur sont utiles et seulement celles-là. Toute opération ou tout objet qui n'a pas besoin d'être utilisé par tout le monde, doit être interne à un autre objet pour limiter sa portée de visibilité. L'encapsulation -ou principe de l'information cachée- est très importante. C'est grâce à elle qu'une application importante peut être effectivement maintenue car les dépendances entre objets sont réduites à quelques opérations.

3.4. *L'objet dans HOOD*

Parmi les méthodes de conception orientées objet on distingue, bien que la séparation ne soit pas toujours évidente à faire, les méthodes d'analyse des méthodes de conception. Les langages de programmation par objets utilisent généralement le concept de classes et d'instance de classes : les objets. Ainsi, suivant la phase où l'on se trouve on manipule le concept d'objet avec différents points de vue. Au niveau de l'analyse, ce sont surtout les objets du domaine d'application qui sont concernés. Au niveau du codage, les objets sont des variables de programmation accédées via des références - en fait des pointeurs. Au niveau de la conception logicielle, il peut s'agir de classes reprises de l'analyse, que l'on complète ou que l'on précise. Lorsque la relation d'héritage n'est pas nécessaire on considère des structures statiques pouvant être représentées par un module logique. La méthode HOOD est définie sur ce dernier principe.

Un objet HOOD implante un type abstrait de données. Rappelons qu'un type abstrait de données est défini par un nom, par un ensemble d'attributs (défini par une liste de noms) et par une liste d'opérations. Chaque opération est définie par une signature exprimant le domaine de définition (produit cartésien formé des types des paramètres) et le domaine d'arrivée (type de résultat de l'opération). Les types abstraits de données font l'objet de descriptions formelles rigoureuses cependant, pour construire les représentations nécessaires de données nous n'avons pas besoin de plus de choses.

Un objet HOOD possède une représentation graphique ainsi qu'une description textuelle qui doit être cohérente avec les informations graphiques contenues dans le diagramme dans lequel se trouve la représentation de l'objet. Du point de vue du concepteur, un objet HOOD représente un objet du monde réel introduit normalement en phase d'Analyse. Si l'objet ou la classe de l'objet n'a pas été introduit, l'objet HOOD constitue un utilitaire de la solution informatique ou un composant de la solution introduite par le concepteur. Lorsque le concept existe déjà il n'y a pas forcément correspondance biunivoque entre les objets de l'analyse et ceux de la conception. En effet, nous ne sommes pas au même niveau de granularité. En phase d'Analyse un objet est vu comme un composant alors qu'en phase de Conception un objet est vu comme un englobant. Ce qui est aisé à comprendre lorsqu'on sait qu'un objet HOOD constitue une vue logique d'un module de compilation.

Certaines méthodes utilisent le concept de module avec des significations différentes. Ainsi un module peut être une fonctionnalité ou une fonction ou une procédure de programmation. Dans le cas de HOOD le modèle de module utilisé est celui introduit par Ada avec les paquetages. Un paquetage et une unité de compilation comportant une partie appelée spécification du paquetage est une autre partie (pouvant être compilée séparément) appelée corps du paquetage. Le corps contient la partie implantation du module. Il n'est pas visible par les autres modules. Seules les parties publiques de l'interface du module est accessible aux autres modules qui en demandent l'utilisation -clause with-.

Au point de vue du modèle objet, un module issu d'un objet HOOD constitue une instance statique d'une classe isolée (sans sous-classes et sans super-classe) décrivant un type abstrait de données. Si plusieurs objets statiques d'une même classe sont nécessaires il est possible de paramétrer un module -module générique- chaque module est alors une instance d'un module générique.

Enfin rappelons que les objets HOOD sont caractérisés par un mode. Ce sont soit des objets passifs, soit des objets actifs, soit des objets de contrôle, soit des objets classe soit des objets d'environnement ou soit enfin des objets noeud virtuel.

3.5. *Demarche de HOOD*

Les cycles de vie de logiciel expriment la nécessité de passer par des étapes et par des phases correctement identifiées afin de faciliter le suivi et le contrôle du développement. Le cycle de la cascade normalisé par l'AFNOR permet de rationaliser correctement des projets importants. On connaît bien les défauts d'un tel modèle de processus. La remontée dans le cycle est très lourde et pénalisante. Les erreurs sont détectées très tard et celles-ci coûtent d'autant plus chères qu'elles ont été créées plus tôt.

L'essentiel de la phase de conception logicielle est la fourniture d'un dossier de conception. La méthode HOOD produit sous une forme "standard" un tel document. La documentation est formée à partir des graphiques de diagrammes d'objets et des descriptions textuelles. Chaque objet possède une description textuelle (ODS). Les outils de conception supportant la méthode HOOD doivent assurer la cohérence entre les descriptions textuelles en particulier sur les relations de visibilité entre les parties produites et les parties requises. La grammaire décrivant les ODS définit un langage intermédiaire indépendant des machines cibles et des langages de programmation qui seront par la suite utilisés. En fin de conception on passe à la phase de codage. Le codage est initialisé en générant les squelettes de code à partir des descriptions textuelles.

Au niveau du plan qualité l'engagement de suivre une méthode telle que HOOD par une équipe formée à son utilisation est un élément important mais non suffisant. Il reste encore à contrôler les bonnes règles d'application. Ceci se fait généralement par des lectures entre les intervenants. Il est possible également d'utiliser des métriques faisant apparaître clairement que la conception n'est pas équilibrée ou qu'elle présente des couplages anormaux.

4. Comparaison des deux méthodes

4.1. *Axes et critères de comparaison*

Comme nous l'avons souligné dans le paragraphe précédent, MERISE et HOOD n'ont pas le même positionnement en génie logiciel. Afin de mieux appréhender la complémentarité et l'articulation de ces deux méthodes, nous avons établi une comparaison selon différents axes, chacun étant détaillé par des critères d'appréciation :

Champ d'application : domaine d'application, diffusion, outils.

Cycle de vie, démarche : nature de la démarche, position par rapport au cycle de vie, rappel des étapes, phases et tâches, aspect itératif.

Niveaux d'abstraction : nature ou base du découpage en niveaux, passage inter-niveaux.

Modélisations et formalismes : objets modélisés, modèles fonctionnel, structurel, événementiel.

Mise en œuvre: organisation, normes, qualité, conduite de projet.

L'explicitation de ces axes et critères ainsi que contenu détaillé de cette comparaison sont présentés sous forme de grille en annexe.

4.2. *Synthèse*

L'analyse comparative des critères d'évaluation a parfois été difficile, chaque méthode ayant son propre positionnement et sa propre interprétation des concepts. Néanmoins, on retiendra les points suivants :

1- MERISE et HOOD bénéficient toutes deux d'une notoriété certaine dans des domaines d'applications différents. Toutes deux sont supportées par des outils logiciels.

2 - Les deux méthodes comportent une démarche structurée. Ces démarches sont mixtes (ascendant et descendant) bien que concernant des phases différents du cycle de vie AFNOR. MERISE se positionne en amont de HOOD.

3 - Bien que de natures différentes, les deux méthodes utilisent des niveaux d'abstraction. Dans MERISE ces niveaux correspondent à la prise en compte progressive de préoccupations homogènes. Dans HOOD, ces niveaux sont moins formels et sont associés plutôt à une présentation d'une organisation d'objet de niveau d'intérêt semblable.

4 - Les objets modélisés par MERISE sont des données et du fonctionnement (les traitements) abordés selon les points de vue des différents niveaux. Dans HOOD les objets sont des objets logiciels réalisant des traitements sur des données. Ces objets sont articulés (pour MERISE en différents modèles) afin de décrire la solution de façon fonctionnelle, structurelle et temporelle.

5 - Dans leur mise en œuvre les deux méthodes ont le soucis de préconiser une répartition du travail. Elles préconisent des procédures d'assurance qualité, même si des normes ne sont pas toujours formellement définies (MERISE). Les deux méthodes n'abordent pas la dimension conduite de projet, pour laquelle elles peuvent s'associer à d'autres méthodes spécialisées.

5. Complémentarité et articulation des deux méthodes

5.1. *Orientations générales adoptées*

L'orientation objet des méthodes de conception peut sembler inévitable. Dans ce mouvement, il faut reconnaître l'influence importante des méthodes orientées objet, principalement d'origine nord américaine, comme OOD de G. Booch [Booch 93], OOA de Coad & Yourdon [Coad et Yourdon 92], OOA/OOD de Shlaer & Mellor [Shlaer et Mellor 90], OMT de Rumbaugh [Rumbaugh et al. 91]. Toute ces méthodes peuvent être qualifiées de méthode de génie logiciel et non de conception de système d'information.

De nombreux chercheurs pensent toutefois que cette (r)évolution objet est également inévitable pour les méthodes de conception de systèmes d'information ; elles devraient avoir une approche objet dès le niveau conceptuel du SIO, notamment [Cauvet & Rolland 92], [Castellani 93], [Rochfeld et Bouzeghoub 93]. Pour notre part, nous estimons cette affirmation quelque peu prématurée et souhaitons y opposer quelques contre-arguments.

Tout d'abord, quinze années d'expérience témoignent que la méthode MERISE, 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. Les principales critiques portent sur l'absence de continuité entre les spécifications conceptuelles/organisationnelles et les spécifications informatiques, surtout au niveau des traitements, permettant d'arriver à l'architecture logicielle. Sur ce dernier point, il faut remarquer que, dans les méthodes objet précédemment citées, la même césure entre analyse et conception du logiciel se présente.

Dans OMT [Rumbaugh et al. 91] on distingue certes les phases d'analyse, de conception système et de conception objet, mais ce n'est qu'une description du processus d'abstraction menant aux modèles détaillés. Dans OOA/OOD [Shlaer et Mellor 90], la phase d'analyse est bien détaillée, mais la phase de conception logicielle reste assez vague. L'argument de "continuité-unité méthodologique" avancé pour les méthodes objet reste donc à ce jour à confirmer.

Ensuite, l'approche objet nous apparaît réellement opérationnelle que dans le génie logiciel, au niveau de la conception de l'architecture technique de programme (conception orientée objet), ainsi que pour la réalisation (programmation orientée objet). En outre, bien que très séduisants, les systèmes de bases de données orientés objets sont aujourd'hui encore au stade expérimental. Il ne peuvent actuellement apporter une réponse efficace à la gestion d'un volume important de données structurées et partagées tel que le connaissent tous les systèmes d'information en activité. Le tout objet n'est pas encore tout à fait prêt pour "révolutionner" la situation actuelle. Il nous semble que les praticiens qui ont ces dernières années majoritairement opté pour des SGBD relationnels, mettent plus d'espoir dans les bases de données relationnelles actives [Masi 93] qui leur permettraient de mieux associer données et traitements. Dans ce cas, on ne voit pas pourquoi les raisonnements préconisés par la méthode MERISE pour obtenir des structures relationnelles (et qui ont connu le plus de succès) seraient remis complètement en cause.

Enfin, l'adoption et la mise en œuvred'un ensemble méthodologique comme MERISE a constitué pour les entreprises et les sociétés de services en informatique (SSII) un investissement considérable, tant financier qu'humain, qui ne peut être remis en cause du jour au lendemain. Dans le contexte actuel, une remise en cause radicale nous apparait inenvisageable 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.

Par contre il nous semble tout à fait possible, après une spécification du SIO selon les modèles proposés par MERISE, de passer à une spécification du SII orientée objet, ceci afin de tirer parti des avantages de l'approche objet dans le domaine du génie logiciel. L'analyse comparée des méthodes MERISE et HOOD a mis en évidence, pour cette dernière,

ses capacités de structuration dans l'élaboration des architectures logicielles, lacune majeure de MERISE. Ces aspects "objet" et structuration du logiciel pourraient ainsi rénover de façon significative la mise en œuvredes modèles logiques et physiques de données et de traitements. C'est l'orientation générale adoptée par le projet MERISE+.

Dans cette perspective,

- pour la conception du SIO, nous conservons les modélisations de la méthode MERISE (MCD, MOD, MCT, MOT) ainsi que les étapes de la démarche concernées (étude préalable et étude détaillée).
- pour la conception du SII, nous adoptons la méthode HOOD en aménageant sa démarche et ses raisonnements pour assurer une interface avec les modélisation issues du SIO.

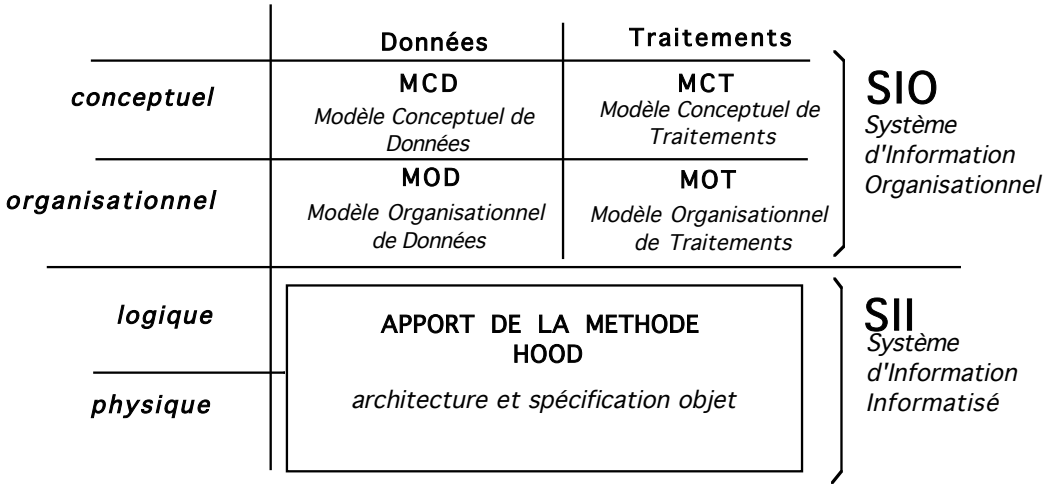


Figure 4 : *Articulation générale des méthodes HOOD et MERISE dans le projet MERISE+*

Il ne s'agit pas de définir une nouvelle méthode remplaçant l'une ou l'autre méthode existante mais, par l'alliance des deux méthodes, de tirer bénéfice des forces et faiblesses de chacune d'elles, tout en offrant une continuité méthodologique opérationnelle.

Les praticiens de la conception de systèmes d'information pourront ainsi disposer de plusieurs options méthodologiques leur permettant d'introduire graduellement l'approche objet suivant les techniques et cibles de développement à leur disposition. Par exemple, comme l'illustre la figure 5, quatre grandes options méthodologiques sont possibles :

1 - une équipe devant concevoir un système d'information dans un environnement "classique" conservera intégralement la méthode MERISE.

2 - une équipe souhaitant s'orienter vers des bases de données actives dans un environnement L4G, continuera à utiliser essentiellement MERISE tout en introduisant certains raisonnements de MERISE+ permettant d'incorporer les traitements dans les données.

3 - une équipe souhaitant s'orienter vers un environnement de développement objet pour concevoir un système d'information trouvera la démarche et les modèles adaptés dans MERISE+. Actuellement peu utilisée, cette option commence cependant à se développer. En effet, du fait de la "microtisation" (down-sizing) des systèmes informatisés, la part des logiciels, locale aux postes clients dans une architecture client-serveur, s'accroît de plus en plus. Il est alors nécessaire d'organiser un logiciel dans un système distribué. L'utilisation d'objets (ou "proxy") implantés sous forme d'objets appelés noeuds virtuels permettent d'appréhender des solutions intégrant des concepts mis en avant par l'ingénierie ainsi que de standard tel que CORBA de l'OMG [Corba 93].

4 - les équipes confrontées à la conception et au développement d'applications ne relevant pas réellement de système information continuerons d'appliquer la méthode HOOD.

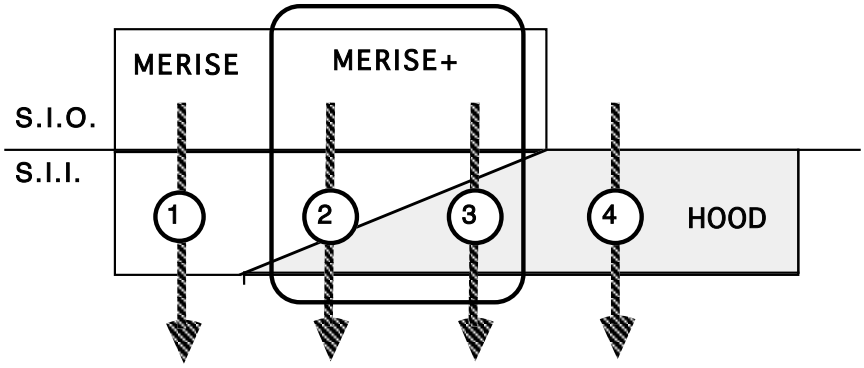


Figure 5 : *Différentes options méthodologiques possibles.*

5.2. *Articulation HOOD-MERISE*

L'apport de HOOD se situe ainsi au niveau logique et plus précisément sur le volet des traitements. HOOD ne remettant pas en cause la dichotomie Données - Traitements de MERISE, les objets abordés étant des objets logiciels utilisant des objets "statiques" (ou données) considérés comme des ressources externes accessibles. Cet apport de HOOD, défini dans une problématique purement génie logiciel, ne peut s'articuler directement avec les spécifications de niveau organisationnel de MERISE. Une phase intermédiaire permettant cette transition s'avère nécessaire. Ceci nous a conduit à proposer une différenciation du niveau logique en deux : le niveau logique 1 qui permettra de conduire cette transition et le niveau logique 2 qui, à partir du précédent niveau mettra en œuvrela méthode HOOD.

5.2.1. Le niveau logique 1

Le niveau 1 permet de reprendre les spécifications des MOD et MOT MERISE et de les compléter de spécifications, choix ou contraintes nouvelles de niveau logique, HOOD ne se préoccupant que d'objets logiciels. Cependant, la conception de ces objets logiciels passe par une intégration des données et des traitements.

Aussi, il nous semble judicieux, dans la définition de ce niveau logique 1, de faire en sorte que les effets de la dichotomie amont (MOT/MOD) soient réduits voire évacués. Ceci semble possible en développant à ce niveau logique 1 une modélisation orientée objet. Comme la plupart des méthodes orientées objet de génie logiciel, trois modèles (fonctionnel, structurel et temporel) seront nécessaires pour spécifier ce niveau logique 1 :

- le Modèle Structurel (MS) : ce modèle permet la spécifications des classes en indiquant leurs comportement spécifiques. Ce modèle, de type classe-relation sera dérivé du MOD (Modèle Organisationnel de Données) dont il héritera de la structure et du MOT (Modèle Organisationnel de Traitements) dont il héritera d'aspect comportementaux intrinséques aux données concernées par les tâches.
- le Modèle Logique de Traitement (MLT) : il s'agit d'un modèle fonctionnel directement issu du MOT. Il est plus détaillé notamment en ce qui concerne les classes et relations concernées (suivant le niveau du MOT, le MLT peut être alors très réduit),
- le Modèle d'Etats (ME) : le modèle d'états participe à la spécification du comportement d'un objet (via le comportement défini sur une classe). Auparavant, les diagrammes d'états-finis étaient associés à la spécifcation détaillée d'un processus terminal dans une décomposition fonctionnelle (SA-RT). Lorsqu'une classe nécessite un protocole d'utilisation (ordonnancement dans l'appel de ses méthodes) celui-ci est décrit par un diagramme d'états. Une classe possédant un diagramme d'état sera appelée classe active.

5.2.2. Le niveau logique 2

Le niveau logique 2 permet à partir du niveau logique 1 de concevoir l'architecture logicielle du SII. Trois modèles sont nécessaires :

- le modèle des objets logiciels statiques à mémoriser (données, objets persistants,....) dans des bases de données relationnelles (actives ou non) ou objets ainsi que l'architecture de ces dernières. On parlera de Modèle Logique de Données (MLD).
- le modèle des objets logiciels spécifiant l'architecture des modules logiciels applicatifs exploitant les classes ainsi que les objets logiciels "interfaces" de l'IHM. Ce modèle est un modèle HOOD.

A ces divers modèles logiques doit être rajouté un Modèle des objets logiciel Interface Homme Machine (MIHM) spécifiant l'interface homme-machine. Ce dernier modèle recouvre les deux sous-niveaux logiques.

Les différents modèles associés à ces 2 niveaux logiques sont illustrés par la figure suivante :

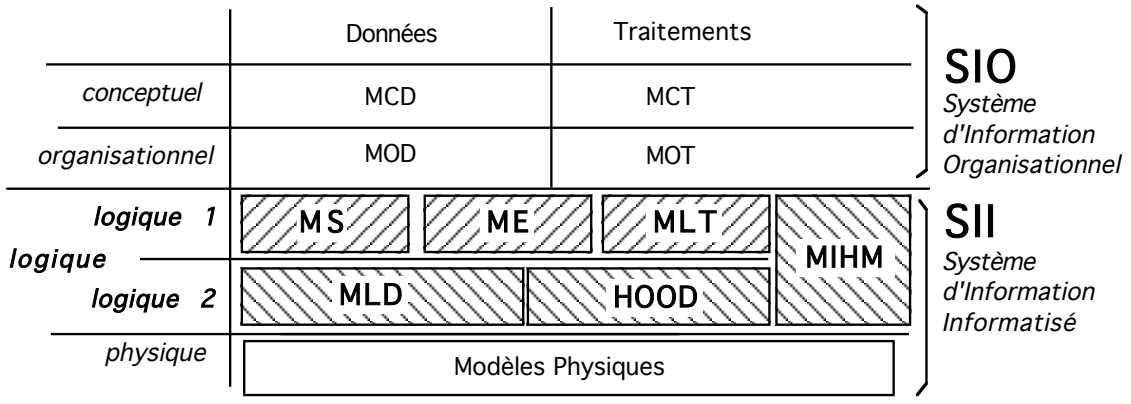


Figure 5 : *Les modèles du niveau logique*

Le passage de ces modèles logiques de niveau 1 aux modèles logiques de niveau 2 permet de prendre en compte de nouveaux choix ou contraintes de plus en plus technologiques, notamment le fait de choisir une technologie centrée sur une Base de Données Relationnelle (active ou non) ou Objet. Il est évident qu'un tel choix a des incidences sur les modèles de données (MLD) mais aussi sur les modèles HOOD de traitements.

Enfin, "l'ancrage" de HOOD à des modèles logiques objets du niveau logique 1 peut poser quelques difficultés. La principale étant que ces modèles logiques objets de niveau 1 disposent des concepts de classe et d'héritage, ce qui nous apparaît comme fortement souhaitable (d'autant que MERISE deuxième génération utilise déjà un peu ces concepts sous la forme de sous-types dans les modèles de données). Ce type de spécification ne peut pas être exploité en HOOD, qui ne dispose pas de ce concept. Aussi, nous avons pensé coupler MERISE non plus à HOOD standard (ne disposant pas de ce concept de classe) mais à une de ses extensions, la méthode MOOD [Lai 93].

6. Conclusion

La principale lacune que présente la méthode MERISE concerne le passage de la conception du S.I. à la conception du logiciel. le projet MERISE + se propose de combler les insuffisances de la méthode MERISE au niveau du SII par un enrichissement avec la méthode orientée objet HOOD. Il ne s'agit pas de définir une nouvelle méthode remplaçant l'une ou l'autre méthode existante mais, par l'alliance des deux méthodes de tirer bénéfices des forces et faiblesses de chacune d'elles, tout en offrant une continuité méthodologique opérationnelle.

Comme nous l'avons développé, l'essentiel de l'apport de HOOD se situe ainsi au niveau logique et plus précisément sur le volet des traitements. HOOD ne remettant pas en cause la dichotomie Données - Traitements de MERISE, les objets abordés étant des objets logiciels utilisant des objets "statiques" (ou données) considérés comme des ressources externes accessibles. Cet apport de HOOD, défini dans une problématique purement génie-logiciel ne peut s'articuler directement avec les spécifications de niveau organisationnel de MERISE. Une phase intermédiaire permettant cette transition s'avère nécessaire. Ceci nous a conduit à proposer une différenciation du niveau logique en deux : le niveau logique 1 qui permettra de conduire cette transition et le niveau logique 2 qui, à partir du précédent niveau mettra en œuvrela méthode HOOD ou plus précisement son extension aux classes, la méthode MOOD [Lai 93].

Nous définissons actuellement les différents modèles de ces deux niveaux, leurs formalismes, les règles d'élaboration et de passage selon les grandes option méthodologiques retenues par les concepteurs. Ce papier avait pour but d'introduire les modèles introduits dans la méthode MERISE+. Les travaux en cours permettront prochainement de les présenter plus en détail ainsi que d'exposer la démarche proposée.

Post-scriptum

Cet article a été présenté au Congrés INFORSID en mai 94, il correspond aux résultats des études réalisées dans le cadre du projet MERISE+ pour la période s'étant terminée en décembre 93. Depuis les travaux ont portés sur : des compléments sur le formalisme Entité-Relation (MCD, MOD) notamment sur la formalisation des règles et des contraintes sur les propriétés - des compléments sur le formalisme des traitements (MCT, MOT) précisant les notions de Sous Schéma de Données et introduisant la formalisation des changements d'état (pré-condition, post-condition) - la définition du formalisme (concepts et représentations graphiques) du modèle de classe (MC) - les règles de dérivation d'un modèle de classe à partir d'un modèle de données en Entité-Relation (MCD/MOD) - la définition du formalisme (concepts et représentations graphiques) des modèles dynamiques (MD) : Diagramme d'Etat de Classe, Diagramme de Communication de Classes - la définition du formalisme (concepts et représentations graphiques) du modèle de module (MM) - des principes généraux de constitution des modules à partir des classes - une illustration des modélisations par un cas.

Remerciements

Les auteurs tiennent à remercier Bernard Cohen de la société CECIMA, Xavier Linais de la société INGENIA, Marc Bergman, Papa Coly Faye, Sophie Nabitz et Gilles Trouche de l'Institut SYSTEMIA pour leur contribution significative au projet MERISE+.

Bibliographie

[Booch 88], Booch G., *Ingénierie du logiciel avec Ada*, InterEditions, 1988.

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

[Castellani 93], Castellani X., *Méthodologie générale d'analyse et de conception des systèmes d'objets ; tome 1 : L'ingénierie des besoins,* Masson, 1993.

[Cauvet et Rolland 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 et Yourdon 92], Coad P., Yourdon Y., traduction française de A. Boughlan, *Analyse Orientée Objet*, Masson, 1992.

[CTI 79], Centre Technique Informatique, Méthode de définition d'un système d'information, mission informatique, ministère de l'industrie, 1979.

[Gendre et al. 90], Gendre P., Bitteur H., « HOOD, version 3.0 : l''âge de raison », *Génie Logiciel & Systèmes Experts*, n° 19, juin 1990.

[Heitz 87], Heitz M., (CISI Ingénierie Toulouse), « HOOD, une méthode de conception hiérarchisée orientée objets pour le développement des gros logiciels techniques et temps réel », *Bigre n° 57*, Journées Ada France, déc. 87.

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

[Labreuille et Farail 87], Labreuille B., Farail P., (CISI Ingénierie), « Application d'Ada aux systèmes temps réel distribués », *Bigre n° 57*, Journées Ada France, déc. 1987.

[Lai 91], Lai M., *Conception Orientée Objet : Pratique de la méthode HOOD*, Editions Dunod Informatique, 1991, 320 pages, retirage mai 92.

[Lai 93], Lai M., « La méthode MOOD », Conférence Internationale du Génie Logiciel, EC2, La Défense, 1993.

[MESSAGE 87-92], L'atelier MESSAGE: Manuel d'utilisation, CECIMA. Aix-en-Provence, 88-92.

[Masi 93], Laboratoire MASI, Université de Versailles, *Bases de Données Actives*, Paris, Journée du 8 décembre 93, publication AFCET.

[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*, Editions Sybex, mai 1992, 650 pages, 2° édition.

[Panet et al. 91], Panet G., Letrouche R., Peugeot C., « De MERISE à MERISE/2 : techniques de "refinement" et nouvelles architectures d'application », in *Autour et à l'entour de MERISE : les méthodes de conception en perspective*, Actes du congrès AFCET-CERAM, 17-18-19 avril 1991, Sophia-Antipolis, publication AFCET, 1991.

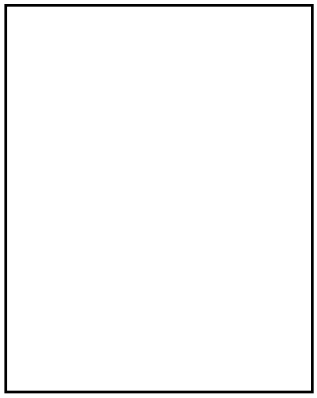
[Rochfeld et Bouzeghoub 93]., Rochfeld A., Bouzeghoub M., «From MERISE to OOM », in *Ingénierie des systèmes d'information*, vo.1 N°2, pp 151-176, 1993.

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

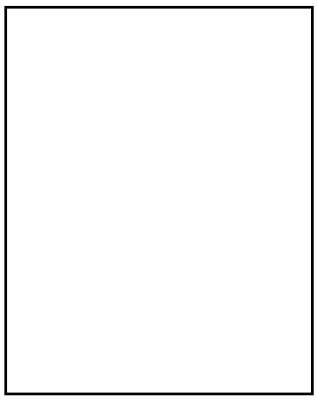
[Shlaer et Mellor 90], Shlaer S., Mellor S.J., *Object life cycles : Modeling the world in states*, Prentice Hall, 1990.

[Tardieu et al. 83], Tardieu H., Rochfeld A., Coletti R., Vahée M., *La méthode MERISE*, Editions d'organisation, 1983.

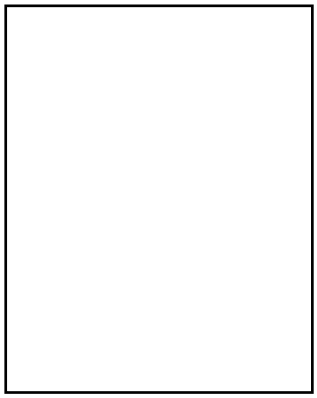
Biographie



Bernard Espinasse est Maître de Conférences à l'Université Aix-Marseille III où il enseigne à l'IUP MIAGE. Il assure aussi la responsabilité scientifique de l'Institut SYSTEMIA qui dispense en collaboration avec l'Ecole des Mines d'Alès et Sup. Télécom. Paris, des Mastères Spécialisés en Systèmes d'Information. Ingénieur Arts et Métiers, Docteur de troisième cycle en Systèmes d'Information et diplômé de l'IAE. Il participe dès 1979, aux recherches sur la méthode MERISE au sein de l'équipe animée par Hubert Tardieu. Il a enseigné durant trois années les systèmes d'information à l'Université Laval (Canada) en tant que Professeur Adjoint. Il menne actuellement ses recherches au sein du DIAM-LGIM, recherches centrées sur la conception et la réalisation de systèmes d'information et d'aide à la décision distribués.



Michel Lai est actuellement Directeur Qualité-Méthode de INGENIA, société de services en informatique avancée et intelligence artificielle ainsi que Professeur associé à l'IUP MIAGE de l'Université d'Aix-Marseille III. Il est titulaire d'un doctorat de troisième cycle en Intelligence Artificielle de l'Université d'Aix-Marseille 2. Ses travaux actuels portent sur l'amélioration du processus de réalisation de logiciel industriel de qualité par l'apport notamment des technologies objet. Il est enfin auteur d'un ouvrage et de plusieurs articles sur la méthode HOOD.



Dominique Nanci est Directeur Technique de CECIMA, société de conseil en ingénierie de systèmes d'information, spécialisée dans la méthode Merise et concepteur de l'AGL Message. Ingénieur INSA LYON, diplômé de l'IAE, Docteur de troisième cycle en Système d'Information. De 1973 à 1978, enseignant-chercheur à l'IAE d'Aix, il participe au sein de l'équipe "aixoise" d'Hubert Tardieu, aux travaux de recherche initiaux sur la méthode MERISE. En une quinzaine d'années de mise en pratique de cette méthode, à travers l'animation de plusieurs centaines de séminaires de formation et par la mise en oeuvre opérationnelle dans de nombreuses études et missions d'expertise, il a capitalisé une grande expérience dans la mise en œuvre de cette méthode. Il est enfin co-auteur de deux ouvrages et de plusieurs articles sur la méthode MERISE.

8 - Annexe : tableaux de comparaison des methodes MERISE et HOOD

MERISE	HOOD
1 - Axe champ d'application	
Domaine d'application	*Domaines pour lesquels la méthode a été initialement conçue et dispose de l'essentiel de ses références d'utilisation.*
A l'origine (74-76), la méthode MERISE à été élaborée pour concevoir des bases de données dans le domaine de l'informatique de gestion. Influencée par l'approche systémique en gestion, elle s'est rapidement (76-80) enrichie en abordant également les traitements dans une optique plus générale de systèmes d'information organisationnels. La méthode MERISE est majoritairement mise en œuvredans le domaine de la gestion.	Bien qu'initialement développée par des industriels du secteur spatial ou aéronautique la méthode HOOD n'est pas spécifique d'un domaine particulier. Le concept d'objet représente un modèle logique applicable dans n'importe quel domaine.
Diffusion :	*Place sur le marché des méthodes et implantation dans des secteurs d'activité.*
Soutenue par le Ministère de l'Industrie lors des diffusions des résultats de la recherche (79-80), la méthode MERISE s'est progressivement "imposée" auprès de l'Administration. Les grandes sociétés confrontées à des problèmes de conception de bases de données l'on assez rapidement adoptée. Aujourd'hui la méthode MERISE est la méthode la plus utilisées en France et est devenue un standard de fait. Elle est largement enseignée non seulement dans les formations informatiques (écoles d'ingénieurs et universités) mais également dans certaines écoles de gestion. De très nombreux ouvrages ont été publiés dont certains à plus de 30 000 exemplaires. Au delà de la France, la méthode MERISE est assez bien popularisée dans le monde francophone. Elle est pratiquée également, sous des adaptations locales, dans les pays de l'Europe du Sud.	La méthode HOOD est partiellement supportée par l'Agence Spatiale Européenne qui impose à ses sous-traitants d'utiliser la méthode HOOD. Les évolutions de la méthode est contrôlée par une association d'utilisateurs composée d'un groupe de direction (HSG), d'un groupe d'experts techniques (HTG) et de l'ensemble des utilisateurs de la méthode (industriels, centres de recherche et universitaires -HUG). La méthode HOOD est enseignée dans certaines universités et certaines écoles d'ingénieurs.
Outils	*Présence d'outils informatisés (A.G.L.) supportant cette méthode.*
Dès 83, les premiers outils associés à MERISE ont été commercialisés. L'offre a évolué avec la technologie et s'est enrichie de nombreux produits. On peut estimer aujourd'hui qu'une vingtaine de logiciels sont directement dédiés à la mise en œuvrede MERISE, sans compter les produits AGL qui annoncent "supporter" la méthode MERISE.	STOOD de TNI, AdaNice d'Intects, Concerto (CNET/SEMA), VSF,...
2 - Axe cycle de vie - démarche	
Nature de la démarche	*Parcours des différents niveaux d'abstraction ou de détail. Désignée souvent par ascendante (bottom-up), descendante (top-down), ou mixte..*
On peut qualifier la démarche MERISE de mixte. A partir de la description de la situation existante (données et traitement), le concepteur, faisant abstraction des composantes techniques et organisationnelles, élabore les modèles conceptuels de données et de traitements. Sur ces modèles, il prend en compte les nouvelles orientations et bâti le niveau conceptuel du futur système. On peut qualifier cette démarche d'ascendante. Dans les phases suivantes de conception du système futur, à partir des modèles conceptuels, le concepteur prend en compte successivement les préoccupations organisationnelles puis techniques. On peut qualifier cette démarche de descendante.	La décomposition s'effectue à partir d'un objet racine suivant une arborescence décrivant un arbre de conception. Cependant à chaque étape le concepteur doit se poser la question, l'objet dont j'ai besoin existe-t-il déjà, si oui il faut introduire ou réutiliser des classes qui sont en fait des modèles génériques de modules à partir desquels des instances d'objets seront construites.
Position par rapport au cycle de vie	*Les démarches de conception de systèmes d'information et de logiciel peuvent se classifier selon des cycles types : .cascade, en V, en spirale. La norme AFNOR proposant le cycle en V classique. Quelles phases sont couvertes par chaque méthode ?*
MERISE est essentiellement une méthode de conception de système d'information. Bien que sa démarche intègre également les aspects conception du logiciel (étude technique, production du logiciel), son apport se positionne essentiellement sur les phases de conception et spécification du système.	La position de la méthode HOOD est la suivante : fin des spécifications logicielles, conception globale ou préliminaire - logicielle-, début de la phase de codage. La méthode HOOD est utilisée principalement pour exprimer l'architecture logicielle, néanmoins, les frontières entre les phases précédentes et suivantes de la conception sont partiellement prises en compte par la méthode HOOD.

Rappel des étapes phases tâches	*Présentation succincte du découpage de la démarche, pour la partie concernée du cycle de vie.*
La méthode MERISE propose un découpage en trois niveaux de détail : les étapes, les phases puis les tâches. Le découpage en étapes, voire en phases recueille le consensus des praticiens; le niveau de découpage en tâches ainsi que leur enchaînement fait l'objet de plus de personnalisations en fonction du contexte. Nous avons retenu le découpage suivant : Etape / phase **Etude Préalable** Lancement Analyse de l'existant Conception des solutions Evaluation des solutions **Etude Détaillée** Conception général Conception détaillée Procédures transitoires Procédure de secours **Etude Technique** Conception logique Spécifications physiques Préparation réalisation **Production du logiciel** Organisation production Ecriture programmes Qualification des codes Intégration Qualification interne **Mise en service** Planification Mise en place ressources Préparation lancement Mise en exploitation	La méthode HOOD reprend l'étape de base introduit par G.Booch. Pour chaque objet on écrit une stratégie informelle exprimant, d'une manière volontairement concise, la solution apportée par l'objet. A partir d'un texte de stratégie informelle on applique les activités élémentaires suivantes : - recherche des objets et des attributs par l'identification des noms - recherche des opérations par l'identification des verbes - regroupement des noms d'objets, caractéristiques et opérations - établir les relations d'utilisation entre les objets - formalisation de l'interface des objets (écriture dans un pseudo-code) et représentation graphique.
Aspect itératif	*La notion d'itération traduit la réutilisation de raisonnements ou modèles à des phases successives de la démarche. On précise ci-après la nature de l'itération et son instant de mise en œuvre dans la démarche.*
Dans les phases de conception (étude préalable puis détaillée), le concepteur applique les mêmes modélisations (MCD, MCT, MOD, MOT) sur les mêmes données et traitements à modéliser mais avec des degrés de détails différents. L'itération se justifie également par l'étendue du champ d'étude. En étude préalable on travaille sur un sous-ensemble représentatif; en étude détaillé on généralise à l'ensemble du domaine à étudier.	Le résultat d'une étape de conception est une suite d'objets avec leur interface externe. Il suffit d'appliquer à nouveau sur chaque objet produit les actions d'une nouvelle étape. Lorsque le concepteur juge que le détail d'un objet est suffisant pour passer à la phase de codage (objet terminal) il stoppe le processus d'itération. (Décomposition en "largeur" d'abord).
3 - Axe niveaux d'abstraction Nature ou base de découpage en niveaux	*Nature ou base de découpage en niveaux : chaque méthode applique dans ses raisonnements un processus de découpage en niveaux. Quelle est la nature de ce découpage ?*
La méthode MERISE propose un découpage en niveaux d'abstraction (inspiré initialement des niveaux ANSI-SPARC) qui exprime la prise en compte progressive de différentes préoccupations. La méthode MERISE propose aujourd'hui 4 niveaux : - le niveau conceptuel qui prend en compte les choix de gestion -pour les données : la signification des informations utilisées dans le domaine sans contrainte technique ou économique de mémorisation -pour les traitements: l'activité du domaine conditionné par ses échanges avec l'environnement sans prise en compte des ressources nécessaires à l'exercice de ces activités. - le niveau organisationnel qui prend en compte les choix d'organisation -pour les données: le choix de mémorisation des informations en fonction du volume et le choix de disponibilité des informations par les structures organisationnelles, sans contrainte logicielle particulière. -pour les traitements: le choix des moyens (répartition homme/machine, répartition en postes de travail, enchaînement en procédure, travail immédiat ou différé) - le niveau logique qui prend en compte les choix de moyen logiciel -pour les données: choix de mémorisation informatique en fonction de l'équilibre coût performance et des technologies utilisées. -pour les traitement: choix de l'architecture logicielle (découpage en modules) - le niveau physique qui traduit les "fichiers" et "programmes" concrétisant l'application informatique support du système d'information.	La méthode HOOD est basée sur la représentation de types abstraits de données. Chaque objet est la représentation d'un type abstrait de données. Les services d'un objet sont implantés par des opérations d'objets fils qui sont encapsulés dans l'objet parent. Ceci forme une hiérarchie d'objets reliés par la relation d'inclusion. La hiérarchie sénior/junior est issue de méthodes basées sur les machines abstraites placées sur des couches hiérarchiques. La relation sénoir/junior exprime une relation d'utilisation entre un objet plus "élevé" avec un objet du même niveau ou du niveau inférieur. Dans les machines abstraites les règles sont plus strictes qu'en HOOD. HOOD utilise un autre objet sans qu'il soit nécessairement sur une couche immédiatement inférieure. De ce fait le découpage en couches reste assez informel. Cependant des restrictions demeurent entre objets passifs et objets actifs. Les objets actifs font partie des niveaux supérieurs, tandis que les objets passifs sont nécessairement dans des couches inférieures. Normalement la relation utilise ne devrait pas "remonter".

Passage inter niveaux	*Passage inter niveaux : quels raisonnements sont appliqués dans chaque méthode pour passer d'un niveau à l'autre ? Quelle est la nature de ces raisonnements (règles de dérivation formelle, heuristique) ?*
Dans la méthode MERISE, seul le passage du MCD-MOD (exprimé en formalisme entité-relation) vers le MLD (exprimé en formalisme relationnel) est basé sur des règles formelles de transformation. Les autres passages s'effectuent d'une manière moins formelle par la prise en compte des choix et contraintes exprimées à chaque niveau. Dans les traitements, le passage d'un niveau à un autre se traduit également par une décomposition (raffinement) des activités.	relations USE (senior/junior), INCLUDE (hiérarchie d'objets)
4 - Axe modélisation / formalisme Objets modélisés	*Objets modélisés : quels objets ou concepts du monde réels intéressent la méthode et seront donc pris en compte dans la modélisation. Autrement dit, sur quoi travaille la méthode ?* *Modèles : de nombreux spécialistes proposent de classer les méthodes en se basant sur les notions prises en compte dans les modèles utilisés; d'une façon générale, ces typologies retiennent les aspects fonctionnement, structure et comportement. Nous avons choisi d'examiner si et comment chaque méthode modélisait ces trois aspects.*
Globalement, la méthode MERISE modélise deux catégories d'objets: les données et les activités (fréquemment appelées traitements) d'un système d'information. Ces objets modélisés prennent des appellations différentes suivant les niveaux de modélisation.	Les entités représentées par HOOD sont : les objets passifs, les objets actifs, les opérations de contrôle, les classes (au sens module générique), les objets d'environnement et les nœuds virtuels. Graphiquement HOOD ajoute la représentation des flots de données entre objets ainsi que les exceptions -cas exceptionnels lors de retours d'opérations non exécutées normalement jusqu'au bout-.
Modélisation fonctionnelle	*Modélisation fonctionnelle : l'aspect fonctionnel exprime la modélisation de l'activité : "ce que l'on fait". Les méthodes axées essentiellement sur une telle approche sont appelées orientées opérations.*
L'aspect fonctionnel est exprimé à travers les MCT et MOT. Au niveau du MCT, le concept d'opération modélise l'activité du domaine en échange avec les sollicitation de son environnement. Au niveau du MOT, le concept de phase ou de tâche(selon le degré de détail) modélise l'activité élémentaire exercée au sein du domaine (homogénéité de l'activité, unité de lieu, unité de temps, homogénéité de ressources, ...)	Du point de vue fonctionnel HOOD relie chaque opération avec une ou plusieurs opérations du niveau de décomposition inférieur. Lorsque le lien est de cardinalité 1-n le lien est effectué par un objet de contrôle. Ce lien est désigné par la relation "implemented_by".
Modélisation structurelle	*Modélisation structurelle : l'aspect structurel permet de décrire "statiquement" les objets modélisés; "ce dont on parle, sur quoi s'applique les traitements". Bien souvent, les méthodes axées sur une telle approche ont été appelées orientées données.*
L'aspect structurel se traduit essentiellement sur la modélisation des données à travers les MCD, MOD, MLD. Au niveau du MCD et MOD, la méthode MERISE utilise le formalisme entité relation. Au niveau du MLD, elle utilise les formalismes relationnel ou Codasyl.	Le modèle structurel ou statique de HOOD est l'objet qui dénote en fait une structure de module qui sera ensuite codée au niveau physique par une unité syntaxique -si le langage cible le permet- de compilation. La description des données se fait par l'intermédiaire de descripteurs de types suivant une syntaxe simplifiée similaire à celle de Ada ou Pascal.
Modélisation temporelle ou événementielle	*Modélisation temporelle ou événementielle : l'aspect événementiel permet de décrire le comportement des objets modélisés ou le fonctionnement et de décrire l'évolution dans le temps des objets modélisés ou du fonctionnement :"quand et pourquoi".*
L'aspect événementiel est pris en compte dans les modèle de traitement (MCT et MOT) par le concept d'événement/ résultat qui modélise le stimulus déclencheur d'une activité ou sa réaction. La prise ne compte du temps est également présente dans le MOT à travers la notion de délai de réponse d'un tâche (immédiate ou différée), ainsi que dans le MLT à travers de la notion de transactionnel (ou "temps réel") ou batch.	Un objet actif est un objet comportant au moins une opération contrainte. Une opération contrainte possède une désignation de contrainte qui exprime le protocole d'appel de l'opération (synchrone, asynchrone, avec délais...). A chaque objet actif est associé obligatoirement une description du comportement dynamique de l'objet via les opérations contraintes. Ceci est fait dans la partie textuelle de description de l'objet appelée "structure de contrôle de l'objet" OBCS. Le formalisme n'est pas imposé par HOOD. On peut utiliser du pseudo-code Ada ou bien d'autres formalismes - diagrammes d'états-finis, réseau de pétri, langage réactif tels que Esterel ou Electre.

5 - Axe mise en oeuvre	
Organisation	*Quelle est l'organisation du travail préconisée par la méthode pour sa mise en oeuvre?*
La méthode MERISE propose d'abord une distinction entre maîtrise d'ouvrage et maîtrise d'oeuvre. Elle préconise ensuite une organisation en trois groupe aux rôles distincts : - le groupe de projet chargé de la réalisation opérationnelle du projet, de l'étude à la réalisation - le groupe de pilotage responsable des objectifs et des choix majeurs sur le projet - le groupe de validation chargé de donner son avis sur les propositions du groupe de projet Les rôles et composition de ces groupes sont évolutifs au cours du déroulement du projet.	La méthode HOOD est particulièrement adaptée pour les gros projets comportant plusieurs équipes. Chaque équipe traite un objet. Chaque objet est décomposé et les objets sont à nouveau attribués pour à la fin avoir un - ou plusieurs- objet par ingénieur. La sous-traitance est facilitée par le fait des spécifications externes d'un objet. Les utilisateurs d'un objet n'ont pas à connaître les détails internes de l'objet. Tant que l'interface externe d'un objet interne n'est pas remise en cause chaque équipe peut prendre des décisions internes sans répercussion sur les utilisateurs des objets qu'ils développent.
Normes	*Existe-t-il des normes de définition de la méthode, de mise en oeuvre, de présentation de la documentation..*
Bien que largement diffusée, la méthode MERISE ne dispose pas de norme officielle; cette absence fait l'objet de nombreuses critiques de la part des utilisateurs du fait de trop nombreuses "variantes" et adaptations. Il existe toutefois un consensus sur de nombreux éléments de cette méthode mais cet accord informel reste très fragile et sensible aux effets commerciaux. Une tentative de normalisation serait en cours...	Bien que non normalisée par des instances nationales ou internationales la méthode HOOD suit un processus contrôlé par les représentants des utilisateurs de la méthode. Actuellement la méthode est à la version 3.1. Le manuel de référence est distribué en librairies (version française et anglaise). Comme la méthode définit la syntaxe à utiliser pour représenter la documentation correspondante au dossier de conception logicielle, avec des éditeurs spécialisés incorporés dans les outils supportant la méthode, tous les utilisateurs de la méthode sont assurés d'une même représentation. Ce qui permet une meilleure communication est une meilleure portabilité de l'information ainsi que des hommes.
Qualité	*Qualité : par quelles procédures la méthode assure une qualité des spécifications produites.*
L'assurance qualité n'a été introduite qu'à posteriori dans la méthode MERISE, assez naturellement dans les étapes de logiciel, plus difficilement dans les étapes d'étude. L'assurance qualité en études se traduit globalement dans MERISE par : - des relectures de modèle au sein du groupe de projet et auprès du groupe de validation - des règles formelles de confrontation données/traitements Ces activités ont partie intégrante de la démarche MERISE	La méthode HOOD provenant d'un secteur industriel dans lequel l'application d'une démarche qualité rigoureuse est coutumière, comporte des règles de contrôle et de validation des étapes de conception. Ainsi un objet parent doit être validé au niveau petit-fils. Les contrôles ou les inspections de conception s'effectuent par des lectures croisées entre les membres de l'équipe ou si possible par des lecteurs externes expérimentés.
Conduite de projet	*La méthode intègre-t-elle des procédures et techniques de gestion de projet ?*
L'aspect conduite de projet(estimation, planification, affectation des ressources, suivi d'avancement, analyse des écarts, révision) n'est pas très fortement développé dans MERISE. De nombreux utilisateurs de MERISE ont réalisé le couplage de MERISE avec des méthode de conduite de projet, en particulier la méthode SDM/S.	Hormis le fait que la méthode HOOD est supportée par des ateliers de génie logiciel, l'activité de conception est vue comme une phase au niveau d'un projet. La fourniture attendue en fin de conception est un dossier HOOD de conception et une maquette de squelette du code exprimant l'architecture logicielle codée. La fin de l'application de la méthode HOOD se traduit par une revue dite revue de la phase de conception logicielle.