for Object Identification".Procc.of the 24th annual Hawaii Intern. Conf. on Syst. Sc., Hawaii, 1991, pp. 205-218.
[Jac 92] Jacobson I. et al. "Object-Oriented Software Engineering : A Use Case Driven Approach". Addison-Wesley, 1992.
[Kin 89]Kin R., "My Cat is Object-Oriented ". In Object-Oriented Concepts and Databases. Kim & Lochovsky (eds.), Addison-Wesley, 1989.
[Kor 90]Korson J., McGregor J-D., "Object-Oriented Design". CACM, Vol.33, N 9, Sept. 90, pp.39-60.
[Lee 91]Lee S., Carver D-L., "Object-Oriented Analysis and Specification: a knowledge base approach ". JOOP, Jan. 1991, pp.35-43.
[Lie 88]Lieberherr K., Holland I., Riel A., "Object-Oriented Programming : An Objective Sense of Style". Proc.of OOPSLA'88 , San Diego (Calif.), Sept. 1988, pp.323-334.
[Mar 92] Martin J., Odell J., "Object-oriented analysis and design". Prentice-Hall, Inc., EnglewoodCliffs, NJ, 1992
[Mon 92] Monarchi D-E., Puhr G-I., "A Research Typology for Object-Oriented Analysis and Design". CACM, Vol.35, N 9, Sept. 92, pp.111-123.
[Roc 92]Rochfeld A., "Les méthodes de conception orientées objet". Congrès INFORSID. Clermont-Ferrand, Mai 1992, pp.563-593.
[Roc 93] Rochfeld A., Bouzeghoub M., "From Merise to OOM". Revue ISI, Vol. 1, n 2, Paris, 1993.
[Rol 91]Rolland C., Cauvet C., "Modélisation conceptuelle orientée objet". VIIèmes Journées BDA, Lyon, 25-27 Sept. 1991, pp. 299-325.
[Rub 92]Rubin K-S., Goldberg A., "Object Behavior Analysis". CACM, Vol.35, N 9, Sept. 92, pp.48-62.
[Rum 91] Rumbaugh J. et al., "Object-Oriented Modeling and Design". Prentice-Hall, 1991.
[Shl 88]Shlaer S, Mellor S-J., "Object-Oriented System Analysis : Modeling the World in Data". Prentice Hall, Englewood Cliffs, N.J., 1988.
[Shl 92]Shlaer S, Mellor S-J., "Object Lifecycles : Modeling the World in States". Yourdon Press, Englewood Cliffs, N.J., 1992.
[Was 90] Wasserman, A-I., Pircher P-A., Muller R-J., " The Object-Oriented Structured Design Notation for Software Design Representation". IEEE Computer, Mar. 90, pp.50-62.
[Wir 89] Wirfs-Brock R., Wilkerson B., "Object-Oriented Design : A Responsability-Driven Approach". Proc. of OOPSLA'89, New Orleans (La.), Oct. 1989, pp.71-75.

sition d'Etats (STDs),
- le modèle fonctionnel est proche des Diagrammes des Flux de Données (DFDs).

Son point faible concerne certainement l'intégration. En effet, comment faire correspondre le modèle DFD + Evénement + Activité + STD au comportement d'un objet.

2) Approche adaptative

Elle utilise les approches existantes en y incluant l'orientation objet. On étend une méthode traditionnelle par des fonctionnalités du paradigme objet, c'est le cas, en particulier, de la méthode OOM.

3) Approche orientée objet pure

Le paradigme objet est utilisé de manière uniforme depuis la phase d'analyse jusqu'à la réalisation. On peut citer à ce niveau la méthode MCO et la méthode O*.

On remarque, cependant, que toutes les méthodes préconisent la réutilisation, mais aucune méthode n'ose reprendre des parties bien développées par d'autres méthodes et ajouter uniquement "le plus" qu'elle propose. En effet, chaque méthode développe sa propre démarche, en ignorant totalement l'existant méthodologique composé de méthodes et modèles couvrant, à des degrés divers, différentes phases du cycle de développement des applications.

Nous terminons enfin cette présentation en signalant que les méthodes du futur doivent intégrer nécessairement les outils qui permettraient de prendre en compte un existant informatique, représenté par des schémas (souvent Entité-Relation) ou du code. En effet, face à la nouvelle donne économique nous pensons que l'ère des grands projets est révolue. L'industrie informatique (celle du logiciel en particulier) doit, en effet, prendre exemple sur les autres secteurs industriels et apprendre à développer des composant (kits) logicels. Le concepteur aura alors à sa disposition une "boite à outils" logicielle constituée d'un ensemble de composants logiciels lui permettant de bâtir "sur mesure" son système.

BIBLIOGRAPHIE

[Ala 88]Alabiso B., "Transformation of Data Flow Analysis Models to Object-Oriented Design". Proc. of OOPSLA'88, San Diego (Cal.), Sept. 1988, pp. 335-353.

[Bai 89]Bailin S-C., "An Object-Oriented Requirements Specification Method". CACM, Vol.32, N 5, May 1989, pp.608-623.

[Boo 91]Booch G., "Object-Oriented Design with Applications". Benjamin Cummings, 1991.

[Cas 91] Castellani X., "Le modèle de la méthode MCO d'analyse et de conception des systèmes d'objets". Congrès INFORSID, Paris, 1991.

[Cas 93]Castellani X., "MCO : Méthodologie d'analyse et de conception des systèmes à objets. Tome 1: Ingénierie des besoins". Masson, Paris, 1993.

[Cau 91]Cauvet C., Rolland C., "Conception d'un système d'information: une approche orientée objet". Journées AFCET : Autour et à l'entour de Merise. Sophia Antipolis, Avril 1991.

[Cha 92a] Champeaux (de) D., Lea D., Faure P., "The Process of Object-Oriented Design". Proc. of OOPSLA'92, Vancouver (Can.), Oct. 1992, pp.45-62.

[Cha 92b] Champeaux (de) D., Faure P., "A Comparative study of Object-Oriented analysis Methods". Journal of Object-Oriented Programming, March/April 92, pp.21-33.

[Coa 90]Coad P., Youdon E., "Object-Oriented Analysis". Prentice-Hall, 2nd Edition, 1990.

[Cun 86] Cunnigham W., Beck K., "A diagram for Object-Oriented Programs". Proc. Of OOPSLA'86. Portland (Oregon), Sept. 1986.pp.39-43.

[Hen 90] Henderson-sellers B., Edouards J-M., "The Object-Oriented Life Cycle". CACM. Vol.33, N 9, Sept. 90. pp.141-159.

[Hon 93]Hong S., Goor G., Brinkkemper S., "A formal approach to the comparison of object-oriented analysis and design methodologies". 26th Hawaii Intern. Conf. on Syst. Sc., Hawaii, 1993, pp.689-698.

[Iiv 91] Iivari J., "Object-Oriented Information System Analysis: A Framework

suite repris et raffiné après la spécification des modèles dynamique et fonctionnel.

OMT, comme la plupart des autres méthodes, développe selon un processus plus itératif qu'incrémental.

4.4. Réutilisation

La réutilisation est utilisée ici au sens de réutilisation de composants logiciels (applicatifs). Tout comme pour le développement incrémental, on souligne les avantages de la réutilisation mais peu de méthodes en font cas explicitement. Signalons toutefois 3 méthodes qui intègrent la réutilisation comme principe de développement.

- OOD gère une bibliothèque de classes contenant les classes prédéfinies et les composants spécifiques (du domaine) réutilisables.
- OOSE construit des applications à partir de deux types de composants logiciels : standards (génériques) et spécifiques (liés à l'application). Des règles de développement et de sélection de composants sont définies.
- MCO intègre la réutilisation à différentes étapes du cycle de vie d'un système d'objets (conception, évolution, etc.). Des gabarits sont définis pour spécifier les caractéristiques communes et particulières réutilisables.

4.4. Gestion de la complexité

La gestion de la complexité est un mécanisme de regroupement conceptuel d'entités d'un système en vue de maîtriser sa complexité (taille, graphe de communication, etc.). Cinq méthodes supportent de tels mécanismes :

1) - MCO utilise le mécanisme de système d'objets, d'objets types et d'agents pour réduire la complexité du système en le décomposant en sous-systèmes et en opérant des regroupements d'objets en objets types, etc.

2) - O* utilise le mécanisme d'abstraction pour la réduction de la structure en regroupant les classes composites avec leurs classes composantes et les calsses généralisées avec leurs classes spécialisées.

1)- La méthode OOA de Coad-Yourdon utilise le mécanisme de sujet (subject) pour encapsuler les objets dans des structures arborescentes de généralisation/spécialisation [Coa 90].

2)- La méthode de Wirfs-Brock utilise le mécanisme des sous-systèmes. Un sous-système consiste en un ensemble de classes ou d'autres sous-systèmes qui collaborent en vue d'atteindre un objectif donné (une fonction) [Wir 90].

3)- La méthode OMT utilise la notion de module comme mécanisme de regroupement des classes et des relations [Rum 91].

Le processus de gestion de la complexité couvre les phases suivantes :

- gestion de la complexité des structures
- gestion de la complexité du comportement
- représentation (statique et dynamique).

5 CONCLUSION

Les méthodes d'analyse et de conception orientées objet sont très récentes et manquent par conséquent de maturité. Elles souffrent naturellement des erreurs de jeunesse. Il est, en particulier, difficile de dégager un standard de fonctionnalités que doit offrir obligatoirement toute méthode de conception. Ceci explique en partie la difficulté des études comparatives sur les méthodes actuelles.

On distingue trois approches suivies par les méthodes d'analyse et de conception orientées objet. Celles-ci sont caractérisées par la façon et le degré d'intégration des divers paradigmes utilisés :

1) Approche mixte ou "combinative"

Avec cette approche, on modélise séparément la structure, les fonctionnalités et / ou le comportement du système avec des modèles orientés objet, fonction et dynamique.

Il est nécessaire de définir ensuite une méthode d'intégration comme dans la méthode OMT. En effet OMT utilise les processus et la représentation pour la modélisation structurelle, dynamique et fonctionnelle.

- la représentation de la structure est dérivée des diagrammes Entité-Relation (ERDs),
- le modèle de la dynamique est constitué par les Diagrammes de Tran-

C) Processus de représentation

La représentation consiste à donner une expression graphique des produits de l'analyse et/ou de conception pour permettre une communication aisée. Les modèles de représentation sont nombreux et les conventions utilisées varient beaucoup d'un modèle à un autre. Ce sont souvent des modèles statiques. Certains modèles intègrent aussi l'aspect dynamique, ainsi que la prise en compte des contraintes. La démarche consiste alors à représenter les trois aspects.

a) Vue statique : objets, attributs, comportement et relations (généralisation, agrégation, association).
b) Vue dynamique : communication et contrôle.
c) Contraintes : de structure (valeurs d'attributs, cardinalités de relations) et de comportement.

Les 3 méthodes suivantes s'attachent uniquement à l'aspect représentation :
- Message Diagrams de Cunningham et Beck [Cun 86],
- Object-Oriented Structured Design Notation (OOSD) de Wasserman et al. [Was 90],
- Class Diagrams de Wilson [Wil 90].

Il est difficile de classifier les méthodes par rapport aux processus d'analyse et de conception, tant la frontière entre les deux est difficile à fixer. Ainsi, la méthode OOA dite d'analyse couvre aussi les aspects conception et implémentation alors que la méthode OOD, reputée orientée conception, couvre aussi les phases d'analyse. Nous regroupons alors les deux aspects spécification et représentation.

4.1.2 Spécification et représentation

La plupart des méthodes orientées objets englobent les procédures d'analyse et de conception (aspect processus), ainsi que l'expression graphique des résultats (aspect représentation). C'est sans doute la catégorie la plus importante. On y distingue les méthodes suivantes :
- "Object-Oriented Software Engineering " de Jacobson [Jac 92],
- "Transformation from Analysis to Design (TAD)" de Alabiso [Ala 88],
- "Object-Oriented Requirements Specification Method" de Bailin [Bai 89],
- "OOD " de Booch [Boo 91],
- "MCO" de Castellani [Cas 91], [Cas 93],
- "OOA& OOD " de Coad-Yourdon [Coa 90]
- "A Framework for Object Identification" de Iivari [Iiv 91],
- "OOM " de Rochfeld [Roc 93],
- "O*" de Rolland et al. [Rol 91],
- "OMT" de Rumbaugh et al.[Rum 91],
- "OOSA" de Shlaer et Mellor [Shl 88],
- "Designing Object-Oriented Software" de Wirfs-Brock et al. [Wir 90].

4.2 Développement incrémental

Le principe de développement incrémental est cité comme étant un apport fondamental de l'objet. On constate cependant que le cycle de développement des méthodes objet suit rarement un processus incrémental.

Deux méthodes préconisent explicitement un développement incrémental : OOD et OOSE.

- OOD commence par construire une première hiérarchie de classes qui sera ensuite révisée en appliquant les opérations d'abstraction, d'agrégation, de dérivation, etc. Chaque étape intermédiaire produit un prototype représentant une version du système.
- OOSE spécifie dès la première phase le scénario (use case) global du système. Le processus de modélisation consiste alors à détailler cette spécification selon plusieurs niveaux de granularité.
- La méthode OMT a un cycle de développement qui se rapproche davantage du cycle en cascade avec retour arrière vers l'étape précédente. Le développement débute toujours par la spécification du modèle statique qui sera en-

des méthodologies et des techniques de développement orientées objets. On trouve le plus souvent une énumération des caractéristiques souhaitables des méthodologies orientées objets. Ceci s'explique par le fait que la plupart des travaux sur les systèmes orientés objet sont essentiellement orientés vers les langages de programmation ou vers les concepts généraux de l'approche objet. C'est essentiellement à partir de 1990 que des travaux sont entrepris sur les méthodes et les représentations des systèmes de développement orientés objet.

On recense actuellement trois propositions pour la classification des méthodes orientées objet :

1 – la classification proposée par De Champeaux et Faure [Cha 92b] est basée sur des caractéristiques peu discriminantes. Elle se limite à présenter les caractéristiques objet supportées par les différentes méthodes. Elle ne met pas en évidence la pertinence et le degré de couverture des différentes étapes du cycle de développement.

2 – la classification proposée par Hong [Hon 93] est fondée sur le développement d'un méta-modèle pour le processus de conception et les données du système. La comparaison distingue alors les catégories suivantes :

- Les étapes d'analyse et de conception,
- Les concepts,
- Les techniques utilisées.

3 – la typologie des méthodes proposée par Monarchi et Puhr [Mon 92] nous semble plus complète car elle utilise une grille d'analyse assez étoffée tout en situant celle-ci par rapport au cycle de développement. Elle distingue trois catégories de méthodes en fonction de l'aspect supporté :

- représentation uniquement,
- conception seulement,
- représentation et conception.

Les critères que nous adoptons sont plus généraux et recouvrent ceux utilisés par ces classifications.

4.1 Modélisation

La modélisation comporte le processus de spécification (analyse et conception) et le processus de représentation des resultats de la spécification.

4.1.1 Processus de spécification

On entend par processus, l'aspect procédural de l'analyse et /ou de la conception. Ceci n'inclut pas l'aspect représentation (diagrammes). Une seule méthode couvre spécifiquement cet aspect : méthode de Henderson-Sellers et Constantine [Hen 90].

A) Processus d'analyse

Cette phase consiste à rechercher les entités (classes/objets) de l'espace du problème (i.e. le quoi). L'identification est un processus informel; chaque méthode propose ses propres heuristiques, fondées le plus souvent, sur le bon sens. Les entités identifiées sont ensuite placées de façon judicieuse. Ceci suppose donc un ensemble de règles permettant de mesurer la validité des placements effectués. Un travail intéressant, connu sous le nom de Loi de Demeter, a été fait dans ce sens par Lieberherr [Lie 88]. La loi de Demeter constitue une règle de bonne conduite dans la construction du graphe de collaboration des classes.

B) Processus de conception

L'étape de conception consiste en un raffinement des entités identifiées au niveau de l'analyse, plus d'autres entités déterminées durant la conception. On les décompose ensuite en entités de l'espace des solutions. On détermine ainsi l'aspect technique du problème (i.e. le comment). Cette phase doit aussi s'assurer que la spécification est complète.
La démarche du processus se réduit à deux étapes :

a) Identification des classes d'application, d'interface et utilitaires,

b) Optimisation des classes.

- définir les objets qui présente un certain comportement, qui offrent un service,

- définir le modèle objet en classifiant les objets identifiés ainsi que leurs relations,

- définir le modèle dynamique en précisant le cycle de vie des objets et le séquencement des opérations.

Cette méthode offre un avantage par rapport aux autres méthodes, en permettant, en particulier, l'identification des objets du système.

3.11 Méthode OOSA de Shlaer& Mellor

Shlaer et Mellor sont parmi les premiers à publier un ouvrage sur l'analyse et la conception orientées objet. Le modèle OOSA (Object-Oriented System Analysis) de Shlaer & Mellor [Shl 88] est historiquement l'un des premiers modèles orientés objet. Celui-ci est en fait une extension, par l'héritage, du modèle E-R. Cependant, le modèle OOSA reste ambigu sur plusieurs points, particulièrement ceux concernant les aspects structuration, visibilité, etc. Le modèle retenu utilise un nombre très restreint de concepts, en particulier, les notions de cardinalité et de rôle sont absentes. Les traitements sont modélisés en utilisant les diagrammes de flots de données (DFD) et les automates d'états finis. Ce modèle a servi de base au développement d'autres modèles orientés objet, en particulier le modèle OOA de Coad et Yourdon. Une extension récente du modèle est réalisée en vue de prendre en compte les aspects dynamiques qui manquaient dans OOSA [Shl 92].

3.12 Méthode OOAD

La méthode OOAD (Object-Oriented Analysis and Design) de Martin et Odell [Mar 92], est fondée sur la logique et la théorie des ensembles. Les fondements mathématiques de la méthode permettent des manipulations élégantes d'expression des classes, de dérivations d'associations et de migration d'objets. La méthode est essentiellement orientée vers la description du comportement des objets ainsi que leur identification avec leurs relations.

4 Caractérisation des méthodes orientées objet

Bien que la technologie objet soit maintenant largement adoptée dans les méthodes et techniques de développement des systèmes d'information, il n'y a cependant toujours pas de consensus sur les caractéristiques essentielles de ces méthodes. Toute tentative d'évaluation et de classification se heurte alors à une double difficulté :

1) - Les caractéristiques essentielles que doit vérifier toute méthode orientée objet doivent être définies au préalable, pour servir de critères d'évaluation et de classification.

2) - Les concepts utilisés doivent être clarifiés. En effet les mêmes termes ne désignent pas toujours les mêmes concepts. Un effort d'unification de la terminologie est nécessaire. Les concepts et les termes utilisés par les méthodes présentées au paragraphe précédent illustrent cette diversité.

L'analyse des méthodes orientées objet permet de faire deux observations :

a) - On constate au premier abord qu'il existe deux extrêmes en ce qui concerne les approches :

- D'une part, il y a celles que l'on pourrait qualifier de "puristes", c'est-à-dire les méthodes de développement totalement orientées objet, englobant les phases d'analyse et de conception et / ou d'implémentation. Elles développent des approches et des représentations qui diffèrent totalement des traditionnelles méthodes de modélisation orientées données ou orientées traitements.

- D'autre part, il y a les approches que l'on présente comme "orientées objet" mais qui ne sont en réalité que de nouveaux termes pour désigner de vieux concepts (voir à ce sujet l'article de King "my cat is object-oriented" [Kin 89]).

- Entre ces deux extrêmes, on trouve toute une panoplie de méthodes couvrant, à des degrés divers, les différentes phases du cycle de développement.

b)- Il existe très peu de travaux sur l'évaluation et la classification

d'une phase est repris puis affiné dans la phase suivante. Le modèle Classe-relation repose en fait sur trois modèles complémentaires :
- le modèle à objet ou modèle structurel est bâti sur le modèle E-R étendu. Ce modèle permet de décrire les notions de classes, de les structurer et de définir leurs propriétés (attributs et méthodes);
- le modèle opératoire permet de spécifier les pré- et post-conditions. Les séquences d'activation des méthodes sont spécifiées par les diagrammes Etats-transitions;
- le modèle dynamique définit les automates de déclenchement en utilisant les diagrammes Etats-transitions. Ce modèle permet aussi de décomposer les traitements en processus, en utilisant les diagrammes Object flow.

Figure 4: Composantes des modèles Classe-relation

q Étapes de développement

Le cycle de développement de la méthode Classe-Relation se déduit des 3 modèles sous-jacents:

- Construire le modèle structurel en définissant les classes avec leurs liens ainsi que les invariants.
- Construire le modèle opératoire en définissant les règles de contrôle ainsi que les pré et post conditions.
- Construire le modèle dynamique en spécifiant les automates de déclenchement ainsi que les flots de données.

Avant de clore ce paragraphe sur la présentation des méthodes, signalons quatre autres méthodes, aux caractéristiques intéressantes et qui diffèrent totalement quant à l'approche adoptée jusqu'ici. Les autres méthodes seront mentionnées chaque fois qu'elles répondent de façon spécifique à un critère de classification.

3.9 Méthode RDD

La méthode de Wirfs-Brock, appelée RDD (Responsability Driven Design), ou encore CRC (Classes, Responsabilités et Collaborations) est basée sur le concept de conception dirigée par la responsabilité [Wir 89]. A l'inverse des autres méthodes, qui reposent sur une modélisation dirigée par les données, la méthode RDD de Wirfs-Brock décrit un système en termes de responsabilités assurées par les classes. Ce principe permet de spécifier un système en termes de modèle client/serveur. En effet, le modèle client/serveur spécifie les interactions entre deux entités : le client et le serveur. Un client envoie des requêtes au serveur, pour exécuter des services. Un serveur offre un ensemble de services en réponse aux requêtes qui lui sont adressées. Le protocole d'interaction entre le client et le serveur est spécifié par un contrat, décrivant une liste de requêtes qu'un client peut formuler à un serveur. L'intérêt du modèle client/serveur réside dans le fait qu'un système est vu à travers ce qu'il fait (le quoi) et non comment il fait. Ainsi, dans cette approche, l'encapsulation est totale, contrairement à l'approche dirigée par les données où l'encapsulation est violée car la structure de l'objet fait partie de sa définition (les opérations sont décrites au vu de la structure). Les clients d'une classe sont de trois types : les clients externes, les sous-classes et la classe elle-même.
Le cycle de développement consiste à définir un contrat, grâce aux deux questions suivantes :

- Quelles sont les actions dont un objet est responsable ?
- Quelles sont les informations partagées ?

3.10 Méthode OBA

La méthode OBA, Object Behavior Analysis de Rubin et Goldberg [Rub 92], se distingue des autres méthodes par le fait qu'elle s'intéresse au comportement du système pour déterminer les acteurs qui l'initialisent et ceux qui participent. Ces deux types d'acteurs permettent de comprendre le rôle des différents aspects d'un système. La méthode OBA utilise les cinq étapes suivantes :

- définir le contexte d'analyse,
- expliciter le problème en spécifiant le comportement du système,

- identifier les activités du système,
- définir le modèle conceptuel des objets,
- définir le modèle conceptuel des traitements
- définir l'architecture du système.

Les étapes suivantes du cycle de développement consistent en un raffinement de cette architecture (architecture logique puis physique, etc.).

3.7 La méthode O*

Le modèle O* est un prototype réalisé par l'équipe de Rolland [Rol 91], [Bru 91].

Dans le modèle O*, un objet est défini selon trois perspectives : la perspective statique, la perspective dynamique et la perspective comportement [Rol 91], [Cau 91].

La perspective statique permet de spécifier les propriétés spatiales (attributs) et temporelles (événements) d'un objet.

La perspective dynamique tente de définir les règles selon lesquelles un objet évolue. Deux types de règles décrivent la dynamique d'un objet :

- les opérations qui modifient les valeurs d'attributs
- les occurrences d'événements qui déterminent les conditions pour lesquelles les événements peuvent se produire dans le cycle de vie d'un objet.

La perspective comportement considère que les objets sont actifs. Un objet réagit à certains changements d'état en déclenchant des opérations sur des objets ou en produisant d'autres événements.

q Schéma objet, cycle de vie

Un schéma objet spécifie la structure et le comportement d'une classe d'objets, caractérisé par un quintuplet : Sch (A) = (Pro, Ope, Cns, Ref, Evt).

Pro = ensemble des propriétés de structure,
Ope = ensemble des opérations applicables,
Cns = contraintes statiques ou dynamiques,
Ref = ensemble des références,
Evt = ensemble des événements.

Les changements d'états d'un objet, dus adéclenchement d'opérations un événement, caractérisent le cycle de vie de l'objet, représenté par l'ensemble des événements t (o) = {Evi (o), i =1,.n}. Cet ensemble est totalement ordonné car les événements sont datés.

q Héritage

Un classe S spécialise une classe générique G ssi les deux axiomes suivants sont vérifiés :

- axiome d'inclusion de schémas, c'est-à-dire Sch(G) Ã Sch(S) S hérite des propriétés de G (références, contraintes, opérations et événements).
- axiome d'inclusion des extensions, c'est-à-dire Ext(S) Ã Ext(G). Cela signifie que toute instance de S est aussi instance de G.

q Abstraction

L'abstraction est utilisée comme mécanisme de réduction de la complexité. Le regroupement des classes est fait sur la base des 2 heuristiques :

- les classes composites peuvent être regroupées avec leurs classes composantes.
- Les classes généralisées peuvent être regroupées avec leurs classes spécialisées.

q Étapes de développement

On peut réduire le cycle de développement de O* à 3 étapes principales :

- construire le schéma objet,
- réduire par abstraction le schéma obtenu,
- définir le comportement des objets.

3.8 Méthode Classe-Relation

La méthode Classe-relation [Des 92] couvre tout le cycle de développement d'un produit, depuis la phase de spécification des besoins jusqu'à la phase de production du code. La méthode adopte une approche descendante, le résultat

- sélectionner les objets pertinents.

2) - Modélisation par les sous-systèmes
   - décomposer les système en sous-systèmes,
   - identifier les agents externes.

3) - modélisation par les types et dépendances
   - définir tous les types,
   - identifier les dépendances,
   - construire les objets.

4) - modélisation par les services
   - définir les services demandés,
   - regrouper les services en objets,
   - associer les connaissances utilisées.

3.6 Méthode OOM

La méthode OOM (Orientation Objet de Merise) est réalisée par Rochfeld et Bouzeghoub comme une extension objet de la méthode Merise. Le modèle de OOM est proche du modèle OMT. En effet, OOM utilise un référentiel à trois dimensions : le modèle statique décrivant le diagramme des objets ainsi que le diagramme E-R, le modèle dynamique décrivant le modèle des traitements avec les contraintes associées, et enfin le modèle fonctionnel spécifiant le diagramme des flux de données [Roc 92].

q Modèle statique

Le modèle statique (de données) de OOM étend celui de Merise par les notions suivantes :

- le mécanisme de généralisation/spécialisation,
- les propriétés multivaluées pouvant prendre plusieurs valeurs distinctes pour une même occurrence.
- les propriétés et entités composées (décrites par une autre propriété ou entité).

q Modèle des traitements

On retrouve dans OOM tout l'héritage du modèle de Merise : événement, synchronisation et opération. Ceci est complété par les notions de phase-type, de graphe de structuration des opérations et de cycle de vie des entités.

Dans le modèle conceptuel structuré des traitements, une opération est décomposée en phases. Un graphe de gestion (graphe de calcul) associe à chaque phase les propriétés des entités ou des relations au travers de la règle de gestion associée.

Le découpage en phases-type obéit à la logique suivante : toute opération ou activité est décomposée en un nombre quelconque d'occurrences de phases-type, chacune matérialisant une étape significative du traitement. Certaines phases peuvent être conditionnelles, d'autres liées à des phases antérieures. On distingue les phases-type suivantes :

- acquisition et création de l'information,
- édition et restitution des résultats,
- contrôle des informations,
- présentation et traitement des erreurs.

q Modèle fonctionnel

Le modèle fonctionnel permet de décrire la dimension fonctionnelle d'un système. Il est basé sur les notions de diagrammes de flux de données et de décomposition fonctionnelle. On le retrouve ainsi au niveau conceptuel sous la forme d'un diagramme d'activités, au niveau organisationnel sous forme d'un diagramme d'acteurs et enfin au niveau physique sous la forme d'un diagramme de modules.

Figure 3: Relation entre objet et modèles de OOM

q Étapes de développement

La démarche de modélisation de OOM peut être résumée par les 4 étapes suivantes :

capsulés. Un objet est caractérisé par ses connaissances, ses services et ses transmutations.
Une classe est définie par un objet type et par l'ensemble des instances de cet objet type.

q Administrateurs, instances et serveurs d'objets

L'administrateur d'un objet type, classe OB–a est une instance d'un objet type OB–b :

– qui connaît le type de OB–a, toutes ses instances, et éventuellement les autres connaissances collectives de OB–b (son type, les services et les transmutations).

– et qui assure des services collectifs relatifs à OB–a (instanciation, suppression d'instances de OB–a, gestion d'interfaces de sélection de services collectifs).

Les instances d'un objet d'une classe ont des connaissances individuelles et assurent des services individuels. Les caractéristiques communes aux instances d'une classe sont des connaissances dont les types et les valeurs sont identiques à toutes les instances.

Un serveur d'un objet type (classe) est constitué par l'administrateur de cet objet, de l'objet type et de la classe (i.e. extension de la classe).

q Objets sémantiques, de servitude, de construction

Les objets sémantiques (objets utilisateurs) représentent des entités du monde réel.
Les objets de servitude (objets techniques) sont les ressources nécessaires pour rendre les objets sémantiques utilisables. Les objets de construction sont définis dans le seul but de factoriser des caractéristiques communes à plusieurs objets.

q Transmutations

La transmutation est le mécanisme de transformation d'instances. La transmutation d'un objet OB–a en instance(s) d'un ou plusieurs objets OB–b, OB–c,..., dont l'un peut être OB–a, correspond à sa suppression en tant qu'instance de Ob–a et sa création en tant qu'instance(s) des objets OB–i.

q Héritages

MCO distingue deux catégories d'héritage : les héritages de spécialisation et les héritages de construction. Les héritages de spécialisation représentent les liens "est-un" au moyen desquels sont définis des sous-types; les héritages de construction n'ont pas de signification sémantique, ils sont définis uniquement pour permettre la réutilisation des caractéristiques d'objets dans d'autres objets.
Le modèle supporte les héritages simple et multiple et autorise le masquage des connaissances et des services. Dans le cas d'héritage multiple, les conflits de noms sont résolus par renommage ou dénommage.

q Acteurs, systèmes

Un acteur est un agent intelligent car il est capable de modifier le type de ses connaissances et son comportement et ce, sans l'invention d'un autre agent. Il peut également décider de rendre et demander des services. Le concept d'acteur est proche de celui des langages d'acteurs, ainsi que du concept d'activités des bases de données actives.

Un système est un agent constitué d'agents de pilotage, d'agents opérant et/ou de système d'objets qui se demandent ou se rendent des services via un système de communication. Un système d'objets est constitué uniquement d'objets; ses demandeurs de services sont des acteurs, des objets ou des systèmes.

q Étapes de développement

MCO préconise plusieurs démarches de modélisation :

1) – Modélisation directe

– identifier les objets sémantiques,

q Contraintes

Les contraintes correspondent à des relations fonctionnelles qui existent entre les entités (objets, classes, attributs, liens et associations) du modèle statique (objet). Elles sont spécifiées de manière déclarative.

B) - Le modèle dynamique

Le modèle dynamique de OMT permet de représenter l'évolution du système dans le temps. L'évolution est décrite en termes d'événements et d'états. L'événement correspond à un stimuli externe du système et l'état correspond à la valeur des objets (valeur des composants).
Les événements et les états permettent de spécifier le contrôle d'un système sous forme de diagrammes d'états, représentant les transitions possibles en fonction des événements (modèle temporel).

OMT introduit aussi la notion d'activité. Celle-ci est associée à un état et représente une séquence logique d'actions indissociables, avec un début et une fin. La notion d'activité est à rapprocher de la notion de transaction.

C) - Le modèle fonctionnel

Le modèle fonctionnel permet de décrire les transformations des données (résultats des calculs) dans le système sans spécifier comment ni quand elles sont effectuées. Ce modèle spécifie le sens des opérations et des contraintes du modèle objet ainsi que les actions du modèle dynamique. Le modèle fonctionnel est généralement décrit par un ensemble de diagrammes de flots de données, où les noeuds représentent des processus et les arcs des flots.

q Étapes de développement

On distingue 4 étapes dans le cycle de développement de OMT:

1) - Construction du modèle statique (objet)
   - identifier les objets et classes,
   - construire le dictionnaire de données,
   - identifier les associations,
   - identifier les attributs des classes.
2) - Construction du modèle dynamique
   - identifier les événements,
   - construire le diagramme des transitions,
   - vérifier de la consistance.
3) - Construction du modèle fonctionnel
   - élaborer le diagramme des flux,
   - définir les contraintes,
   - définir les fonctions.
4) - Conception logique
   - organiser le système en sous-systèmes,
   - identifier les problèmes de concurrence.

3.5 Méthode MCO de Castellani

La méthode MCO couvre tout le cycle de développement d'un système d'information, en partant de la phase d'analyse, jusqu'à l'implémentation [Cas 91], [Cas 93]. Le modèle de base est bâti sur les concepts de service, agent, connaissance, objet, objet type, instance et classe, administrateur et serveur d'objet, transmutation, héritage, acteur et système.

q Services, agents et connaissances

Un service est une prestation demandée ou assurée par un agent. Un agent est une entité concrète ou abstraite possédant des connaissances. Les services qu'un agent peut rendre ou demander définissent son comportement. MCO distingue les agents intelligents (acteurs) et les agents non intelligents (objets). Une connaissance est une caractéristique qu'un agent peut utiliser. Les connaissances d'un agent définissent son savoir et caractérisent son état.

q Objets, objets types, instances, classes

Un objet est un agent dont les connaissances et les services sont en-

utilisant plusieurs processeurs.

Le diagramme de transition d'états décrit le comportement dynamique des classes, en exprimant les différents états et actions déclenchés par les événements.

Le diagramme d'ordonnancement temporel représente un autre aspect de la modélisation de la dynamique, en introduisant un ordre entre les messages exprimés dans le diagramme des objets.

q Étapes de développement

L'utilisation de la méthode de Booch pour le développement d'un système ne permet de couvrir que la phase dite de conception (au sens de Design). Toutefois la distinction entre analyse et conception n'est guère opérante tant la frontière est difficile à tracer; néanmoins le couplage avec une méthode couvrant la phase en amont de la conception est recommandé. Le cycle de développement débute par la mise en évidence des classes et des objets, à partir du vocabulaire du domaine d'application. Les principales Ètapes de la méthode peuvent être résumées ainsi :

- identifier les classes et les objets,
- identifier la sémantique des classes et des objets,
- identifier les relations entre les classes et les objets,
- implémenter les classes et les objets.

3.4 Méthode OMT de Rumbaugh

La méthode OMT (Object Modeling Technique) de Rumbaugh est une méthode orientée objet pour l'analyse et la conception des systèmes d'information [Rum 91]. Elle se distingue par le fait qu'elle est trËs largement adoptée et qu'elle couvre la totalité du cycle de développement.

OMT repose sur trois modèles différents : le modèle objet ou modèle statique, le modèle dynamique et enfin le modèle fonctionnel.

A) – Le modèle objet

Le modèle objet permet de décrire la structure statique du système en termes d'objets et de relations entre classes d'objets. La description est souvent un graphe Entité-Relation Etendu. Ce modèle permet de mettre en évidence les objets du système avec leurs propriétés ainsi que les opérations offertes. Le modèle est bâti sur les notions de classe d'objets, association, héritage, généralisation et contraintes.

q Objet / Classe

Une classe OMT regroupe l'ensemble des objets ayant les mêmes propriétés, i.e. les mêmes attributs, opérations et sémantique. OMT introduit une nouveauté concernant les notions de classe concrète et classe abstraite. Une classe concrète possède des instances directes alors que la classe abstraite n'en contient pas. Une classe abstraite sert à factoriser des propriétés entre classes non reliées par une hiérarchie d'héritage, elle ne possède donc pas d'instances.

q Liens / Associations / rôles

Un lien traduit une relation conceptuelle entre instances. Une association décrit un groupe de liens ayant même structure et même sémantique. Les associations expriment des relations bidirectionnelles entre classes. Elles sont spécifiées en termes de rôles, de qualification et d'agrégation.
Un rôle est relatif à une extrémité d'une association dont il précise la sémantique. Une association binaire aura donc deux rôles. L'utilisation des noms de rôles permet de se référer à deux objets en relation sans expliciter l'association. La qualification d'une association permet de réduire sa multiplicité; les associations de type 1-n et m-n peuvent être qualifiées.

q Module

Un module est un constructeur qui permet le regroupement de classes, d'associations et de généralisations. Un module représente une perspective ou une vue d'une situation. Le modèle statique (objet) d'un système est composé d'un ou plusieurs modules.

Figure 1: Les concepts de la méthode OOA

Le cycle de développement de la méthode OOA ne couvrait initialement que la phase d'analyse qui consiste à construire le modèle objet en identifiant les classes et les objets, les structures, les sujets, les attributs et les services. La dernière version de la méthode couvre aussi la phase de conception. L'aspect comportement est spécifié par des scénarios textuels et des diagrammes d'états.

q Étapes de développement

Le cycle de développement se déduit directement du modèle de base et comporte une phase d'analyse et une phase de conception :
La phase d'analyse consiste alors à construire le modèle objet en identifiant :

- les classes et les objets,
- les structures,
- les sujets,
- les attributs et les services,
- les scénarios et les diagrammes d'états.

La phase de conception consiste à affiner le modèle objet de la phase précédente et à prendre en compte les détails d'implémentation.

3.3 Méthode OOD de Booch

La méthode OOD de Grady Booch est l'une des méthodes les plus citées dans la littérature orientée objet. Cette méthode est orientée conception, au sens anglo-saxon (Design). Le modèle de la méthode est représenté par la figure 2 suivante.

Figure 2 : Le modèle de la méthode OOD de Booch

La conception d'un système avec la méthode de Booch est basée sur la construction de six types de diagrammes : le diagramme des classes, le diagramme des objets, le diagramme des modules, le diagramme des processus, le diagramme de transition d'états et enfin le diagramme d'ordonnancement temporel. Par rapport au modèle de la figure 2, l'aspect statique est décrit par les quatre premiers diagrammes, où la vue logique est décrite par le diagramme des classes et celui des objets, le diagramme des modules et le diagramme des processus représentent la vue physique. L'aspect dynamique est modélisé par les deux derniers diagrammes.
Le diagramme des classes est utilisé pour représenter les classes avec les relations qui existent entre elles, ainsi que les cardinalités correspondantes.
Le diagramme des objets met en évidence les échanges de messages entre les objets des classes.
Le diagramme des modules représente la répartition des classes et des objets entre les différents modules, lors de la conception physique du système. Le concept de module dans la méthode de Booch est proche de celui de "package" de ADA.

Le diagramme des processus est utilisé pour représenter l'affectation des processeurs aux processus, lors de la conception de grandes applications

1 - Analyse des besoins,
2 - Identifier les objets et les classes,
3 - Identifier les liens entre les classes,
4 - Définir les attributs,
5 - Spécifier la dynamique,
6 - Implémenter,
7 - Valider,

Ces phases servent à décrire les trois vues complémentaires d'un système : statique (i.e. structure), dynamique (i.e. comportement) et fonctionnel ou architectural (i.e. sous-système, DFD).

La phase 1 d'analyse des besoins est rarement prise en compte dans les méthodes actuelles.

3 Présentation des principales méthodes à objet

Nous nous intéressons aux concepts fondamentaux de chaque méthode ainsi qu'à sa démarche de modélisation. Nous ne ferons pas de comparaison des concepts et des termes utilisés.

3.1 Méthode OOSE de Jacobson

La méthode OOSE est la version objet de la méthode Objectory de Jacobson. Elle est basée sur un concept central, celui de scénarios (appelé *Use Cases*). La méthode OOSE intègre dès la première phase de modélisation d'un système d'information, outre la description des objets et de leur comportement, celle des interfaces utilisateurs. La modélisation d'un système est organisée en plusieurs niveaux. Le processus de développement détaille pas à pas les transformations des produits à chacun des niveaux. Le premier niveau utilise le modèle des Use Cases et consiste à identifier les acteurs du système et à définir leurs interactions avec le système. Les use Cases sont initialement extraits de l'analyse des documents de spécification.

L'approche par les scénarios guide la construction des modèles de domaine, d'analyse, de conception (technique), d'implémentation et de test.

q **Étapes de développement**

Le cycle de développement comporte les trois phases suivantes :

- phase d'analyse consistant à identifier les besoins puis spécifier le modèle conceptuel comportant les objets de structure, de contrôle et d'interface,
- phase de conception (technique) et d'implémentation,
- phase de test (unitaire, d'intégration et du système global).

La méthode OOSE présente plusieurs points forts, en particulier une couverture complète du cycle de développement et des domaines d'application très divers incluant le temps réel. Elle réussit à réunir un large consensus et, à terme, elle sera certainement la méthode objet de référence, tout au moins dans le cadre des eurométhodes.

3.2 Méthode OOA de Coad-Yourdon

La méthode OOA (Object-Oriented Analysis) de Coad et Yourdon [Coa 90] reprend les idées exposées dans le modèle OOSA de Shlaer &Mellor. Le modèle OOA complète le modèle OOSA et l'oriente plus spécifiquement vers l'analyse. Le modèle E-R de base est plus complet et plus précis. Les aspects traitement ne sont qu'esquissés, avec des scénarios textuels supportés par des messages numérotés. OOA considère qu'un système est composé de quatre éléments : le domaine du problème, l'interface usager, la gestion des données et enfin la gestion des tâches. La première phase de la méthode consiste à construire le modèle objet en mettant en évidence les cinq niveaux suivants : *sujets*, *classes*, *structures*, *attributs* et *services*.

Le sujet est une unité de partitionnement des systèmes importants. Chaque sujet représentera alors un sous-domaine. Le niveau classe représente les entités du systèmes sans leurs propriétés. Le niveau structure identifie les relations de généralisation/spécialisation, ainsi que les relations de composition entre les entités. La spécification de la structure de chaque classe est faite au niveau "attributs". Le comportement des objets de chaque classe est spécifié au niveau "services". La méthode a évolué pour couvrir tout le cycle de développement mais reste cependant très limitée sur l'aspect prise en compte de la dynamique [Coa 92]. Les concepts du modèle sont illustrés par la figure 1 ci-dessous.

Etude comparative des méthodes de conception orientées objet

M. Ayache, A. Flory

L.I.S.I.- Bât. 401
INSA , 20 Av. Albert Einstein
69621 - Villeurbanne Cedex
Tél. 72 73 84 67 Fax : 72 73 85 97
email : ayache@lisiflory.insa-lyon.fr
flory@insa.insa-lyon.fr

RÉSUME : *L'abondance des écrits et la variété des produits disponibles sont des indicateurs sur la généralisation de la technologie objet dans les organisations. Le nombre élevé de méthodes de conception orientées objet prouve que leur potentiel de pénétration dépasse de loin celui des méthodes classiques de conception. La question qui se pose actuellement n'est pas tant de savoir s'il faut passer à l'objet mais plutôt comment. Cet article est rédigé dans cet esprit et se propose d'aider l'utilisateur à se frayer un chemin et à se faire une idée face à l'offre méthodologique actuelle.*

1 Introduction

Bien que très récentes, les méthodes de conception orientées objet sont nombreuses, on en recense déjà plusieurs dizaines [Bai 89], [Boo 91], [Coa 90], [Cas 91], [Jac 92], [Rol 91], [Rum 91], [Roc 93], etc. L'orientation objet des méthodes de développement obéit d'une part au besoin d'exploiter au mieux la puissance des systèmes à objets disponibles (langages, SGBDOO, etc.).et, d'autre part, à la nécessité de prise en compte des besoins des applications dites non traditionnelles (CAO, Bureautique, etc.) gérant des entités complexes. L'approche traditionnelle par les données ou les traitements ayant montré ses limites car elle crée une distance sémantique entre les entités et la réalité modélisée. L'objet permet de réduire considérablement cette distance. Les apports de l'objet, outre l'encapsulation et la modularité, se situent essentiellement au niveau du développement incrémental, de la réutilisabilité et donc de l'efficacité et de la productivité.

La caractérisation ou la classification des méthodes de conception orientées n'est pas toujours aisée car, d'une part, la définition de critères d'évaluation n'est pas évidente et les critères retenus restent discutables et, d'autre part, leur jeunesse les rend instables car elles évoluent rapidement et toute classification est aussitôt dépassée. Nous tenterons toutefois de mettre en évidence les caractéristiques communes ainsi que leur spécificité.
Précisons d'abord le terme **conception**. Il est fréquemment utilisé avec des significations qu'il est utile de distinguer :
- La conception au sens Génie Logiciel permet de déterminer l'architecture logicielle d'une application, à partir d'un ensemble de spécifications fonctionnelles. Cette architecture doit être la mieux adaptée, au vu des critères de validité, extensibilité, réutilisabilité, etc.
- La conception au sens des Systèmes d'Information (SI) permet de spécifier les fonctionnalités du système à réaliser, à partir d'un énoncé incomplet et non élicité. Les anglo-saxons emploient les termes "Design" et "analysis" pour distinguer ces deux aspects. Pour notre part, nous nous intéressons principalement à la deuxième interprétation.

2 Orientation objet des méthodes

Les méthodes orientées objet utilisent toutes un concept central, l'objet auquel elles ajoutent les notions de structuration statique et dynamique avec les mécanismes d'abstraction (agrégation et spécialisation) et d'encapsulation. Toutes les méthodes orientées objet sont potentiellement basÈes sur le principe de développement par réutilisation avec retour arrière, contrairement au développement traditionnel où le cycle de développement est parcouru de façon systématique. Dans le cycle de développement orienté objet, le parcours est écourté par le fait de la réutilisation sauf pour les nouveaux composants.

Il faut cependant remarquer que les méthodes orientées objet développent dans leur grande majorité suivant un modèle vieux de vingt ans : analyse, conception (globale et détaillée) et enfin implémentation. Elles couvrent à des degrés divers les sept phases d'analyse et de conception suivantes :