Article

Complexité des systèmes d'information et de leur ingénierie

Jean-Pierre GIRAUDIN, Responsable de l'équipe SIGMA Laboratoire LIG, Laboratoire d'Informatique de Grenoble 681, rue de la Passerelle. Domaine universitaire 38402 Saint Martin d'Hères cedex. France, Jean-Pierre.Giraudin@imag.fr.

Date de publication : 9 mai 2007

Résumé

L'article porte sur la modélisation des systèmes d'information. Une présentation générale de la complexité des systèmes d'information est introduite selon trois types de caractéristiques : hétérogénéité, évolutivité et autonomie. La modélisation est d'abord évaluée au travers des modèles, des méta modèles et des diagrammes pour mettre en évidence les éléments de complexité comme l'interdépendance entre modèles ou les insuffisances des diagrammes. Ensuite, les principes d'abstraction comme la classification, l'association, la composition et la généralisation sont situés vis-à-vis de la complexité. Enfin, une synthèse sur les démarches d'ingénierie est proposée pour introduire de nouveaux processus de développement comme un cycle de vie en flocons centré métier ou l'ingénierie dirigée par les modèles.

Abstract

This paper deals with information systems modeling. A general view of information systems complexity is introduced based on three kinds of characteristics : heterogeneousness, evolution and autonomy. Modeling is first evaluated through models, metamodels and diagrams to put forward the elements of complexity such as the interdependance of models or inadequacy of diagrams. Then, the principles of abstraction such as classification, association, composition and generalization are evaluated versus the complexity. Finally, a summary of engineering approach is proposed to introduce new development process such as life cycle in flakes or models driven engineering.

Table des matières

1. INTRODUCTION

- **1.1 Constat**
- **1.2 Systèmes compliqués et systèmes complexes**
- **1.3 Cadre de référence**

2. MODÈLE

- **2.1 Principes de base**
- **2.2 Modèles et méta modèles**
- **2.3 Multimodélisation**
- **2.4 Modèles et diagrammes**
- **2.5 Premier bilan**

3. MODÉLISATION

- **3.1 Techniques d'abstraction**
- **3.2 Classification**
- **3.3 Association**
- **3.4 Composition-décomposition**
- **3.5 Généralisation-spécialisation**
- **3.6 Deuxième bilan**

4. PROCESSUS DE DÉVELOPPEMENT

4.1 Roue de Deming
4.2 Cycles de vie
4.3 Nouveaux processus de développement
4.4 Méthode
5. CONCLUSION ET PERSPECTIVES

Texte intégral

1. INTRODUCTION

1.1 Constat

Les systèmes d'information (SI) apportent-ils les réponses attendues en matière d'information et de traitement de l'information ?

Si les techniques informatiques ont évolué d'une manière rapide en 60 ans et ont révolutionné plusieurs fois les moyens mis à la disposition des organisations et des entreprises (langages de programmation de haut niveau, bases de données, progiciels intégrés et le couple Internet-Web notamment), les décideurs ne reçoivent pas toujours dans de bonnes conditions de lieu, de temps et de qualité, les informations qu'ils attendent, et les utilisateurs restent assez critiques à l'égard des systèmes développés par les professionnels de l'informatique. On pourrait résumer les causes de ces difficultés en disant qu'elles sont liées à une mauvaise informatisation sans respect de règles d'ingénierie, pour ensuite analyser les différents problèmes et leurs solutions et finalement promouvoir une nouvelle démarche d'informatisation. Une telle démarche peut être basée sur de nouveaux langages de spécification et des outils formels combinés à des outils graphiques appropriés à la conceptualisation des systèmes tout en étant directement « exécutables » et compris par des non informaticiens. Il y a plus de deux décennies, James Martin nous engageait ainsi vers une informatique sans programmeurs pour concevoir des systèmes d'information intégrant des facilités de gestion mais aussi d'aide à la décision (Martin 1982, 1985).

Malgré les progrès récents impressionnants de l'informatique notamment en termes d'interfaces et de réseaux sans fils, si la question initiale reste d'actualité, c'est que d'une part les systèmes d'information ne sont pas des systèmes simples et que d'autre part les exigences des nombreux et différents acteurs sont variées et très évolutives. Une autre manière d'analyser le constat initial est de centrer les difficultés de développement des SI sur leur complexité intrinsèque combinée à la complexité de leur ingénierie. Il est alors nécessaire de préciser ce que l'on entend par complexité. Dans cet article nous n'avons pas pour but de dire que tout va mal en informatique mais nous souhaitons simplement apporter une contribution à l'analyse de quelques traits de la complexité des systèmes d'information par le prisme de leur modélisation.

1.2 Systèmes compliqués et systèmes complexes

Un système d'information est-il compliqué ou complexe ?

Les définitions du Larousse encyclopédique sont d'une aide limitée pour différencier les systèmes d'information compliqués des systèmes d'information complexes (Larousse, 1961).

- Compliqué : qui est difficile à comprendre ou à exécuter par suite du grand nombre et de la diversité des éléments composants ou des opérations nécessaires… (Synonymes : ardu, difficile, confus, embrouillé).

- Complexe : qui contient plusieurs parties ou plusieurs éléments combinés d'une manière qui n'est pas immédiatement claire pour l'esprit… (Synonymes : compliqué, embrouillé, emmêlé).

Aujourd'hui, on a tendance à qualifier les systèmes de très grande taille de complexes alors qu'ils ne sont souvent que compliqués. Ce n'est pas le volume en giga, téra ou péta octets d'une base de données qui la rend complexe, mais sa complexité structurelle, les interrelations dynamiques entre ses objets, la gestion temporelle et la réactivité aux événements, le maintien de l'intégrité des données quelles que soient les transactions et les

pannes matérielles ou logicielles naturellement toujours imprévisibles, mais dont les risques et les réactions doivent avoir été évalués. Ces éléments sont à combiner avec la distribution et l'autonomie tant en matière de données que de processus. Dans le cas du SI d'une multinationale basé sur Internet et le web ou dans le cas du SI d'un opérateur de télécommunications, ce n'est pas le nombre d'instances ou le nombre d'événements qui induisent la complexité, mais c'est la diversité des éléments ainsi que l'hétérogénéité des architectures et de leurs entrelacements. Dans un tel système, la diversité est en termes de : machines, réseaux, postes de travail des utilisateurs, systèmes, intergiciels, architectures, bases de données, bases d'ontologies, applications partagées et distribuées, applications locales, etc.

Une compréhension du compliqué et du complexe nous est apportée par Jean-Louis Le Moigne qui, dans son approche systémique, nous précise que l'intelligibilité du compliqué se fait par simplification alors que l'intelligibilité du complexe peut se construire par modélisation. En d'autres termes, le compliqué est lié à une accumulation d'éléments qui conduit à une accumulation de difficultés. On peut simplifier un système compliqué en ne retenant que les éléments essentiels : on le « mutile » pour l'analyser. Alors que la nature complexe d'un système est inhérente plus aux combinaisons et coordinations nombreuses et variées entre éléments qu'aux éléments eux-mêmes. Un système complexe doit être **modélisé** pour le comprendre, mais il ne faut pas le « mutiler ». Une simplification hasardeuse ou une mutilation d'un système complexe peut détruire a priori son intelligibilité (Le Moigne, 1990).

Dans un contexte de sciences humaines, Jean Fourastié nous apporte un éclairage complémentaire pour appréhender les difficultés des phénomènes complexes : « …la science a fui la réalité synthétique et complexe, parce qu'elle n'y trouvait pas les simples déterminismes qu'elle y cherchait. Elle a démembré ces réalités en éléments, chacun fécond du point de vue de la recherche. Mais la juxtaposition des sciences élémentaires ainsi constituées, s'est révélée inapte à former la science du phénomène complexe : un être organisé, un ensemble, une société, sont autre chose que leurs constituants, de même qu'une molécule est autre chose qu'une brochette d'atomes… ainsi nous ignorons encore à peu près tout des ensembles organisés, vivants ou non, … » (Fourastié, 1966). Ainsi, un modèle de SI et la modélisation d'un SI sont tous deux complexes par essence :

- Un SI est généralement constitué d'un grand nombre d'éléments (composants) pour la plupart fortement inter reliés. Dans la théorie connexionniste, on construit un indice de complexité d'un système par le calcul d'une distance sur sa matrice de connexion par rapport à une matrice identité.

- Le modèle d'un SI s'appuie sur des interrelations entre représentations, certaines privilégiant les aspects dynamiques, d'autres les aspects statiques. Ce n'est pas la juxtaposition de représentations mais bien leur intrication qui rend ces modèles complexes.

- La modélisation d'un SI est un processus coordonnant des techniques d'abstraction, elles-mêmes non simples (classification, association, composition, généralisation, etc.) et dépendantes les unes des autres dans leur usage.

Dans la suite de cet article, nous nous limitons à la modélisation des systèmes d'information pour en caractériser certaines facettes plus liées à la complexité des systèmes et de leur ingénierie qu'aux aspects « compliqués », ces derniers étant largement pris en charge par un bon choix d'outils de présentation, de documentation, de sélection et de gestion. Pour conduire notre étude, nous utilisons un cadre de référence mettant en évidence des défis que doivent relever les SI.

1.3 Cadre de référence

Aujourd'hui, parmi tous les facteurs de complexité des systèmes, nous pouvons en retenir trois qui rendent difficiles la modélisation des systèmes ainsi que la définition de méthodes d'ingénierie de ces systèmes : l'hétérogénéité, l'autonomie et l'évolution (cf. figure 1).

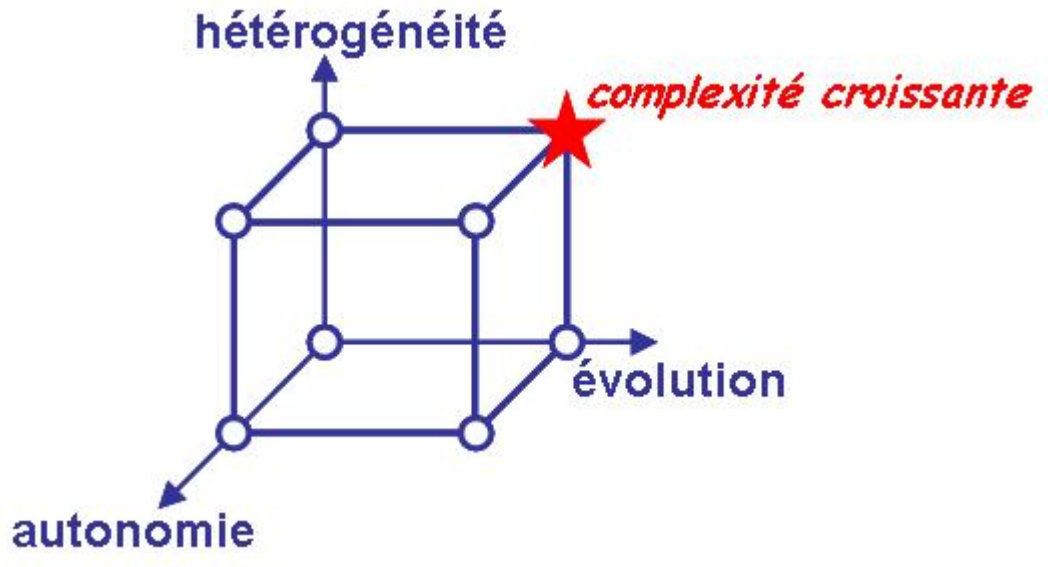

Figure 1. Facteurs de complexité

L'hétérogénéité est liée à la multiplicité des modèles utilisés : modèles différents selon des points de vue, niveaux d'abstraction (conceptuel, physique, etc.), types d'abstraction (données, transactions, etc.), catégories d'usage (gestionnaires, utilisateurs, etc.), domaines (étude, production, finance, *etc.*), etc. Un « modeleur » de SI, atelier pour modéliser ou spécialiste de la modélisation, serait une sorte de kaléidoscope aux nombreuses combinaisons avec autant de miroirs que de points de vue.

Un système est autonome s'il est déconnecté des autres systèmes. L'autonomie ou plutôt la très relative autonomie est induite par le fait qu'un SI n'est jamais isolé ; il s'insère dans son environnement donc dans un autre système avec lequel il a des échanges au travers d'interfaces, de dispositifs techniques, d'événements temporels ou factuels, etc. Un tel environnement contraint l'autonomie du SI étudié (contraintes temporelles, contraintes de ressources matérielles, humaines, financières, etc.). L'autonomie d'un SI est aussi limitée par le fait qu'il ne faut pas considérer un seul SI, mais des SI qui doivent coopérer a priori ou a posteriori (par exemple, le SI d'un bureau d'études, le SI d'une chaîne de production et le SI de la commercialisation).

L'évolution constitue l'aspect « vie » d'un SI. Le SI n'est pas figé dans le temps, il évolue. Les modèles du SI doivent aussi évoluer, s'adapter. Ces évolutions sont de natures variées : évolution technique (hier de type bases de données, aujourd'hui de type mobilité, demain…), évolution de domaine (hier « cantonné » dans un domaine spécifique, le SI devient aujourd'hui omniprésent dans tous les actes de traitement de l'information). Dans certains cas, on parlera de réingénierie, dans d'autres cas, on élaborera des mécanismes de coopération ou d'interopérabilité entre systèmes.

Globalement, ces trois facteurs sont liés à des entrelacements de modèles et de systèmes à situer d'une manière spatio-temporelle ; l'autonomie relève principalement de l'axe espace, alors que l'évolution se place sur l'axe temporel. La complexité d'un SI et de son ingénierie croît avec l'hétérogénéité et l'évolution de ses éléments, et décroît avec son autonomie. Nous utilisons ces trois facteurs comme points de repère pour les éléments présentés dans cet article. La section 2 précise des caractéristiques générales des modèles qui sont sources de complexité de la modélisation des SI. La section 3 complète ces premiers éléments en adressant les techniques de base d'abstraction appliquées dans toute modélisation de SI. La section 4 reprend les premières démarches de type cycle de vie du domaine du génie logiciel pour s'interroger sur les besoins et la complexité des processus d'ingénierie des SI. Une conclusion et des perspectives constituent la dernière section.

2. MODÈLE

2.1 Principes de base

Dans un premier temps, il est utile d'avoir des points de repère dans le labyrinthe des modèles, des méta modèles et de leur ingénierie.

« Pour un opérateur O, un objet M est un modèle d'un objet A dans la mesure où O peut utiliser M pour

répondre aux questions qui l'intéressent au sujet de A » (Minsky, 1968).

Les modèles des différentes normes ou méthodes de modélisation ne sont ni compliqués, ni complexes. Ils s'appuient sur quelques concepts de base assez naturels (classe, objet, propriété, association, acteur, événement, processus, état, composant, système, etc.) et un choix (pas toujours judicieux) de notations graphiques d'une part nécessaires à la communication, à l'aide à la compréhension, et d'autre part, utilisables pour des transformations ou transitions progressives du SI « idéal » vers un système ou des systèmes informatiques « produit ».

La complexité de la modélisation est liée à quatre problèmes principaux :

- Interdépendance entre modèles (cf. § 2.3) : par exemple, un modèle (dynamique) de type état-transition pour représenter les évolutions des objets d'une classe ne peut pas être finalisé sans le modèle (statique) des classes du système en cours d'étude.

- Prégnance des diagrammes sur les modèles (cf. § 2.4) : le lecteur d'un diagramme se limite trop souvent à celui-ci comme représentation externe « aboutie » d'une modélisation d'une partie d'un système et peut en oublier le contexte réel comme par exemple les justifications des contraintes vis-à-vis du système opérationnel futur ou du système en cours de déploiement. Un diagramme est forcément réduit donc souvent réducteur car sa taille pose un problème direct de représentation dans une page, alors qu'un modèle peut combiner plusieurs représentations dans une variété de langages sans limites matérielles de pagination.

- Choix entre concepts et processus d'abstraction (cf. §3) : par exemple, il est souhaitable d'utiliser un processus d'abstraction comme la composition ou la généralisation multiple même si l'on a par la suite une contrainte technique de réalisation comme par exemple avec le modèle relationnel et sa première forme normale qui impose l'atomicité, ou la programmation Java sans héritage multiple.

- Faiblesse des processus d'ingénierie (cf. §4) : les processus d'ingénierie décrivent le cheminement à suivre pour aller d'une intention d'un SI à un but, la réalisation finale de celui-ci. Si aujourd'hui, les modèles de données et de programmes voire d'objets sont bien définis et largement utilisés dans les équipes informatiques, les modèles de processus organisant les activités de ces équipes restent trop du domaine de la recherche. Il est nécessaire à terme de trouver un compromis entre des modèles trop directifs, comme les modèles orientés activité centrés sur l'ordonnancement de ce qu'il faut faire, et des modèles orientés stratégie qui offrent plusieurs cheminements possibles pour atteindre le même but : un modèle d'un système (Rolland, 2005).

2.2 Modèles et méta modèles

Dans le contexte informatique, un modèle est un système formel ou semi formel qui permet d'établir une abstraction d'un SI ou d'un logiciel. Il nous faut distinguer le modèle du système produit du modèle (méta modèle) utilisé pour modéliser (Rieu, 1997). Dans les néologismes scientifiques (métalangue : langue qui prend pour objet une autre langue et la formalise, etc.), méta signifie ce qui dépasse ou englobe une science ou un objet de pensée (Larousse, 1961). Dans le domaine de la modélisation des SI, méta modèle est pris au sens modèle de modèles ou modèle pour construire (instancier, représenter) des modèles. Par exemple en UML, on distingue le méta modèle des objets, exprimés dans le modèle MOF - *Meta Object Facility* de l'OMG (OMG, 1997), du modèle des classes et des objets d'un SI particulier. Aujourd'hui, le modèle d'un SI est souvent conforme au méta modèle UML, lui-même conforme au méta modèle MOF.

La figure 2 est un exemple très simplifié des trois niveaux de modèles auxquels l'ingénieur en SI se trouve confronté quand il s'intéresse à la description d'un processus, ou qu'il suit lui-même un processus de développement de SI. Il s'agit de l'exemple très ancien du cycle de développement en « cascade » (Royce, 1970) qui est en partie modélisé en SPEM - *Software Process Engineering MetaModel* (OMG, 2001) lui-même modélisé en MOF.

Cet exemple permet d'introduire un nouvel élément de complexité de la modélisation des SI. Ce processus d'ingénierie selon la cascade décrit dans la figure 2 utilise lui-même des modèles de produits comme par exemple le modèle entité-association lors de l'activité d'analyse, le modèle relationnel lors de l'activité de conception, etc. Ces modèles de produits seront aussi (méta) modélisés en MOF. Dans ce cas, la complexité est liée aux multiples entrelacements (verticaux, horizontaux) entre modèles : entre modèles de produits (entre

modèle entité-association et modèle relationnel, ou entre modèle de classes et modèle d'état-transition par exemple), et entre modèles de processus et modèles de produits (entre le processus d'analyse et le modèle entité-association utilisé lors de ce processus d'analyse par exemple). Un méta modèle comme MOF permet :

- de décrire formellement des modèles,
- d'unifier les représentations de modèles (comme UML, CWM, SPEM, etc.) (OMG, 1997, 1999, 2001),
- de créer de nouveaux modèles par instanciation.

Il est admis que dans le cas de la modélisation d'un SI, d'un point de vue modèle de produits, cette architecture de méta modélisation comporte quatre niveaux : le méta méta modèle MOF (par exemple le classifieur MOF), le méta modèle UML (par exemple le classifieur Classe), le modèle du SI (par exemple la classe Entrepôt), le SI avec ses objets terminaux (par exemple l'entrepôt pour stocker les produits intermédiaires). Pour limiter le nombre de niveaux, il est possible d'utiliser un modèle réflexif qui s'autodécrit ; il convient alors de distinguer le sous-modèle utilisable lors d'une activité spécifique de modélisation.

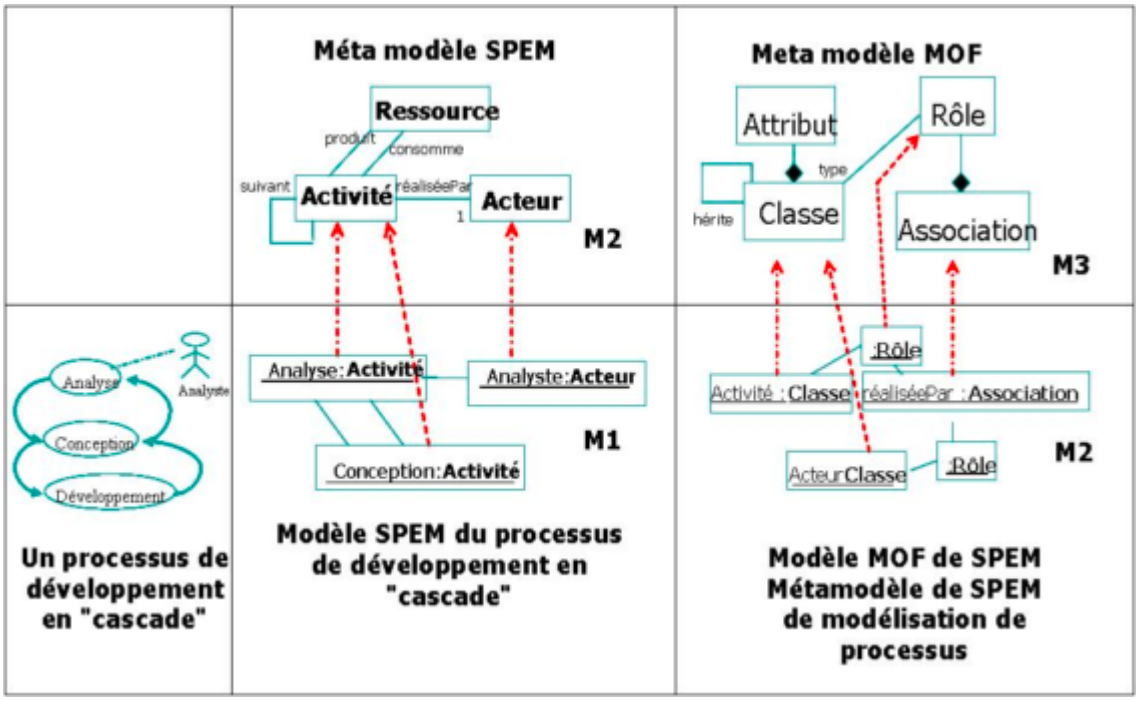

Figure 2. Interdépendances entre modèles et méta modèles

2.3 Multimodélisation

Si l'on se limite à un seul niveau de modélisation, la modélisation d'un SI spécifique, le processus de modélisation se trouve grandement simplifié. Plusieurs sources de complexité persistent néanmoins ; certaines sont liées à la notion de multi modélisation. Cette notion de multi modélisation recouvre plusieurs situations.

Plusieurs modèles entrelacés concourent à la modélisation d'un système selon différents domaines d'abstraction (structure, état, processus, etc.). Ces modèles sont à coordonner pour avoir l'abstraction la plus complète possible du système. La cohérence de ces modèles complémentaires doit être assurée.

Plusieurs modèles concourent à la modélisation d'un système selon différents points de vue externes dépendants des acteurs. Ces modèles sont à intégrer (fusionner) pour distinguer une abstraction commune du système des abstractions spécifiques à certains acteurs. Les problèmes posés sont alors en termes de conflits syntaxiques voire sémantiques entre modèles.

Plusieurs modèles concourent à la modélisation progressive d'un système selon un processus de développement allant par raffinements successifs d'une spécification abstraite à une implantation

opérationnelle. C'est le cas de l'approche MDA – *Model Driven Architecture* qui organise le processus de développement en distinguant des modèles indépendants de plates-formes techniques (PIM – *Platform Independant Model*), des modèles dépendants de plates-formes techniques (PSM - *Platform Specific Model*) (OMG, 2005). Dans un tel processus de raffinement de modèles, les difficultés se situent au niveau des règles de transformation ou de raffinement ainsi qu'en matière de maintien d'une forme de traçabilité entre modèles pour conserver en particulier les motivations des choix de conception réalisés lors du déroulement du processus de développement.

Nous pouvons illustrer le premier cas de multi modélisation par l'usage de différents groupes de notations UML, comme par exemple les méta modèles des séquences de messages et des transitions d'états qui sont liés aux méta modèles des classes et des objets. Colette Rolland organise une telle situation comme étant la possibilité de construire une méthode d'ingénierie de systèmes par composition de fragments (Rolland, 2005). Elle illustre son propos en « associant » les concepts d'un modèle de classes-objets avec les concepts d'un modèle d'états-transitions. Dans la figure 3, nous reprenons des correspondances entre un modèle de classes, un modèle d'objets et les diagrammes associés. Ces correspondances sont inhérentes au méta modèle UML et sont décrites dans (Muller, 1997). Les interdépendances entre modèles doivent être clairement identifiées dans le méta modèle général.

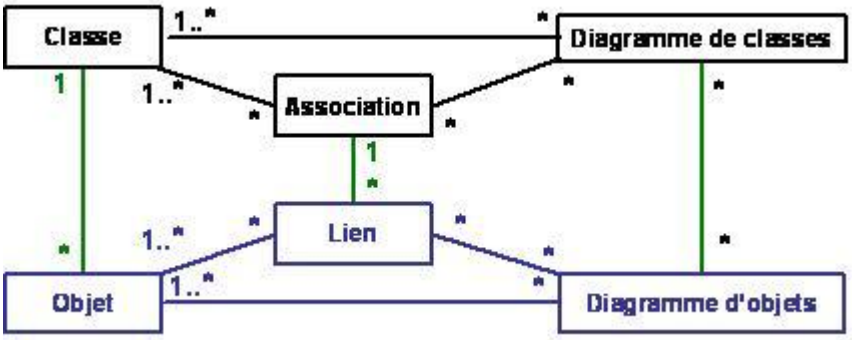

Figure 3. Interdépendances de concepts

L'une des difficultés en multi modélisation se situe au niveau de la coordination de modèles décrivant les aspects statiques d'un système avec des modèles des aspects dynamiques afin d'établir un modèle unifié supposé complet du système. Avec le standard UML, il faut par exemple coordonner les descriptions réalisées selon les diagrammes de classes, de séquence et de collaboration (Booch, 1998). La description du comportement requiert davantage d'informations que la description des structures de données ou des architectures.

2.4 Modèles et diagrammes

Si nous nous plaçons dans le cas d'UML (Booch, 1998), deux difficultés apparaissent nettement : l'une est liée au nombre de modèles utilisables et l'autre est liée au système de représentation graphique qui n'établit pas systématiquement une association 1-1 entre modèles et diagrammes. Ainsi UML1.x propose six modèles et neuf diagrammes pour établir la modélisation d'un SI (Muller, 1997).

Les modèles UML1.x utilisés pour le développement d'un SI sont les suivants : modèle de classes pour capturer la structure statique, modèle des états pour exprimer le comportement dynamique des objets, modèle des cas d'utilisation pour décrire les besoins de l'utilisateur, modèle d'interaction pour s'intéresser aux scénarios et aux flots de messages, modèle de réalisation pour formaliser les unités de travail, modèle de déploiement pour préciser la répartition des processus.

Les diagrammes permettent à l'utilisateur de visualiser, de manipuler et de communiquer des éléments de modélisation. Les diagrammes sont mis au point dans des éditeurs de diagrammes qui sont les seuls outils largement utilisés dans les équipes informatiques. Neuf diagrammes permettent de « parler » UML1.x : diagramme de classes pour représenter la structure statique en termes de classes et de relations, diagramme d'objets pour représenter plus concrètement un système par les objets le composant et leurs relations, diagramme d'états-transitions pour représenter le comportement des objets d'une classe, diagramme de cas d'utilisation pour représenter les services attendus du système du point de vue des utilisateurs, diagramme de séquence pour établir une représentation temporelle des objets et de leurs interactions, diagramme de

collaboration pour une représentation spatiale des objets, des liens et des interactions, diagramme d'activités pour représenter le comportement d'un système, diagramme de composants pour représenter les composants physiques du système, diagramme de déploiement pour représenter le déploiement des composants logiciels sur les dispositifs matériels.

La nouvelle norme UML2.x propose pas moins de treize diagrammes. Certains diagrammes sont qualifiés par les auteurs comme utiles pour avoir une vision des parties statiques d'un système (diagrammes de classes, diagrammes d'objets, diagrammes de composants, diagrammes de déploiement) alors que les autres diagrammes servent à visualiser d'une manière complémentaire les parties dynamiques d'un système (Booch, 1998). Certains diagrammes sont dits complémentaires ou supplémentaires par les auteurs d'UML (diagrammes de collaboration par exemple), ce qui introduit de réelles difficultés de mise en œuvre opérationnelle : par quel diagramme commencer ?

Dans le cas d'UML, on retiendra plus facilement les éléments et les notations du langage qui se concrétisent dans ces diagrammes distincts que les modèles fondamentaux à coordonner pour modéliser un SI. Dans la mesure où un diagramme est une représentation d'une description pour capturer une sémantique partielle d'un système, il est souhaitable pour éviter des interprétations divergentes d'associer clairement un diagramme au modèle de base sous-jacent, mais aussi aux modèles connexes ainsi qu'au cadre ou au niveau d'abstraction de référence. Par exemple, un diagramme de classes est à interpréter conjointement avec des diagrammes d'états-transitions caractérisant les classes à objets dont l'évolution est à gérer. Ce même diagramme d'états élaboré lors d'une phase d'analyse du domaine d'un SI sera repris pour donner naissance à un nouveau diagramme dans une phase de conception détaillée du logiciel correspondant. Ce dernier point soulève une nouvelle difficulté liée au processus de développement basé sur des élaborations successives de modèles. Les premiers modèles sont plus proches d'une spécification abstraite de besoins, alors que les derniers modèles deviennent des spécifications exécutables sur des plates-formes techniques déterminées. Mais dans cette vision de l'ingénierie dirigée par les modèles, il manque encore de bons outils formels comme supports permanents des modèles UML et de leurs transformations comme des sortes de machines abstraites que l'on raffinerait progressivement selon une approche type méthode B. Après avoir été l'un des principaux concepteurs du langage Z dans les années 1970-80, Jean-Raymond Abrial a conçu la méthode formelle B (Abrial, 1996) qui permet de passer des activités de spécification aux activités de conception et génération de code exécutable selon trois principes de base : le concept de machine abstraite comme unité de modélisation, le raffinement de machines abstraites pour opération de transformation et l'obligation de preuves formelles pour valider toute activité.

A défaut d'utiliser des notations et des approches formelles pour la spécification de systèmes, l'ingénieur UML multiplie les « dessins » de diagrammes plus ou moins corrects. En considérant les notations graphiques mises à la disposition de l'ingénieur, on pourrait parler de « boxologie » pour mettre en évidence les risques de confusion qu'elles peuvent engendrer. On peut citer, par exemple, la subtilité des notations UML pour distinguer l'agrégation (losange blanc) de la composition (losange noir). Plus récemment, Bernard Morand parle d'une conception diagrammatique des SI : « Un diagramme désigne dans son objet immédiat des relations entre parties qui ressemblent à celles de son objet réel. C'est ce caractère de désignation qui … en fait un instrument de représentation… Bref le diagramme ressemble à ce dont il est signe non pas par description (image) ou continuation (métaphore) mais par indication…Il s'agit donc d'une manière de contraindre l'interprétation…» (Morand, 2004). Une difficulté supplémentaire apparaît ; elle est induite par la possibilité de disposer de modalités différentes de présentation du même diagramme correspondant à un modèle d'un SI. Ainsi, par exemple, dans les différents outils d'édition de diagrammes de classes, il est possible de donner ou non les propriétés des classes, les noms, les rôles et les cardinalités dans les associations. Il faut noter que cette dernière facilité peut conduire à des erreurs d'interprétation : par défaut, un trait simple exprime une cardinalité monovaluée totale (1..1) pour le rôle correspondant ; il est donc important de préciser s'il s'agit d'un diagramme sommaire ou brut pour le différencier d'un diagramme épuré et terminé. Notre propos n'est pas de bannir tout diagramme, mais de souligner que les diagrammes utilisés en modélisation de SI sont des graphes et à ce titre les conventions d'interprétation doivent être normalisées ou explicites pour éviter des interprétations discordantes.

2.5 Premier bilan

Les diagrammes et les notations associées restent indispensables pour visualiser des cartographies des systèmes. Comment circulerions-nous dans notre monde concret en particulier dans les zones qui nous sont

encore inconnues ou peu connues sans une carte ou un guide ? Les modèles et leurs représentations sous formes de diagrammes sont des éléments incontournables pour comprendre des systèmes qui nous sont inconnus ou que nous croyons connaître mais dont nous avons oublié ou mésestimé certains traits.

Cette section a mis l'accent sur les principaux éléments d'hétérogénéité conduisant à une réelle complexité des modèles et des diagrammes associés des SI. Nous les avons regardés d'une manière globale. Si nous nous intéressons aux modèles d'une manière plus précise, il est nécessaire de les associer aux différentes techniques d'abstraction produisant ces modèles. Cet aspect étudié dans la section suivante concerne non seulement le facteur d'hétérogénéité mais aussi les facteurs d'autonomie et d'évolution, car il concerne la multiplicité des concepts et des modèles dans un cycle de développement dans l'espace et le temps.

3. MODÉLISATION

3.1 Techniques d'abstraction

En nous inspirant du développement sur l'abstraction dans le dictionnaire de Claude Allègre (Allègre, 2005), nous pouvons dire qu'en ingénierie des SI, la démarche scientifique consiste à traduire les observations dans un langage conforme à un modèle type (méta modèle). Les techniques d'abstraction que nous utilisons sont fondées implicitement sur un va-et-vient permanent entre le concret et l'abstrait, l'abstrait étant un modèle explicatif pour comprendre ou faire comprendre le concret.

Dans cette section, nous porterons un intérêt particulier à quatre techniques d'abstraction, la classification, l'association, la généralisation-spécialisation et la composition-décomposition, car elles sont assez générales et utilisées dans différents domaines scientifiques. Si dans un premier temps ces techniques semblent simples et naturelles, il est un peu plus délicat de les utiliser et de les coordonner dans des démarches de modélisation où les phases n'ont pas les mêmes exigences (analyse de besoins, conception globale, conception détaillée, implantation, etc.), les approches n'ont pas les mêmes buts (réutilisation, rétroconception, etc.), tout en utilisant le même ensemble de méta modèles. La variété et l'usage combiné des techniques d'abstraction et des concepts pour les supporter selon des contextes plus ou moins spécifiques rendent l'acte de modélisation d'une réelle complexité.

3.2 Classification

La classification constitue sans nul doute la technique la plus ancienne pour mettre de l'ordre dans la complexité de tout système ou phénomène naturel. La classification consiste à ranger dans un même groupe désigné par un nom, des individus, des objets, des faits ou des phénomènes qui ont des caractéristiques communes. Si dans le monde du vivant les classifications sont fondées sur des considérations structurelles ou des propriétés physiologiques effectives, dans le domaine de la modélisation des SI, les classifications sont souvent dirigées par analogie au monde réel qui par des noms communs classe toutes les entités dont on veut pouvoir parler, ou parfois arbitraires dans la mesure où l'on fait référence à un monde virtuel sans limite dans la création de nouvelles entités.

La taxinomie ou science des méthodes de classification ne dit pas que toute classification doit être objective. « En fait, une bonne classification d'objets et de concepts est celle qui organise la nature avec la meilleure économie de pensée tout en traduisant sa diversité » (Allègre, 2005). En matière de SI, il faut d'abord distinguer deux niveaux de classification : le niveau des concepts communs utilisables (sortes de méta classes) et les classes originales que l'ingénieur spécifiera comme éléments importants de modélisation d'un SI. Par exemple, en UML, classe, association, état, acteur, activité, etc. sont des concepts utilisables pour toute modélisation de SI, alors que LIVRE et ABONNE sont des classes, DISPONIBLE et EMPRUNTE sont des états, RESERVATION et EMPRUNT sont des activités dans le système d'information d'une bibliothèque. La classe « CLASSIFIER » joue un rôle central dans le méta modèle MOF (OMG, 1997) pour définir les concepts d'un modèle comme UML.

Dans le cas d'une modélisation centrée données, la classification peut s'appuyer largement sur le langage mathématique et en particulier sur la théorie des ensembles et des relations. On dispose alors d'un concept théorique, celui d'ensemble pour structurer les données, distinguer des ensembles d'éléments, les définir en extension ou en compréhension, utiliser des opérations sur les ensembles (union, intersection, différence, produit cartésien), exprimer des liaisons entre éléments par des relations, sous-ensembles de produits

cartésiens, etc. Ce type d'approche a permis de disposer, pour la conception de bases de données, de modèles simples comme le modèle relationnel (Codd, 1970) ou plus complets comme le langage Z (Spivey, 1989). On peut qualifier le modèle relationnel de modèle de complexité 1, car il est fondé sur un concept de base central : la relation. Le langage Z et le modèle entité-association seront dits de complexité 2, car des associations (ou relations) sont utilisées pour exprimer des « rapprochements » entre éléments des ensembles.

Avec le concept d'association (cf. §3.3) on quitte définitivement le monde des modèles simples. Ce concept d'association peut lui même être spécialisé pour introduire le fait que les objets d'un SI peuvent être composites (cf. §3.4). Dans une modélisation objet d'un système, les classes peuvent contenir généralement un nombre quelconque d'objets. Certaines classes dites singletons ne contiennent qu'un unique objet. Les classes abstraites ne comportent pas d'objets ; elles figurent alors en amont d'une hiérarchie de généralisation-spécialisation (cf. §3.5). Ainsi, au-delà du simple usage des classes, les liaisons exprimées entre classes sont de natures très différentes comme les hiérarchies de « sens » des généralisations-spécialisations ou les associations plus transversales dont les intrications rendent complexe l'acte de modéliser un SI.

3.3 Association

La notion d'association ou action de rapprocher, d'assembler, de lier des objets est peu distincte de la notion de relation, rapport qui lie un objet à un autre (Larousse, 1961). En modélisation des SI, les associations sont utilisées pour exprimer des connexions sémantiques entre éléments. En UML, le terme d'association est utilisé principalement au niveau structurel dans le modèle des classes et des objets pour décrire des ensembles de liens binaires, ternaires ou plus généralement n-aires entre objets. Le terme de relation quant à lui est réservé pour distinguer les différents types de liaisons :

- une association est une relation structurelle entre différents objets,
- une agrégation est un type particulier d'association qui représente une relation structurelle entre un tout et ses parties,
- une généralisation-spécialisation est une relation d'organisation hiérarchique des classes selon le principe « est-un, est-une-sorte-de » de représentation de connaissances,
- etc.

Si l'on regarde le cas d'une association définie entre deux classes, un nom permet de l'identifier et sa définition doit être complétée par les noms des rôles de chaque classe et les cardinalités pour préciser des caractéristiques d'unicité ou de multiplicité, de partialité ou de totalité des liaisons possibles. Ces caractéristiques jouent un rôle analogue aux propriétés d'injection ou de surjection dans la définition des fonctions en mathématique. Alors qu'il peut y avoir plusieurs associations différentes entre mêmes classes, il est surprenant de voir des éditeurs de diagrammes qui autorisent des descriptions d'associations sans nom, ni nom de rôles.

Dans un modèle de classes d'objets, l'association est une technique d'abstraction fondamentale dans la perception d'un système comme un ensemble d'éléments structurels inter reliés. Par défaut, les associations entre classes d'objets sont « navigables » dans toutes les directions. Lors de la réalisation d'un système concret support d'un SI, il est possible de contraindre cette navigabilité en indiquant les seules directions de navigation utilisables (flèche portée par un rôle sur un diagramme de classes UML). Cette dissymétrie de navigation ne doit pas conduire à supprimer les cardinalités du rôle inverse ce qui nuirait à une bonne interprétation de l'association : la référence ou le pointeur dans un langage de programmation est une technique pratique de navigation et non une technique de représentation de la sémantique d'une liaison. Ce « glissement » d'usage de l'association ou de la relation vers la fonction (d'accès, de service) ne doit pas faire oublier qu'en mathématique, la définition d'une fonction est établie tant du point de vue de son domaine que de son co-domaine (propriétés d'injection et de surjection par exemple). Si l'on veut éviter les pertes d'informations sémantiques, dans un processus d'ingénierie, il est indispensable de garder et d'actualiser les modèles construits ou déduits lors des activités successives de développement.

3.4 Composition-décomposition

Face à la complexité d'un système, une solution classique consiste à le décomposer en unités moins complexes. Chaque unité ou sous-système traite d'un sous-ensemble particulier de préoccupations. En matière informatique, ce principe cartésien de décomposition a marqué les années 70 avec l'avènement de la modularisation (Parnas, 1972) (Dijkstra, 1976).

« Le fait que la plupart des systèmes complexes aient une structure arborescente quasi-décomposable a une importance primordiale. Grâce à cela, nous parvenons à comprendre, à décrire, et même à « voir » de tels systèmes et leurs composants. Il faudrait peut-être présenter différemment cette proposition. S'il existe dans le monde des systèmes complexes qui ne soient pas arborescents, ils doivent dans une grande mesure échapper à notre observation et à notre compréhension. L'analyse de leurs comportements mettrait en œuvre tant de connaissances et tant de calculs concernant les interactions entre leurs parties élémentaires qu'elle serait très au-delà de nos capacités de mémorisation et de calcul » (Simon, 1974). Naturellement cette proposition d'une part est à resituer dans son contexte et son époque et d'autre part, elle ne rejette pas les interactions mais de fait, seules les propriétés agrégeantes des parties d'un système ou d'un sous-système interviennent dans la description des interactions entre ces parties. Aujourd'hui, nous disposons de logiciels graphiques et de navigation utilisant pleinement les facilités des interfaces interactives pour nous aider dans la « vision » et en conséquence une compréhension de structures non arborescentes. La difficulté est plus au niveau de la compréhension des concepts manipulés qui véhiculent dans certains cas une superposition et un entrelacement de préoccupations.

Dans plusieurs cas, une relation de composition-décomposition est vue de manière équivalente à une relation d'inclusion : un système inclut des sous-systèmes, un paquetage inclut des paquetages, un cas d'utilisation inclut un autre cas d'utilisation. Une telle relation d'inclusion ne doit pas être confondue avec une relation d'utilisation ou d'importation à caractère plus momentané.

Dans le cas du modèle de classes d'UML, une nouvelle difficulté apparaît ; l'association est complétée par deux autres concepts : l'agrégation et la composition. L'agrégation est une association particulière qui définit une relation tout-partie entre l'agrégat (le tout) et un élément (la partie). La navigabilité peut se faire dans les deux sens : « voir » les parties d'un tout, « voir » le tout d'une partie. Le tout ne possède ni impérativement ni exclusivement ses parties ; leurs cycles de vie peuvent être indépendants. La composition complète la notion d'agrégation pour introduire une forte « possession » des parties par le tout par deux contraintes : une contrainte d'exclusivité pour préciser qu'une partie ne peut appartenir qu'à un tout, et une contrainte de dépendance entre cycles de vie avec les parties qui peuvent être créées après le tout, mais qui meurent avec lui. Agrégations et compositions se distinguent dans un diagramme UML par deux représentations graphiques différentes : un losange blanc pour l'agrégation et un losange noir pour la composition.

De nombreux domaines scientifiques (chimie, sciences cognitives, etc.) se sont intéressés à la composition et à en établir des taxinomies (Winston, 1987). Des travaux ont été repris dans le domaine de la modélisation et de la programmation objet (Odell, 1994) qui, au-delà de la différentiation de types de composition, mettent l'accent sur des propriétés de la dépendance entre composé et composant. Les quatre propriétés souvent retenues sont :

- la fonctionnalité précise une dépendance fonctionnelle entre le composant et le composé. Les caractéristiques fonctionnelles du composé seraient « mutilées » sans les composants et leurs propres caractéristiques fonctionnelles,
- la similarité exprime que les composants sont similaires entre eux et à leur composé,
- la séparabilité permet d'indiquer que la séparation entre le composant et le composé est possible,
- la simultanéité conduit à décrire un composé et ses composants en même temps.

Ainsi, une combinaison de chacune de ces propriétés avec une valeur de vérité à vrai ou à faux représente une catégorie de composition. Par exemple, une composition matérielle généralement nommée composition « composite-composant » peut être caractérisée par les propriétés de fonctionnalité, non similarité, séparabilité et simultanéité.

Pour la modélisation des systèmes d'information, les propriétés de fonctionnalité et de similarité permettent de mieux appréhender la signification de l'objet modélisé, alors que les propriétés de séparabilité et de simultanéité constituent une forme de spécification à fort impact sur la conception des opérations de création, suppression et connexion des instances du système. A ces quatre propriétés d'une liaison de composition, on

peut ajouter la propriété d'exclusivité qui permet de préciser qu'un objet composant ne peut faire partie que d'un seul objet composé ; c'est le cas pour la modélisation directe d'objets matériels : une porte est composante exclusive d'un appartement lui-même composant exclusif d'un immeuble. A l'inverse, la non exclusivité d'une liaison, ou le partage d'une liaison, permet à un objet composant de faire partie de plusieurs composés simultanément ; ce type de liaison est utilisé pour des compositions immatérielles voire abstraites : un type de portes peut être composant de plusieurs types d'appartements. Dans le domaine de la fabrication de produit, on distingue couramment deux sous-modèles entrelacés : le sous-modèle des nomenclatures de fabrication et le sous-modèle des produits matériels fabriqués (cf. figure 4).

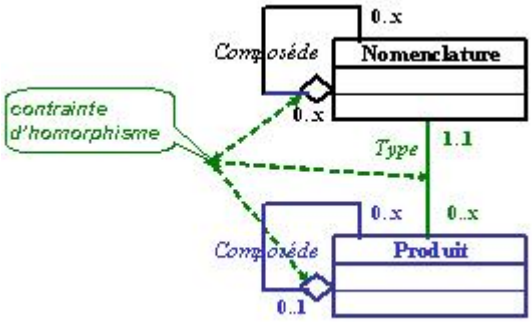

Figure 4. Homomorphisme de modèles

Dans le cas de la figure 4, la composition de produits est exclusive, alors que la composition de nomenclatures n'est pas exclusive. On peut remarquer aussi que ces deux compositions ne sont pas indépendantes ; elles respectent les propriétés d'un homomorphisme entre modèles car vis-à-vis des deux compositions ou lois internes, l'application « Type » est telle que l'image d'un composé de deux objets est le composé des images de ces objets.

3.5 Généralisation-spécialisation

La relation de généralisation-spécialisation permet une organisation hiérarchique des abstractions d'un système : les abstractions les plus générales en haut de la hiérarchie et les abstractions plus spécialisées ou spécifiques en bas.

Une relation de généralisation-spécialisation entre deux classes d'objets peut être étudiée avec un double point de vue ou une double inclusion :

inclusion intensionnelle - Les classes les plus spécialisées incorporent la structure et le comportement de la classe plus générale. On parle aussi d'héritage parent-enfant. Les descriptions d'une classe générale (parent) sont alors applicables aux objets des classes plus spécialisées (enfants).

inclusion extensionnelle - Les objets des classes les plus spécialisées sont aussi des objets de la classe plus générale. Cette inclusion ensembliste peut être contrainte : disjonction ou recouvrement entre classes spécialisées, couverture partielle ou totale, voire partition d'une classe.

La généralisation-spécialisation entre classes introduit une relation d'ordre partiel transitive et antisymétrique. Elle est de type treillis si la généralisation multiple est autorisée.

Des difficultés apparaissent à l'usage de cette technique d'abstraction.

Le vocabulaire n'est pas toujours bien choisi. Nous préférons les qualificatifs de classes générales et de classes spécialisées aux qualificatifs trop ambiguës de super-classes et sous-classes. Le lecteur peu averti sera étonné de voir une super-classe moins « riche » que ses sous-classes.

La possibilité pour une classe d'avoir plusieurs classes plus générales (généralisation multiple) est admise en modélisation des SI mais peut être interdite en phase de codage dans un langage de programmation. En programmation, il est aussi possible de définir une nouvelle implantation d'une opération pour les objets d'une classe spécialisée par rapport à l'opération correspondante de la classe générale. Si d'un point de vue opérationnel, les objets d'une classe spécialisée peuvent remplacer les objets de la classe générale, cette substitution peut devenir délicate en cas de redéfinition d'opérations.

La coordination de l'usage de la généralisation-spécialisation avec l'agrégation-particularisation se conçoit

bien quand on s'intéresse à modéliser des données d'un système (Smith, 1977). Elle reste plus délicate à réaliser sur l'ensemble des concepts fondamentaux utiles en modélisation des SI. Par exemple, pour l'abstraction des cas d'utilisation d'UML on combine trois relations : généralisation, inclusion et extension.

3.6 Deuxième bilan

La multiplication des concepts et des techniques d'abstraction associées est source de difficultés. Dans une première approche, il semble naturel de distinguer par exemple les classes des classes-associations en donnant comme éléments de définition pour ces dernières que, contrairement aux classes, leurs instances n'ont pas d'existence propre indépendante des instances des classes liées ; mais alors que dire d'une classe où tous les rôles vis-à-vis d'associations sont avec une cardinalité totale, ce qui impose que toute instance d'une telle classe est liée à au moins une instance de chaque classe associée.

Une autre difficulté majeure est de ne pas pouvoir utiliser le même ensemble de techniques d'abstraction (classification, association, composition, généralisation) selon les concepts utilisés (classe, cas d'utilisation, état, activité, etc.) et selon les différents niveaux du processus d'ingénierie (analyse, conception, codage). Cette situation ne facilite pas l'apprentissage de la modélisation des SI et rend difficile la traçabilité des modèles au sein d'une démarche d'ingénierie.

La composition-décomposition est une technique simple et générale ; elle a été largement utilisée dès les anciennes approches cartésiennes. Il n'est cependant pas certain qu'il soit nécessaire de la promouvoir systématiquement. L'usage de l'association ordinaire avec des cardinalités (partielle/totale, monovaluée/multivaluée) pour chaque rôle complétée par une propriété de séparabilité voire de simultanéité, serait suffisant et homogène quelle que soit l'étape d'abstraction au sein d'une démarche d'ingénierie. Des notions ambiguës comme composition « forte » et composition « faible » sont avantageusement remplacées par des propriétés plus précises quantitatives (cardinalités) ou opérationnelles (séparabilité par exemple).

4. PROCESSUS DE DÉVELOPPEMENT

Dans le labyrinthe des (méta)modèles et des techniques d'abstraction, les processus de développement sont des guides. On parle aussi de démarche pour guider les activités de modélisation et de réalisation de systèmes. Le choix ou la définition d'une démarche dépend beaucoup des entreprises, de leur organisation, du type d'activités, etc. L'engouement actuel autour des processus de développement témoigne d'une prise de conscience de leur importance. Certains processus sont plus des processus de l'ingénierie du logiciel, d'autres sont plus généraux. Après une présentation succincte d'un processus de développement très général ainsi que des processus reconnus en génie logiciel, nous introduisons des éléments des nouveaux processus de développement qui solutionnent partiellement certains aspects de la complexité de l'ingénierie des SI.

4.1 Roue de Deming

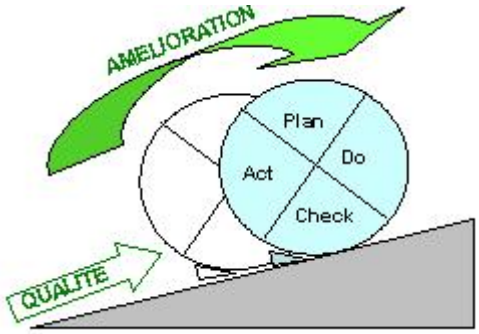

Figure 5. Principe de la roue de Deming

Un exemple de processus général est donné par la roue de Deming utilisé dans les démarches qualité. Ce processus comporte quatre phases, chacune entraînant l'autre dans un cycle vertueux pour améliorer sans cesse la qualité de quelque chose (un produit, un service, etc.) :

- Plan / Préparer - Etablir un cahier des charges en posant les questions classiques : Quoi ? Qui ? Où ? Quand ? Comment ? Combien ? Pourquoi ?
- Do / Faire - Construire ce qui est prévu : concevoir, réaliser, etc.
- Check / Contrôler – Vérifier et valider ce qui a été fait vis-à-vis de ce qui était prévu.
- Act / Réagir – Rechercher des points d'amélioration et recommencer le cycle.

4.2 Cycles de vie

Les processus de développement ou cycles de vie du génie logiciel sont constitués de phases et correspondent aux premières contributions significatives pour organiser la modélisation des SI. Ils mettent en évidence les phases nécessaires, leur ordonnancement, les produits obtenus. Ils sont essentiels en termes de gestion de projets informatiques. En une trentaine d'années, ces modèles de processus d'ingénierie se sont raffinés. Ils s'appuient sur des métaphores (cascade, fontaine, etc.).

Le cycle de la cascade pose comme principe la séquentialité des phases (Royce, 1970). Son usage est délicat en cas d'erreur ; il faut refaire la phase en cours ou la phase précédente, voire une phase plus en amont. Cette démarche simple est critiquée d'une part pour son impact sur le coût d'une erreur décelée trop tardivement dans le cycle de développement, et d'autre part pour l' « effet tunnel » qu'elle induit par un délai trop long entre l'expression des besoins et la mise en exploitation du système opérationnel.

Le cycle en V améliore le cycle de la cascade pour mettre l'accent sur les aspects vérification et validation. Les jeux de tests sont préparés dès les phases de spécification et permettent ainsi d'améliorer la production de logiciels corrects et valides (norme AFNOR Z67-130). Le cycle en W est une première tentative pour s'intéresser à concevoir le système en deux parties, toutes deux développées selon un V. Le premier V représente la partie centrale du système, alors que le deuxième V ou d'autres V correspondent aux parties annexes.

Le cycle en spirale généralise le principe d'incrément et évite « l'effet tunnel » où le temps est trop long entre les premières spécifications de besoins et la livraison de tout le système (Boehm, 1986). Il introduit la notion de prototype. Ce cycle de vie correspond à une adaptation du principe de la roue de Deming.

Le cycle en fontaine met en exergue la notion de composant et favorise un processus de développement orienté réutilisation (Henderson-Sellers, 1990). Le modèle d'un système doit être modularisé afin que tout module soit archivé pour être réutilisé pour modéliser d'autres systèmes.

Le cycle en Y dissocie les aspects techniques (branche droite) des aspects fonctionnels (branche gauche) pour ensuite traiter les phases de conception puis de réalisation (branche centrale) ; c'est le cas par exemple du processus 2TUP - *Two Track Unified Process* (Rocques, 2002). Une variante de ce cycle peut être proposée avec une branche centrale amont pour piloter et coordonner les deux premières branches : cycle en□.

La courbe du soleil est une extension de la démarche Merise (Nanci, 1992). La démarche Merise est basée sur des niveaux d'abstraction (conceptuel, organisationnel, logique, physique) et une dichotomie données-traitements. La courbe du soleil met en évidence les difficultés pour commencer une modélisation. Elle préconise de démarrer par une phase de compréhension ou rétro conception du système existant (partie montante de la courbe) pour ensuite appliquer une phase de type cascade (partie descendante de la courbe) pour réaliser le nouveau système à l'image du soleil qui se lève progressivement pour atteindre son apogée dans la journée (le schéma conceptuel) et ensuite décliner. Ce cheminement du processus est tout à fait adapté dans le cas de la réingénierie d'un SI.

Chacun de ces processus de développement des SI peut constituer un référentiel pour appréhender la complexité de la modélisation des systèmes : à quelle phase correspond un modèle produit ? Dans quel ordre construire les modèles ? Un état de l'art historique et complet de ces processus de développement des systèmes pourrait s'appuyer sur le modèle standard de définition de processus de développement SPEM (OMG, 2001). Ces processus de développement poursuivent leur évolution et de nouvelles approches apparaissent autour d'UML.

4.3 Nouveaux processus de développement

Aujourd'hui, les nouveaux processus de développement s'appuient sur des principes complémentaires.

Le concept de composant induit des processus de développement combinant deux sous-processus : processus « par » réutilisation et processus « pour » réutilisation. En d'autres termes la notion de composant réutilisable combinée à l'approche MDA (OMG, 2005) place les processus de développement dans une perspective « patrimoniale » où il s'agit de construire un patrimoine, thésauriser sur ce patrimoine, utiliser et réutiliser ce patrimoine, faire évoluer ce patrimoine. Le niveau d'abstraction et de réutilisation peut être augmenté avec le concept de patron. Ce concept est proposé pour capitaliser des expériences en conception orientée objet tant d'un point de vue produit que d'un point de vue processus (Gamma, 1995) (Ambler, 1998). Les patrons de type processus constituent une solution efficace pour formaliser une démarche (Gzara, 2000) (Hassine, 2005).

L'approche incrémentale et itérative est reprise dans le concept de processus unifié (UP - Unified Processus, RUP – Rational Unified Processus, etc.) construit sur UML (Jacobson, 1999). La définition d'incréments de réalisation est une bonne pratique pour limiter les risques techniques et fonctionnels. Chaque incrément fournit aux utilisateurs un résultat tangible de manière analogue à un tour de la roue de Deming ou à un pas de la spirale de Boehm. Les itérations caractérisent des degrés d'abstraction de plus en plus précis, des modèles de plus en plus détaillés qui à la fin correspondent à des modèles d'exécution ou à des applications déployées. A ce niveau, on rejoint le cycle primitif de la cascade allant de la capture des besoins jusqu'à la mise en service d'un logiciel.

De multiples vues sont requises pour considérer un système. Dans le cas de la construction d'une maison, on imagine la variété des plans complémentaires destinés aux différents corps de métier (maçonnerie, électricité, plomberie, etc.). L'architecture des 4+1 vues proposée par Philippe Kruchten part du même principe de plans complémentaires pour ordonner l'usage des différents modèles UML dans le cadre d'une approche centrée cas d'utilisation pour la vue centrale pilotant les quatre autres vues (logique, processus, réalisation, déploiement) (Kruchten, 1995).

Les approches dirigées par le métier s'imposent maintenant par opposition aux approches dirigées par les techniques. Ainsi un objet métier peut être vu comme un modèle qui correspond à une abstraction d'un concept relatif à un métier. Un objet métier peut représenter différents types de connaissances : entités, processus, propriétés, etc. Il s'agit de maintenir ces abstractions dans un cycle de développement de l'expression des besoins à la réalisation des logiciels et à leur évolution.

Toutes ces nouvelles approches s'intègrent dans le courant actuel d'une Ingénierie Dirigée par les Modèles (IDM). Les travaux de recherche de l'équipe SIGMA (http://www-lsr.imag.fr/sigma.html) et plus particulièrement la thèse d'Ibtissem Hassine s'appuient sur ces principes pour améliorer le cycle de vie en Y et proposer le **cycle de vie en flocons** (cf. figure 5). Le cycle de vie en flocons est composé d'un ensemble de cycles de développement en Y. Chaque cycle de développement en Y est centré sur un processus métier. Chaque processus métier est raffiné et peut être décomposé en sous-processus métier qui chacun à leur tour seront développés selon un cycle en Y dont l'ensemble constitue un nouveau flocon (Hassine, 2005).

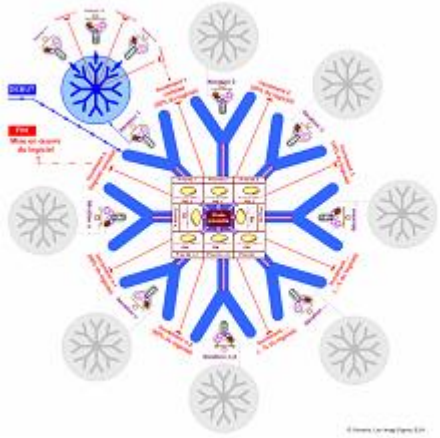

Figure 6. Processus d'ingénierie des systèmes en flocons

4.4 Méthode

Un processus de développement est un modèle des activités des ingénieurs lors du développement de SI. Les

modèles de processus décrits ci-dessus sont des sortes de canevas types plus ou moins directifs et souvent adaptés au contexte spécifique. Une composition de tels processus est souvent réalisée, comme par exemple pour chaque cycle en Y, ajouter en amont une phase analogue à la partie montante de la courbe du soleil pour initialiser les spécifications des contraintes fonctionnelles et des contraintes techniques.

Le processus d'ingénierie (ou démarche) est une donnée essentielle pour la maîtrise du développement des SI, mais n'est cependant qu'une partie d'une méthode d'ingénierie. Une méthode d'ingénierie est constituée par une combinaison de modèles, de langages (diagrammes ou grammaires), d'un processus ou d'une démarche, et d'outils.

Ainsi une méthode est elle-même un système que l'on peut qualifier de compliqué et de complexe car d'une part il accumule des difficultés liées au grand nombre de concepts, d'activités, etc., et d'autre part les interconnexions sont variées et fondamentales, qu'elles soient entre modèles d'une même phase dans un cycle, entre phases d'un cycle, entre sous-systèmes ou lors d'une réingénierie d'un système (cf. les trois facteurs du cadre de référence : hétérogénéité, autonomie et évolution).

Les nouveaux défis se situent maintenant au niveau des méthodes. Il s'agit de mieux les définir, les formaliser et les instrumenter. Par exemple, Colette Rolland coordonne tout un courant de recherche pour composer une méthode ad hoc à partir de fragments de méthodes (Rolland, 2005) par analogie avec la composition d'un SI avec des composants. Elle préconise aussi de disposer de démarches plus flexibles basées sur une approche par buts et des stratégies pour atteindre ces buts : il s'agit du modèle intentionnel MAP (Rolland, 1999).

5. CONCLUSION ET PERSPECTIVES

La complexité croissante des systèmes et de leur ingénierie a révélé ces deux dernières décennies des réponses conjointes apportées par les deux domaines de recherche Systèmes d'Information et Génie Logiciel. Le langage standard UML et les démarches d'accompagnement ont montré l'utilité et l'efficacité d'un même langage pour différentes situations : analyse du domaine, spécification de systèmes, conception détaillée, implémentation, réutilisation de spécifications, séparation de modèles indépendants d'une implémentation (PIM/PSM), etc. Les perspectives les plus immédiates sont marquées par une maturité du concept de composant au niveau des produits d'étapes d'ingénierie (Mourad Oussalah, 2005). En ce qui concerne la définition des processus d'ingénierie, les travaux de recherche marquent d'une part les limites voire la fin des méthodes à vocation universelle et d'autre part les premiers éléments de méthodes à composer selon les spécificités des projets et des domaines d'application (Rolland, 2005).

Les trois facteurs d'hétérogénéité, d'évolutivité et d'autonomie qui ont guidé l'introduction de notre bilan sur les modèles et la modélisation des SI, sont fortement corrélés et contraints par la dualité espace-temps qui sous-tend ce domaine. Ils correspondent à l'organisation de la modélisation d'un SI selon trois axes majeurs :

Modélisation du système perçu comportant principalement des descriptions structurelles pour spécifier ce qu'est le système, comment il est construit, ce qu'il mémorise, etc.

Modélisation du système actif et évolutif basée sur des descriptions de processus pour préciser ce que fait le système, comment il le fait, pourquoi, pour qui et quand il le fait, comment le système évolue selon des états pertinents, etc.

Modélisation du système couplé à son environnement à l'aide de descriptions de coordination et d'échange réalisant l'adéquation entre ses buts propres et ceux de son environnement voire avec d'autres systèmes coopérants.

Ces axes ne sont pas nouveaux. Ils rappellent que la complexité des modèles et de la modélisation est inhérente aux innombrables interconnexions des éléments et des systèmes en interne et avec leur environnement. Les bases de données ont privilégié le premier axe en structurant principalement les données. La programmation procédurale combine les deux derniers axes en décrivant les actions et les entrées-sorties avec l'environnement. Les approches par objets constituent une avancée importante pour combiner les trois axes. Le modèle CRC – Classe/Responsabilité/Collaboration (Wirfs-Brock, 1990) ou le modèle de composants métier SYMPHONY (Hassine, 2005) basé sur une représentation en trois parties de chaque composant « ce que je suis – ce que je sais faire – ce j'utilise » sont des illustrations des possibilités actuelles à combiner avec des besoins en termes de modèles d'architecture des systèmes pour d'une part « composer » des systèmes et d'autre part en assurer une cartographie support de compréhension et d'interprétation. L'évolution continue des modèles comme le passage à UML2 et son nouveau modèle de connexion de composants repousse les défis en matière de modélisation des SI vers les méthodes et les outils.

Au-delà de la définition de nouveaux méta modèles de SI, aujourd'hui, on est en mesure d'améliorer la qualité des modèles produits. On peut par exemple envisager des interfaces d'édition de diagrammes plus intelligentes pour assister l'ingénieur modeleur de SI. De la même manière qu'un assistant Word reconnaît les constructions syntaxiques des phrases en cours de frappe pour signaler à l'auteur des erreurs potentielles, une nouvelle génération d'ateliers de modélisation peut identifier des structures de patterns dans les diagrammes en cours de description ce qui éviterait de souvent « réinventer la roue » et limiterait les erreurs récurrentes en modélisation. Oualid Khayati, dans son travail de recherche sur des modèles formels de recherche de composants, a mis en place une technique originale de recherche de composants basée sur des appariements structurels qui permet de préfigurer une certaine intelligence pour de futurs éditeurs de diagrammes (Oualid Khayati, 2005). La spécification et la formalisation de démarches de développement « à la demande », basées sur deux paradigmes, des composants métier pour les modèles produits et des patrons processus pour l'organisation générique de démarches, peuvent maintenant passer du stade académique à des usages professionnels dans la mesure où des outils appropriés et puissants seront disponibles (Hassine, 2005). Les perspectives en termes d'ingénierie des SI et de modèles de processus sont en termes de modèles contextuels (Jarke, 1999) et de modèles intentionnels Rolland, 1999) qui supplantent les modèles classiques orientés activités que nous avons résumés ci-dessus (cf. section 4.2). Ces nouveaux modèles de processus seront au cœur de nouveaux environnements CAME (Computer Aided Method Engineering) pour supporter le développement de méthodes d'ingénierie des SI en complément aux ateliers de génie logiciel utilisés plus directement pour la modélisation des SI et le développement des logiciels.

Toute assistance automatisée a néanmoins ses limites : « L'ordinateur, langue d'Esope de la science moderne, permet à certains chercheurs dénués d'idées originales d'écrire des programmes à l'aide de lois déjà connues et de faire des simulations – avec l'espoir que la machine résoudra seule les problèmes sur lesquels ils ont buté, et qu'elle fera de nouvelles découvertes à leur place. Avec une telle démarche, on ne découvre pas grand-chose de nouveau, car il y manque les deux piliers de la science : l'observation de la nature et la conceptualisation originale de cette observation » (Claude Allègre, 2005). Nous voilà prévenus, les ordinateurs grâce à leurs capacités de mémorisation et de calcul restent de puissants moyens techniques pour appréhender des systèmes compliqués, alors que les modèles et les diagrammes que nous utilisons en ingénierie des SI, même s'ils sont instrumentés dans des ateliers d'ingénierie efficaces, ne sont que des moyens pour aider à l'observation et à la conceptualisation de SI complexes. L'une des difficultés majeures de la modélisation est inhérente au fait qu'un modèle et sa représentation diagrammatique sont une sorte d'« ombre du système » sans en connaître suffisamment bien la source de lumière (point de vue d'une catégorie d'utilisateurs, cliché d'une activité d'ingénierie, etc.). Ainsi croyant tout savoir d'un système, n'en avons-nous qu'une représentation partielle, un peu comme dans l'allégorie de la caverne et le théâtre d'ombres de Platon (Livre VII - La République).

Bibliographie

Abrial, J. R. (1996). *The B-Book*. Cambridge University Press.

Allègre, C. (2005). *Dictionnaire amoureux de la science*. Plon/Fayard.

Ambler, S. W. (1998). *Process Patterns : building large scale systems using object technology*. SIGS Books, Cambridge University Press.

Booch, G . Rumbaugh, J., Jacobson, I. (1998). *The Unified Modeling Language User Guide*. Addison-Wesley.

Boehm, B. W. (1976). A spiral model of software development. *ACM, SIGSOFT, vol. 11, n°4*.

Codd, E.F. (1970). A Relational Model of Data for large Shared Data Banks. *Communications of ACM, Vol. 13, N°6.*

Dijkstra, E. W. (1976). *A discipline of programming*. Prentice-Hall.

Fourastié, J. (1966). *Les conditions de l'esprit scientifique*. Collection Idées, Editions Gallimard.

Gamma, E. Helm, R., Johnson, R., Vlissides, J. (1995). *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Gzara, L., Rieu, D., Tollenaere, M. (2000). Patterns Engineering for Reuse at Product Information System Development. *Requirements Engineering Journal, Vol. 5, N°3*, Springer-Verlag.

Hassine, I. (2005). *Spécification et formalisation des démarches de développement à base de composants métier : la démarche SYMPHONY.* Thèse d'informatique de l'Institut National Polytechnique de Grenoble.

Henderson-Sellers, B., Edwards, J-M. (1990). The object-oriented software life cycle. *CACM, Vol. 33, N°9.*

Jacobson, I., Booch, G., Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison-Wesley.

Jarke, M., Rolland, C., Sutcliffe, A., Domges, R. (1999). *The NATURE Requirements Engineering*. Shaker Verlag, Aachen.

Khayati, O. (2005). *Modèles formels et outils génériques pour la gestion et la recherche de composants.* Thèse d'informatique de l'Institut National Polytechnique de Grenoble.

Kruchten, P. (1995). The 4+1 View Model of Architecture. *IEEE Software.*

Larousse, (1961). *Grand Larousse Encyclopédique.* Librairie Larousse.

Le Moigne, J.-L. (1990). *La modélisation des systèmes complexes.* Dunod.

Martin, J. (1982). *Application Developpement Without Programmers.* Prentice-Hall.

Martin, J. (1985). *Manifeste pour un système d'information*. Les Editions d'Organisation.

Minsky, M. L. (1968). Matter, Mind and Models. in *Semantic Information Processing*, MIT Press.

Morand, B. (2004). *Logique de la conception – figures de sémiotique générale.* d'après Charles S. Peirce, L'Harmattan.

Muller, P.-A. (1997). *Modélisation objet avec UML*. Eyrolles.

Nanci, D., Espinasse, B., Cohen, B., Heckenroth, H. (1992). *Ingénierie des systèmes d'information avec Merise – vers une deuxième génération.* SYBEX.

OMG (1997). *Meta-Object Facility – MOF*. Object Management Group.

OMG (1999). *Common Warehouse MetaModel – CWM* IBM and Co-submitters, Object Management Group.

OMG (2001). *Software Processing Engineering Metamodel Spécification – SPEM*. Object Management Group.

OMG (2005). *MDA – Model Driven Architecture*. Object Management Group.

Rieu, D., Giraudin, J.-P. et al. (1997). Objets et métamodélisation. in Oussalah, M. *Ingénierie objet – concepts et techniques.* InterEditions.

Odell, J.J. (1994). Six different kinds of composition. *Journal of Object-Oriented Programming, Vol. 6, N°8.*

Oussalah, M. et al. (2005). *Ingénierie des composants – concepts, techniques et outils*. Editions Vuibert Informatique.

Parnas, D. (1972). On the criteria to be used in decomposing systems in modules. *Communications of the ACM, Vol. 15, N°2.*

Rocques, P., Vallée, F. (2001). *UML en action – De l'analyse des besoins à la conception en Java*. Eyrolles.

Rolland, C., Prakash, N., Benjamen, A., (1999). « A multi-model view of process modelling », *Requirements Engineering Journal. Vol. 4, N°4.*

Rolland, C. (2005). L'ingénierie des méthodes : une visite guidée. in *e-TI – e-revue en Technologies de l'Information, N°1.*

Royce, W.W. (1970). Managing the development of large software systems. *IEEE WESCON*.

Simon, H. A. (1969). *The Sciences of the Artificial.* MIT Press.

Smith, J.M., Smith, D.C.P. (1977). Database abstractions : Aggregation and Generalization. *ACM TODS, Vol. 2, N°2.*

Spivey, J.M. (1989). *The Z notation : a reference manual.* Prentice-Hall.

Winston, M.E., Chaffin, R., Hermann, D.J. (1987). A Taxonomy of Part-Whole Relations. *Cognitive Science, N°11*.

Wirfs-Brock, R., Wilkerson, B., Weiner, L. (1990). *Designing Object-Oriented Software*. Prentice-Hall.

Pour citer cet article

Jean-Pierre GIRAUDIN. «Complexité des systèmes d'information et de leur ingénierie». e-TI - la revue électronique des technologies d'information, Numéro 3, 9 mai 2007, http://www.revue-eti.netdocument.php?id=1180.