Chapitre 1

Les modèles au cœur du développement de logiciel

> « Pour un observateur A, M est un modèle de l'objet O, si M aide A à répondre aux questions qu'il se pose sur O » [Min68].

1.1 Principes généraux

Après l'approche objet des années 80, l'ingénierie du logiciel s'oriente aujourd'hui vers l'ingénierie dirigée par les modèles (IDM). Comme décrit dans [Béz04b, Béz05], cette nouvelle approche peut être considérée à la fois en *continuité* et en *rupture* avec les précédents travaux. Tout d'abord en continuité car c'est la technologie objet qui a déclenché l'évolution vers les modèles. En effet, une fois acquise la conception des systèmes informatiques sous la forme d'objets communicant entre eux, il s'est posé la question de les classifier en fonction de leurs différentes origines (objets métiers, techniques, etc.). L'IDM vise donc, de manière plus radicale que pouvaient l'être les approches des *patterns* [GHJ95] et des *aspects* [KLM+97], à fournir un grand nombre de modèles pour exprimer séparément chacune des préoccupations des utilisateurs, des concepteurs, des architectes, etc. C'est par ce principe de base fondamentalement différent que l'IDM peut être considérée en rupture par rapport aux travaux de l'approche objet.

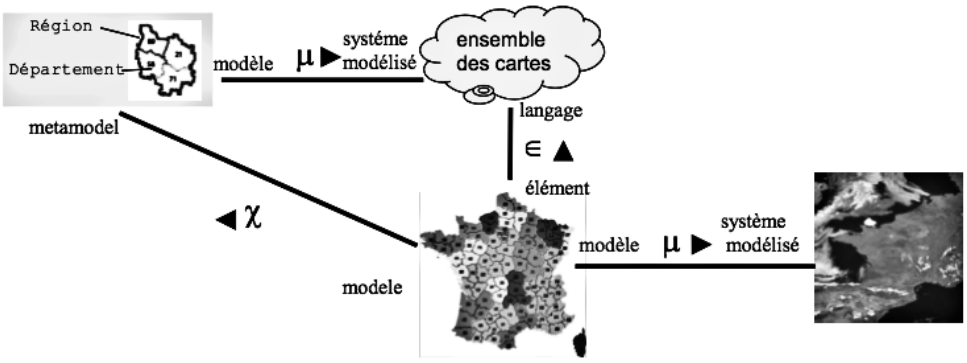

FIGURE 1.1 – Relations entre système, modèle, métamodèle et langage (extrait de [FEB06])

1.1.1 Modèle

Alors que l'approche objet est fondée sur les relations essentielles d'*instanciation* et d'*héritage*, l'IDM est basée sur un autre jeu de concepts et de relations. Le concept central de l'IDM est la notion de *modèle*, pour laquelle il n'existe pas à ce jour de définition universelle. Néanmoins, de nombreux travaux s'accordent à un relatif consensus d'une certaine compréhension. A partir des travaux de l'OMG [1], de Muller *et al.* [MFBC10], de Bézivin *et al.* [BG01] et de Seidewitz [Sei03], nous considérerons la définition suivante d'un modèle.

Définition (*Modèle*) Un modèle est un ensemble de faits caractérisant un aspect d'un système dans un objectif donné. Un modèle représente donc un système selon un certain point de vue, à un niveau d'abstraction facilitant par exemple la conception et la validation de cet aspect particulier du système.

On déduit de cette définition la première relation majeure de l'IDM, entre le modèle et le système qu'il représente, appelée *représentationDe* dans [AK03, Béz04a, Sei03], et nommée μ sur la figure 1.1, où nous reprenons l'exemple utilisé dans [FEB06], qui s'appuie sur la cartographie pour illustrer l'IDM.

Dans cet exemple, une carte est un modèle (une représentation) de la réalité, avec une intention particulière (carte routière, administrative, des reliefs, etc.).

1. *The Object Management Group (OMG)*, cf. `http://www.omg.org/`.

« En cet empire, l´Art de la Cartographie fut poussé à une telle Perfection que la Carte d'une seule Province occupait toute une ville et la Carte de l'Empire toute une Province. Avec le temps, ces Cartes Démesurées cessèrent de donner satisfaction et les Collèges de Cartographes levèrent une Carte de l'Empire, qui avait le Format de l'Empire et qui coïncidait avec lui, point par point. Moins passionnées pour l'Etude de la Cartographie, les Générations Suivantes réfléchirent que cette Carte Dilatée était inutile et, non sans impiété, elle l´abandonnèrent à l´Inclémence du Soleil et des Hivers. Dans les Déserts de l'Ouest, subsistent des Ruines très abimées de la Carte. Des Animaux et des Mendiants les habitent. Dans tout le Pays, il n'y a plus d´autre trace des Disciplines Géographiques. »

FIGURE 1.2 – Citation attribuée par Jorge Luis Borges à un auteur de son invention et publiée en français dans le recueil « L'auteur et autres textes »

« He had bought a large map representing the sea,
without the least vestige of land :
and the crew were much pleased when they found it to be
a map they could all understand.

"What's the good of Mercator's North Poles and Equators,
Tropics, Zones, and Meridian Lines ?"
So the Bellman would cry : and the crew would reply
"They are merely conventional signs !"

Other maps are such shapes, with their islands and capes !
But we've got our brave Captain to thank :
(so the crew would protest) "that he's bought us the best–
a perfect and absolute blank !" »

FIGURE 1.3 – Citation de Carroll Lewis dans « The Hunting of the Snark »

Notons que même si cette relation a fait l'objet de nombreuses réflexions, il reste toutefois difficile de répondre à la question « qu'est ce qu'un bon modèle ? » et donc de formaliser précisément la relation μ.

Sans tomber dans les excès de la carte blanche de Lewis Carroll (figure 1.3) ou de la carte à l'échelle 1 :1 de Borges (figure 1.2), toute la difficulté de l'activité de modélisation est de trouver le bon niveau d'abstraction pour représenter un système selon un ensemble de modèles facilitant sa conception et sa validation.

Un modèle doit donc par définition être une abstraction pertinente du système qu'il modélise pour un point de vue particulier, c'est-à-dire qu'il doit être suffisant et nécessaire pour répondre aux questions sous-jacentes à ce point de vue particulier, en lieu et place du système qu'il représente et exactement de la même façon que le système aurait répondu lui-même. Ce principe, dit de *substituabilité*, assure que le modèle peut se substituer au système pour permettre d'analyser de manière plus abstraite certaines de ses propriétés [Min68].

Définition (*Principe de substituabilité*) Un modèle doit capturer les informations nécessaires et suffisantes pour permettre de répondre aux questions que l'on se pose sur un aspect du système qu'il représente, exactement de la même façon que le système lui-même aurait répondu.

1.1.2 Modèles, aspects et lignes de produits

La modélisation est parfois vue comme le moyen d'exprimer une solution à un plus haut niveau d'abstraction que le code. Cette vision étroite de la modélisation a eu son utilité dans le passé (langages d'assemblages abstrayant du code machine, langages de troisième génération abstrayant les langages d'assemblages, etc.) et est encore utile de nos jours, par exemple pour obtenir une vue holistique d'un gros programme C++. Mais la modélisation ne s'arrête pas à ça.

La modélisation est en effet la pierre angulaire de toute activité scientifique (en lien avec la validation des modèles obtenus vis-à-vis d'expérimentations effectuées dans le monde réel). À cet égard, la spécificité du domaine de l'ingénierie est que les ingénieurs construisent des modèles d'artefacts qui en général n'existent pas encore (ne serait-ce que parce que le but ultime est de construire ces artefacts).

En ingénierie on souhaite en général décomposer un système complexe en autant de modèles que nécessaire pour aborder efficacement toutes les

préoccupations pertinentes [BC04]. Ces modèles peuvent être exprimés avec un langage de modélisation généraliste comme UML, ou avec des langages de modélisation spécifiques à des domaines (DSML en anglais pour *Domain Specific Modeling Language*) lorsque cela semble plus approprié (voir figure 1.4). Chacun de ces modèles peut être vu comme l'abstraction d'un aspect de la réalité pour gérer une préoccupation particulière. L'expression explicite de solutions efficaces pour gérer ces préoccupations permet d'établir des compromis adéquats relativement tôt dans le cycle de vie du logiciel mais aussi de gérer efficacement des variantes du système (notion de ligne de produits) [PBvdL05].

Lignes de produits : il est de plus en plus rare de nos jour de construire des systèmes informatiques qui ne soient pas déclinés en de multiples variantes, et ceci au delà des habituelles différences de fonctionnalités dues à des choix marketing. En effet il faut aussi tenir compte de différences dans le matériel (cartes graphiques, dispositifs d'affichage, variété de périphériques d'interaction etc.), dans les systèmes d'exploitations et autres intergiciels, des différences liées aux aspects législatifs, normatifs ou culturels, et bien sûr des préférences des utilisateurs. Ces points de variations sont souvent assez orthogonaux, ce qui engendre une explosion combinatoire du nombre de variantes possibles d'un système. De plus, comme chacune de ces variantes peut avoir des versions successives, une véritable vision bi-dimentionnelle (axe des variantes et axe temporel) est nécessaire pour appréhender dans sa globalité ce concept de ligne de produits

Notons que dans la communauté de la programmation orientée aspect, la notion d'aspects est définie de manière sensiblement plus restrictive comme la modularisation de préoccupations transverses [FF00]. Si nous avons en effet déjà un paradigme de décomposition (comme l'orientation objet), il existe toujours de nombreuses classes de préoccupations pour lesquels une encapsulation dans des modules n'est pas possible (d'où leur nom de préoccupations transverses) : par exemple certains types de fonctionnalités (comme l'authentification) sont par construction difficiles à encapsuler, ce qui est aussi le cas pour un certain nombre d'exigences non-fonctionnelles qui sont de manière inhérente transverses, comme la sécurité, la disponibilité, la répartition, la gestion des ressources, ou les contraintes temps-réels.

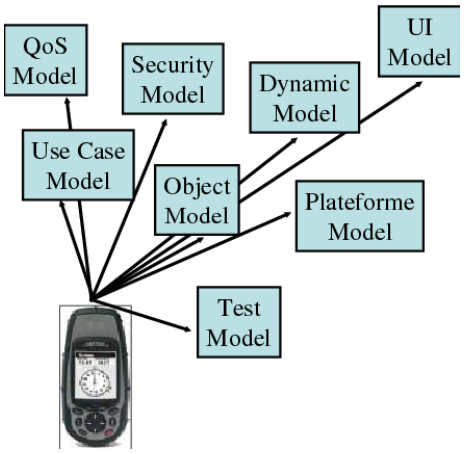

FIGURE 1.4 – Modéliser plusieurs aspects

Maintenant que la notion d'aspect devient de plus en plus utilisée au-delà du monde de la programmation, il y a une acceptation grandissante pour une définition plus large où un aspect est simplement une préoccupation qui peut être modularisée. L'identification systématique de ces aspects, leur modularisation et leur composition ont ainsi pour objectif général d'améliorer notre capacité à raisonner sur le domaine du problème et sur la solution correspondante. Cela doit permettre de réduire la taille des modèles de logiciels et du code des applications, et donc les coûts de développement et le temps de maintenance.

En substance, la modélisation est une activité visant à séparer les préoccupations dans le domaine du problème, activité que l'on appelle souvent *analyse*. Si les solutions à ces préoccupations peuvent être décrites comme des aspects, le processus de conception peut alors être caractérisé comme le tissage (*weaving* en anglais) de ces aspects dans un modèle de conception détaillée (aussi appelé l'espace de la solution, voir figure 1.5) [JPW+02]. Ceci n'est bien sûr pas nouveau : c'est en réalité ce que les concepteurs de logiciels ont toujours fait. Le plus souvent cependant, la plupart des aspects ne sont pas *explicites*, ou quand ils le sont, c'est le plus souvent sous la forme d'une définition informelle. Ceci conduit le concepteur à effectuer ce tissage entre préoccupations en quelque sorte *de tête*. Et ensuite à produire la conception détaillée résultante comme un gros programme où les différentes préoccupations s'entremêlent (et ceci même si un paradigme de décomposition, comme l'orienté objet, est utilisé). Si cela fonctionne sur des projets de taille ou de complexité réduite, il est bien

connu que ça devient extrêmement difficile à gérer lorsque la complexité des projets augmente.

Insistons sur le point que le défi abordé ici n'est pas *comment* concevoir un système pour prendre en compte un aspect particulier : il existe en effet dans l'industrie d'importants savoir-faire, souvent d'ailleurs capturés sous la forme de patrons de conception. Bien sûr, prendre en compte plusieurs aspects à la fois est un peu plus complexe, mais de nombreux projets complexes menés à bien dans l'industrie sont là pour nous montrer que les ingénieurs (la plupart du temps) y arrivent. Le vrai défi est plutôt lié, dans un contexte où l'agilité des développements prime, au besoin fréquent de s'adapter au changement des exigences. Ce changement se traduit par un nouveau choix de variante ou de version d'une variante par aspect du système, et nécessite alors d'être recomposé pour obtenir une nouvelle déclinaison (ou configuration) du système, c'est à dire un nouveau produit dans une ligne de produits. Bien sûr, cette nouvelle déclinaison du système doit être obtenue rapidement, de manière fiable, et à bon marché. De ce point de vue, faire la composition manuelle de chaque aspect n'est plus une panacée.

L'IDM ne propose pas de résoudre ce problème directement, mais simplement de mécaniser et de reproduire le processus que les concepteurs expérimentés suivent à la main [HJPP02]. L'idée est alors que lorsqu'une nouvelle variante a besoin d'être dérivée dans une ligne de produits, on peut automatiquement rejouer la plus grosse partie de ce processus de conception, en apportant juste quelques petites modifications [LMV+07].

Usuellement en sciences, un modèle a une nature différente de la réalité qu'il modélise (par exemple le dessin d'un pont vs. le pont lui-même). C'est seulement en logiciel et en linguistique qu'un modèle a la même nature que la réalité modélisée. En logiciel en particulier, ceci ouvre la possibilité de dériver automatiquement du logiciel depuis un modèle, c'est-à-dire de rendre les modèles "productifs" en automatisant ce processus de tissage. Cette automatisation n'est bien sûr possible que si les modèles ne sont plus informels. Cela implique que le processus de tissage soit lui-même décrit comme un programme manipulant ces modèles pour produire une conception détaillée [MFJ05a]. Cette dernière pourra être finalement transformée en code, en configuration de logiciel, en suite de tests, etc. [PJJ+07].

Cette notion de tissage de modèles est donc au cœur de la conception dirigée par les modèles.

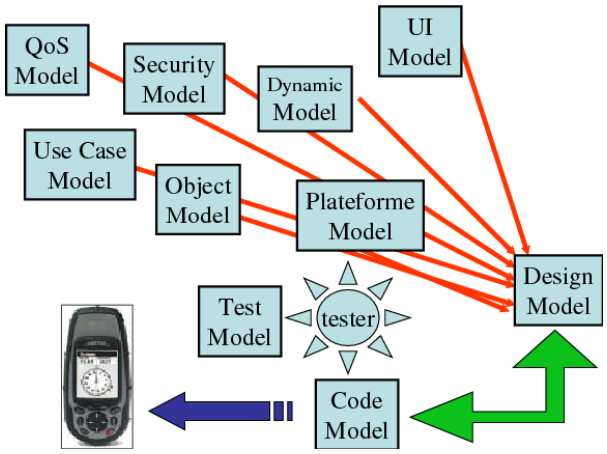

FIGURE 1.5 – Concevoir revient à tisser des modèles

1.1.3 Métamodèle

La notion de modèle dans l'IDM fait explicitement référence à la définition des langages utilisés pour les construire. En effet, pour qu'un modèle soit productif, il doit pouvoir être manipulé par une machine. Le langage dans lequel ce modèle est exprimé doit donc être explicitement défini. De manière naturelle, la définition d'un langage de modélisation a pris la forme d'un modèle, appelé *métamodèle*.

Définition (*Métamodèle*) Un métamodèle est un modèle qui définit le langage d'expression d'un modèle [OMG06a], c'est-à-dire le langage de modélisation.

En pratique, un métamodèle permet de capitaliser un domaine de connaissances. Il devient naturellement le cœur d'un outillage visant à systématiser certaines étapes de développement.

La notion de métamodèle conduit à l'identification d'une seconde relation, liant le modèle et le langage utilisé pour le construire, appelée *conformeA* et nommée χ sur la figure 1.1.

Définition (*conformeA* (χ)) Un modèle est dit conforme à un métamodèle si chacun de ses éléments (objets et relations) est instance d'un élément du métamodèle, et s'il respecte l'ensemble des propriétés (*e.g.,* contraintes d'invariant) exprimées sur le métamodèle.