CHAP 1: INTRODUCTION A LA MODELISATION OBJET

1. Introduction

UML (*Unified Modeling Language* en anglais, soit langage de modélisation objet unifié) est né de la fusion des trois méthodes qui s'imposaient dans le domaine de la modélisation objet au milieu des années 1990 : OMT, Booch et OOSE. D'importants acteurs industriels (IBM, Microsoft, Oracle, DEC, HP, Rational, Unisys etc.) s'associent alors à l'effort et proposent UML 1.0 à l'OMG (*Object Management Group*) qui l'accepte en novembre 1997 dans sa version 1.1.

2. Pourquoi et comment modéliser

Un modèle est une représentation abstraite et simplifier du monde réel en vue de le décrire, de l'expliquer ou de le prévoir.

Concrètement un modèle permet de réduire la complexité d'un phénomène en éliminant les détails qui n'influencent pas le comportement de manière significative.

Modéliser un système avant sa réalisation permet de mieux comprendre le fonctionnement du systeme. C'est également un bon moyen de maîtriser sa complexité et d'assurer sa cohérence. Un modèle est un langage commun, précis, qui est connu par tous les membres de l'équipe et il est donc, à ce titre, un vecteur privilégié pour communiquer.

3. Le cycle de vie d'un logiciel

Le *cycle de vie d'un logiciel* (en anglais *software lifecycle*), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés, et la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre.

Le cycle de vie du logiciel comprend généralement les étapes suivantes :

Définition des objectifs :
Cet étape consiste à définir la finalité du projet et son inscription dans une stratégie globale.
Analyse des besoins et faisabilité :
C'est-à-dire l'expression, le recueil et la formalisation des besoins du demandeur (le client) et de l'ensemble des contraintes, puis l'estimation de la faisabilité de ces besoins.

Spécifications ou conception générale:

Il s'agit de l'élaboration des spécifications de l'architecture générale du logiciel.

Conception détaillée :
Cette étape consiste à définir précisément chaque sous-ensemble du logiciel.

Codage (Implémentation ou programmation) :
C'est la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.

Tests unitaires:
Ils permettent de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.

Intégration:
L'objectif est de s'assurer de l'interfaçage des différents éléments (modules) du logiciel. Elle fait l'objet de tests d'intégration consignés dans un document.

Qualification (ou recette):
C'est-à-dire la vérification de la conformité du logiciel aux spécifications initiales.

Documentation:
Elle vise à produire les informations nécessaires pour l'utilisation du logiciel et pour des développements ultérieurs.

Mise en production:
C'est le déploiement sur site du logiciel.

Maintenance:
Elle comprend toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

4. L'approche orientée objet

L'approche orientée objet considère le logiciel comme une collection d'objets dissociés, identifiés et possédant des caractéristiques. Une caractéristique est soit un attribut (*i.e.* une donnée caractérisant l'état de l'objet), soit une entité comportementale de l'objet (*i.e.* une fonction). La fonctionnalité du logiciel émerge alors de l'interaction entre les différents objets qui le constituent. L'une des particularités de cette approche est qu'elle rapproche les données et leurs traitements associés au sein d'un unique objet.

Comme nous venons de le dire, un objet est caractérisé par plusieurs notions :

L'identité:
L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état. On construit généralement cette identité grâce à un identifiant découlant naturellement du problème (par exemple un produit pourra être repéré par un code, une voiture par un numéro de série, etc.)

Les attributs:
Il s'agit des données caractérisant l'objet. Ce sont des variables stockant des informations sur l'état de l'objet.

Les méthodes:
Les méthodes d'un objet caractérisent son comportement, c'est-à-dire l'ensemble des actions (appelées opérations) que l'objet est à même de réaliser. Ces opérations permettent de faire réagir l'objet aux sollicitations extérieures (ou d'agir sur les autres objets). De plus, les opérations sont étroitement liées aux attributs, car leurs actions peuvent dépendre des valeurs des attributs, ou bien les modifier.

5. UML en œuvre

UML n'est pas une méthode (c'est-à-dire une description normative des étapes de la modélisation).

UML comporte plusieurs diagrammes représentant autant de vue distinctes pour représenter des concepts particuliers du système d'information.

Voici une liste des diagrammes que nous aurons à étudier:

Diagrammes structurels ou diagrammes statiques (*UML Structure*)

- diagramme de classes (*Class diagram*)
- diagramme d'objets (*Object diagram*)
- diagramme de composants (*Component diagram*)
- diagramme de déploiement (*Deployment diagram*)
- diagramme de paquetages (*Package diagram*)
- diagramme de structures composites (*Composite structure diagram*)

Diagrammes comportementaux ou diagrammes dynamiques (*UML Behavior*)

- diagramme de cas d'utilisation (*Use case diagram*)
- diagramme d'activités (*Activity diagram*)
- diagramme d'états-transitions (*State machine diagram*)
- **Diagrammes d'interaction (*Interaction diagram*)**
  - diagramme de séquence (*Sequence diagram*)
  - diagramme de communication (*Communication diagram*)
  - diagramme global d'interaction (*Interaction overview diagram*)
  - diagramme de temps (*Timing diagram*)