Table des matières

Chapitre II : l'approche orientée objets et présentation d'UML 2

I

1. Modélisation

La démarche permettant à passer des besoins au code est bien connue sous le nom de **modélisation**. Le recours à cette dernière est depuis longtemps une pratique indispensable au développement logiciel. La modélisation est en effet une représentation abstraite d'un système qui permet d'en faciliter l'étude, de mieux comprendre le fonctionnement du système à réaliser. De plus, elle constitue un outil majeur de communication au sein de l'équipe de réalisation d'un projet. En outre, les systèmes sont souvent trop complexes, leur compréhension et leur maîtrise dépassent les capacités d'un seul individu. La construction d'un modèle abstrait permet de remédier à ce problème.

Définition : Modèle

Un modèle est une représentation abstraite et simplifiée (i.e. qui exclut certains détails), d'une entité (phénomène, processus, système, etc.) du monde réel en vue de le décrire, de l'expliquer ou de le prévoir.

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

- De visualiser le système comme il est ou comme il devrait l'être.
- De valider le modèle vis à vis des clients.
- De spécifier les structures de données et le comportement du système.
- De fournir un guide pour la construction du système.
- De documenter le système et les décisions prises

1.1. Approches de la modélisation

1.1.1. Approche fonctionnelle

les méthodes fonctionnelles (également appelées méthodes de modélisation structurées) sont des approches utilisant des procédures et des fonctions. En effet les fonctions nécessaires à l'obtention du résultat sont identifiées. De plus, la fonction globale est décomposé jusqu'à obtenir des fonctions simples , maîtrisable (sous programmes)à appréhender et donc à programmer. Ces approches proposent une approche hiérarchique descendante et modulaire.

1.1.2. Approche orientée objets

L'approche orientée objet tourne autour d'une brique de base "l'objet "ceci est a mettre en opposition de la programmation procédurale par exemple qui est constituée de procédures et fonctions agissant sur des données dissociées. L'approche 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. Cette approche est promue d'une approche ascendante.

la modélisation objet apporte :

- Plus grande indépendance du modèle par rapport aux fonctionnalités demandées.
- Des fonctionnalités peuvent être rajoutées ou modifiées, le modèle objet ne change pas.
- Plus proche du monde réel.

1.2. Langages de modélisation

Un langage de modélisation définit:

- La sémantique des concepts
- Une notation pour la représentation de concepts
- Des règles de construction et d'utilisation des concepts.

il existe des langages à différents niveaux de formalisation à savoir :

- Langages formels (Z,B,VDM) : le plus souvent mathématiques, au grand pouvoir d'expression et permettant des preuves formelles sur les spécifications
- Langages semi-formels (MERISE, UML...) : le plus souvent graphiques, au pouvoir d'expression moindre mais plus faciles d'emploi.

L'industrie du logiciel dispose de nombreux langages de modélisation :

- Adaptés aux systèmes procéduraux (MERISE...)
- Adaptés aux systèmes temps réel (ROOM, SADT...) ;
- Adaptés aux systèmes à objets (OMT, Booch, UML...).

2. Concepts de l'approche orientée objets

L'approche objet occupe une place très importante dans le génie logiciel. En effet, cette section a pour objectif de présenter l'essentiel des concepts objet nécessaire à une bonne compréhension d'UML. Les concepts importants à bien maîtriser sont les suivant :

2.1. Concept d'objet

Un objet représente une entité du monde réel ou virtuel qui se caractérise par un ensemble de **propriétés** (attributs), **des états** significatifs et un **comportement**. L'état d'un objet correspond aux valeurs de tous ses attributs à un instant donnée quant au comportement d'un objet est caractérisé par l'ensemble des opérations qu'il peut exécuter en réaction aux messages provenant des autres objets. L'objet a un **identifiant unique** qui permets de le distinguer des autres objets.

Exemple

Considérons l'employé « Mohammed », n°1234, embauché en étant ingénieur en informatique travaillant au sein de l'entreprise « ENTR ».

Cet objet réel est caractérisé par la liste de ses attributs en son état est représenté par les valeurs de ses attributs :

Liste des attributs	**valeurs**
N° employé	1234
NOM	Mohammed
grade	ingénieur
lieu de travail	l'entreprise « ENTR»

Son comportement est caractérisé par les opérations qu'il peut exécuter. Il peut avoir les opérations suivantes : Changer de grade, Changer de lieu de travail, développer des applications,etc.

2.2. Concept de classe

Une classe est l'abstraction d'un ensemble d'objets qui possèdent une même structure (liste des attributs) et un même comportement (liste des opérations).

Exemple

Nom de classe : Employé

Attributs:

Numéro

Nom

Grade

Lieu de travail

Opérations

Engager un employé

Départ d'un employé

Remarque

Un objet est qualifié comme une instance d'une et une seule classe.

Une classe abstraite : est une classe qui n'a pas d'instances.

2.3. Encapsulation et interface

L'approche objet se caractérise par le regroupement dans une même classe de la description de la structure des attributs et de la description des opérations. Ce regroupement des deux description porte le nom **d'encapsulation données- traitement**. Autrement dit les données ne sont accessibles qu'a partir d'opérations définies dans la classe. Ce principe d'encapsulation a comme avantages de renforcer l'autonomie et l'indépendance de chaque classe

L'ensemble des opérations d'une classe rendu visible aux autres classes porte le nom **d'interface**.

2.4. Association et agrégation entres les classes

Une association représente une relation entre plusieurs classes. Elle correspond à l'abstraction des liens existant entre les objets dans le monde réel. Les multiplicités (cardinalités) et les rôles des objets participant aux relations complètent la description d'une association.

L'agrégation est une forme particulière d'association entre plusieurs classes. Elle exprime le fait qu'une classe est constitué d'une ou plusieurs autres classes.

2.5. Généralisation et spécialisation de classe

La généralisation de classes consiste à factoriser dans une classe appelée **super-classe**, les attributs et/ou les opérations des classes considérées. Appliquée à l'ensemble des classes, elle permet de créer une hiérarchie des classes.

La spécialisation représente la démarche inverse de la généralisation puisqu'elle consiste à créer à partir d'une classe, plusieurs classe spécialisées. Chaque sous-classe créée est dite spécialisée puisqu'elle comporte en plus des attributs ou opérations de la super-classe des attributs ou opérations qui lui sont propres. La spécialisation de classe se construit en deux temps : d'abord par héritage des opérations/ attributs d'une super classe et ensuite par ajout d'opérations et/ou attributs spécifiques à la sous-classe

2.6. Polymorphisme

le polymorphisme est la capacité donnée à une même opération de s'exécuter différemment suivant le contexte de la classe ou elle se trouve. il est également possible de définir une opération dans une super-classe peut s'exécuter de manière différentes selon la sous-classe ou elle est héritée .

Exemple : Soit la classe employé et une sous-classe responsable

Nom de classe : employé

Attributs : Numéro , Nom , Salaire de base

Opérations : Calculer-salaire ()

&

Nom de la sous-classe : responsable

Attributs : niveau d'encadrement / responsabilité

Opérations : calculer-salaire()

En fait, le principe de calcul de salaire étant de calculer pour chaque type d'employé une prime spécifique en fonction soit du niveau d'encadrement, soit du niveau de spécialisation selon le type d'employé.

2.7. Persistance

la persistance est la propriété donnée à un objet de continuer à exister après la fin de l'exécution du programme qui l'a crée.

3. Présentation UML 2

3.1. c'est quoi UML ?

Aujourd'hui, le standard industriel de modélisation objet est UML. Il est sous l'entière responsabilité de L'OMG. UML est un langage de modélisation graphique et textuel destiné à:

- Comprendre et décrire des besoin ;
- Spécifier de manière précise et complète, sans ambiguïté ;
- Documenter des systèmes : les différents diagrammes, notes, contraintes, exigences seront présenter dans un document ;
- Concevoir des solutions (Construire les classes, les relations,) ;
- Visualiser (chaque symbole graphique a une sémantique) ;
- Communiquer des points de vue.

UML unifie à la fois les notations et les concepts orientés objets. Il ne s'agit pas d'une simple notation graphique, car les différents concepts présentés par un type de diagramme ont une signification et une sémantique. De plus, l'UML est une norme adopté par toutes les méthodes objet.

Les briques de base d'UML sont :

- Les éléments : sont les abstractions des modèles
- Les relations : sont les liens entre ces abstractions.
- Les diagrammes: sont les regroupements des éléments et des relations.

Rappel

OMG (object Management Group) est un groupement d'industriels dont l'objectif est la standardisation autour des technologies objet, afin de garantir l'interopérabilité des développement. L'OMG comprend actuellement plus de 800 membres, dont les principaux acteurs de l'industrie informatique (Sun, IBN, Microsoft, etc.) mais aussi les plus grandes entreprises utilisatrices dans tous les secteurs d'activité.

3.2. Historique d'UML

Une cinquantaine de méthodes d'analyse et de conception qui existaient au début des années 90, seulement trois d'entre elles se sont détachées nettement au bout de quelques années. En effet, la volonté de converger vers une méthode unifiée était déjà bien réelle et pour cette raison que les méthodes OMT, BOOCH et OOSE se démarquées des autres.

OMT (Object Modeling Technique) de James Rumbaugh et BOOCH de grady Booch ont été les deux méthodes les plus diffusées en France durant les années 90. Par ailleur, OOSE de Ivar Jacobson s'est aussi imposée dans le monde objet pour la partie formalisation des besoins.

Ces trois méthodes se sont rejoints dans l'objectifs de les fusionner de créer UML (Unified Methode Language).

UML est donc une norme de langage de modélisation objet qui à été publiée, dans sa première version , en novembre 1997 par L'OMG (Object Mangement Group), instance de normalisation internationale du domaine de l'objet.

En quelques années, UML s'est imposé comme un standard à utiliser en tant que langage de modélisation objet.

Les étapes de la diffusion d'UML peuvent se résumer comme suit :

1994-1996 : rapprochement des méthodes OMT, BOOCH, OOSE et naissance de la première version d4UML

1997 : version 1.1 d'UML adopté par l'OMG

1998-1999 : sortie des versions 1.2 à 1.3 d'UML

2000-2001 : sortie des dernières versions suiavntes 1.x.

2002-2003 : préparation de la V2.

2004 : sortie de la V2.1

2007 : sortie de la V2.1.1.

.

.

2015 : sortie de la version 2.5

2017 : sortie de la version 2.5.1

.

.

Remarque

Pour plus de détails, veuillez consulter la norme de référence [OMG2007]

3.3. Règles générales

UML propose un certain nombre de règles de représentations graphiques normalisées et un ensemble de concepts applicables à l'ensemble des diagrammes. Cet ensemble de règles et de concepts permet d'assurer un bon niveau de cohérence et d'homogénéité. Les principaux éléments généraux d'UML 2 sont comme suit :

3.3.1. Méta-modèle

UML respecte un certain nombre de règles sur les concepts ainsi que sur la syntaxe d'écriture et le formalisme de représentation graphique. L'ensemble de ces règles constitue en soi un langage de modélisation qui a fait l'objet d'un méta-modèle UML.

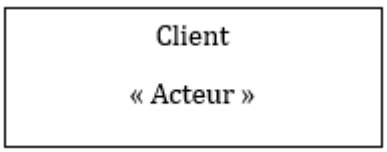
3.3.2. Stéréotype

Un stéréotype est un moyen de classer les éléments de la modélisation qui peut s'appliquer à n'importe quel concept d'UML.

En particulier, dans le diagramme de classe, le stéréotype permet de considérer de nouveaux types de classe.

Le nom du stéréotype est indiqué **entre guillemets**. Un acteur peut être vu comme un stéréotype particulier d'une classe appelée acteur.

L'exemple illustré dans la figure montre une classe Client stéréotypée comme « acteur ».



3.3.3. Valeur marquée

UML permet d'indiquer des valeurs particulières au niveau des éléments de modélisation et en particulier pour les attributs de classe. Une valeur marquée se définit au niveau méta-attribut.

La valeur marquée est **mise entre accolades** avec indication du nom et de la valeur `{persistance: string}` si l'on veut ajouter ce type d'attribut dans une classe .

3.3.4. Profil

Afin de donner la possibilité de spécialiser chaque application d'UML à son propre contexte, UML propose de définir un profil d'utilisation caractérisé principalement par la liste des stéréotypes, la liste des valeurs marquées et les contraintes spécifiées pour un projet donné.

3.3.5. Note

Une note correspond à un commentaire explicatif d'un élément d'UML.

Notation graphique d'une note

3.3.6. Contrainte

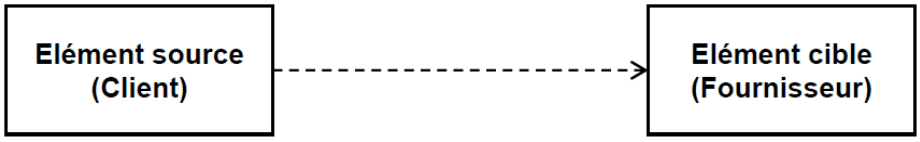
Une contrainte est une note ayant une valeur sémantique particulière pour un élément de la modélisation. Une contrainte s'écrit entre accolades {}. Dans le cas ou la contrainte concerne deux classes ou plus, celle-ci s'écrit l'intérieur d'une note

Remarque

en UML, un langage spécifique d'expression de contraintes est disponible, c'est le langage OCL(Object Contraint Language).

3.3.7. Relation de dépendance

La relation de dépendance, notée par une flèche pointillée, définit une relation d'utilisation unidirectionnelle entre deux éléments de modélisation, appelés respectivement source (exemple : client) et cible (exemple : fournisseur) de la relation.

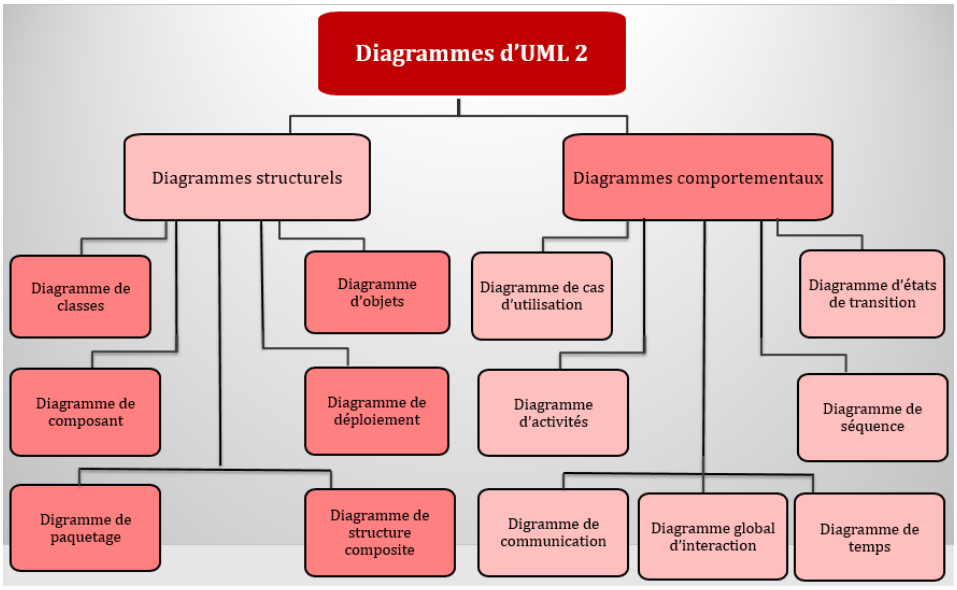


Remarque

Une relation de dépendance peut également être dotée d'une note et/ou d'une contrainte

3.4. Présentation générale des diagrammes

UML dans sa version 2 propose treize diagrammes qui peuvent utilisés dans la description d'un système. Chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Ces diagrammes sont regroupés dans deux grands ensembles comme l'illustre la figure suivante:



3.4.1. Les digrammes structurels

Ces diagrammes, au nombre de six, ont vocation à représenter l'aspect statique d'un système.

a) Diagramme de classes

Le diagramme de classes représente la description statique du système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements. C'est le diagramme pivot de l'ensemble de la modélisation d'un système.

b) Diagramme d'objets

Le diagramme d'objet permet de représenter des instances de classes et des liens entre instances.

c) Diagramme de composants

Le diagramme de composants représente les concepts de configuration logicielle. Il s'agit de montrer comment s'agencent les différents constituants (fichiers source, les packages de code ,etc.) du logiciel au niveau de l'implémentation d'un système.

d) Diagramme de déploiement

Le diagramme de déploiement décrit l'architecture technique d'un système avec une vue centrée sur la répartition de composants dans la configuration d'exploitation.

e) Diagramme de paquetage

Ce digramme donne une vue d'ensemble du système structuré en paquetage. Chaque paquetage représente un ensemble homogène d'éléments du système.

f) Diagramme de structure composite

Ce diagramme permet de décrire la structure interne d'un ensemble complexe composé par exemple de classes ou d'objets et de composants techniques. Ce diagramme met aussi l'accent sur les liens entre les sous-ensembles qui collaborent

3.4.2. Les diagrammes comportementaux

Les diagrammes comportementaux se focalisent sur la description de l'aspect dynamique du système à modéliser. Les sept diagrammes proposés par UML 2 sont :

a) Diagramme des cas d'utilisation

Ce diagramme est destiné à représenter les besoins des utilisateurs par rapport au système. Il constitue un des diagrammes les plus structurants dans l'analyse des besoins d'un système.

b) Diagramme d'état-transition

Ce diagramme montre les différents états des objets en réaction aux événements.

c) Diagramme d'activités

Ce diagramme donne une vision des enchaînements des activités propres à une opération ou à un cas d'utilisation. Il permet aussi de représenter les flots de contrôle et les flots de données.

d) Diagramme de séquence

Ce diagramme permet de d'écrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chnologie des opérations en interaction avec les objets.

e) Diagramme de communication

Ce diagramme est une autre représentation des scénarios de cas d'utilisation qui met plus l'accent sur les objets et les messages échangés.

f) Diagramme global d'interaction

Ce diagramme fournit une vue générale des interactions décrites dans le diagramme de séquence et les flots de contrôle décrits dans le diagramme d'activités.

g) Diagramme de temps :

Ce diagramme permet de représenter les états et les interactions d'objets dans un contexte ou le temps a une force influence sur le comportement du système à gérer.

3.4.3. Schéma d'ensemble des treize diagrammes d'UML2

Plusieurs schémas d'ensemble des diagrammes d'UML ont été proposées et ce, afin de donner quelques points de repères sur le positionnement et les liens entre tous les diagrammes.

Les diagrammes peuvent être regroupés selon leur finalité en quatre ensembles :

1. **Description du système** : cette catégorie regroupe diagramme de cas d'utilisation, diagramme de séquence, diagramme de communication
   diagramme d'activité, diagramme de classes, diagramme d'objet , diagramme d'état-transition et diagramme de temps ;
2. **Architecture technique** : cette catégorie regroupe diagramme de composant et le diagramme de déploiement ;

3. **Vues globales ou spécialisées** : cette catégorie regroupe diagramme global d'interaction et diagramme de structure composite
4. **Partition d'éléments de la modélisation** : cette catégorie comporte le diagramme de paquetage

Remarque

Quatorze diagrammes ont été défini dans la version 2.5.1 d'UML publié en décembre 2017 par OMG.

3.4.4. Utilitaires pour la modélisation UML

Les diagrammes UML pourront être réalisés avec un simple outil de dessin, un outil UML gratuit ou en version d'évaluation. Parmi les outils UML spécialisés, nous pouvons mentionner:

- Papyrus : https://www.eclipse.org/papyrus/
- StarUML: http://staruml.io/
- BoUML :http://boulk.io/
- Astah community http://astah.net/editions/community
- Power Designer (payant - version d'essai 30 jours)
- Visual Paradigm (payant - version d'essai 30 jours)
- Outil en ligne : https://www.draw.io/
- ArgoUML (Open source)
- PlantUM
- Etc. (voir http://uml.developpez.com/outils/).