Chapitre 2 standards UML

1-Introduction

Jusqu'au tout récemment, les systèmes logiciels, qu'ils soient orientes objets ou non, ont toujours soufferts d'un manque de documentation, non seulement une fois le produit complète mais également lorsque le produit est en cours de développement .c'est pour cette raison que le langage UML (pour unified modeling langage) a été conçu, il permet de décrire un système de manière graphique, un peu de la même manière que les circuits numériques sont décrit par des plans et des diagrammes.

Né de la fusion des méthodes objet dominantes (OMT, Booch et OOSE), puis normalisé par l'OMG en 1997, UML est rapidement devenu un standard incontournable. UML n'est pas à l'origine des concepts objet, mais il en donne une définition plus formelle et apporte la dimension méthodologique qui faisait défaut à l'approche objet.

2- UML et la modélisation

2-1Qu'est-ce qu'un 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. Concrètement, un modèle permet de réduire la complexité d'un phénomène en éliminant les détails qui n'influencent pas son comportement de manière significative. Il reflète ce que le concepteur croit important pour la compréhension et la prédiction du phénomène modélisé. Les limites du phénomène modélisé dépendant des objectifs du modèle.

2-2-Pour quoi modéliser ?

- Modéliser est le futur, et je pense que les sociétés qui travaillent dans ce domaine ont raison » B. Gates
- « Obtenir du code à partir d'un modèle stable est une capacité qui s'inscrit dans la durée » R. Soley
- « A quoi bon modéliser puisque in fine il faudra toujours écrire du code? »
- « Un bon schéma vaut mieux qu'un long discours … sauf qu'à un schéma (UML) correspond plus d'un long discours ! »
- Besoin de bonnes pratiques et d'objectifs précis

3-UML1 et ses vues (les 9 diagrammes)

Dans cette présentation d'UML, nous n'abordons que le métamodèle UML et ses diagrammes.

Le méta-modèle UML fournit une panoplie d'outils permettant de représenter l'ensemble des éléments du monde objet (classes, objets, ...) ainsi que les liens qui les relient.

Toutefois, étant donné qu'une seule représentation est trop subjective, UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux **vues**. Une vue est constituée d'un ou plusieurs **diagrammes**. On distingue deux types de vues:

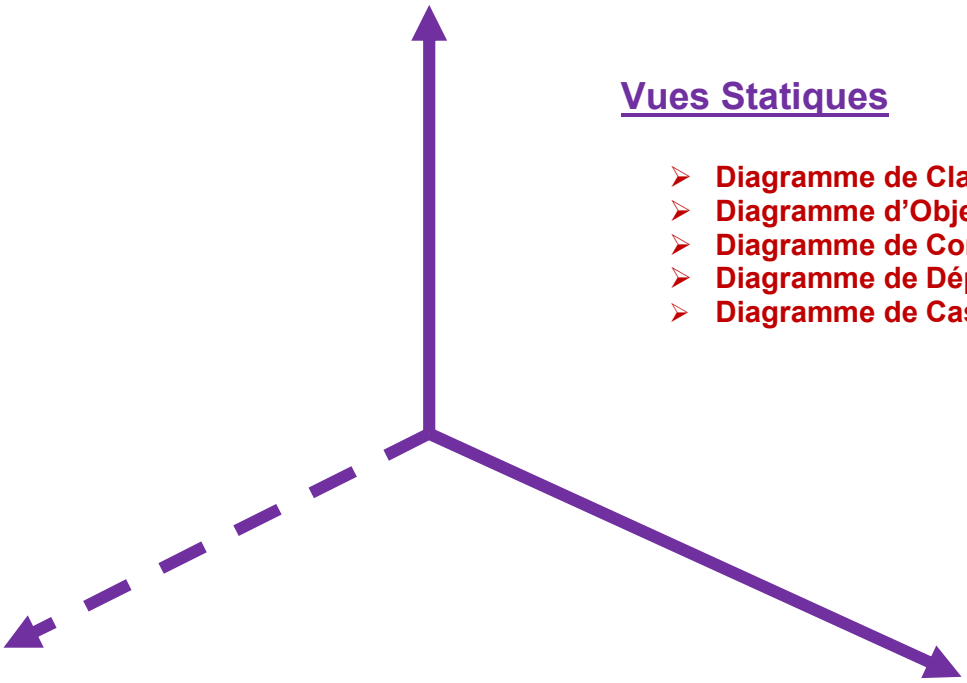
1. Les vues statiques

1. diagrammes de cas d'utilisation
2. diagrammes d'objets
3. diagrammes de classes
4. diagrammes de composants
5. diagrammes de déploiement

2. Les vues dynamiques

6. diagrammes de séquence
7. diagrammes de collaboration
8. diagrammes d'états-transitions
9. diagrammes d'activités

Les diagrammes des cas d'utilisation représentent également une vue fonctionnelle du Système.



Vues Statiques

- Diagramme de Classes
- Diagramme d'Objets
- Diagramme de Composants
- Diagramme de Déploiement
- Diagramme de Cas d 'utilisation

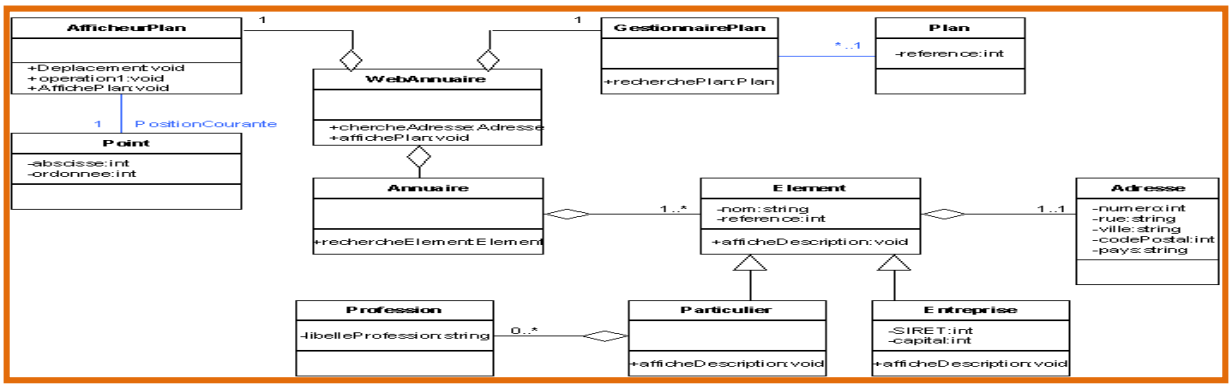
Vue Fonctionnelle

- *Diagramme de Cas d'utilisation*

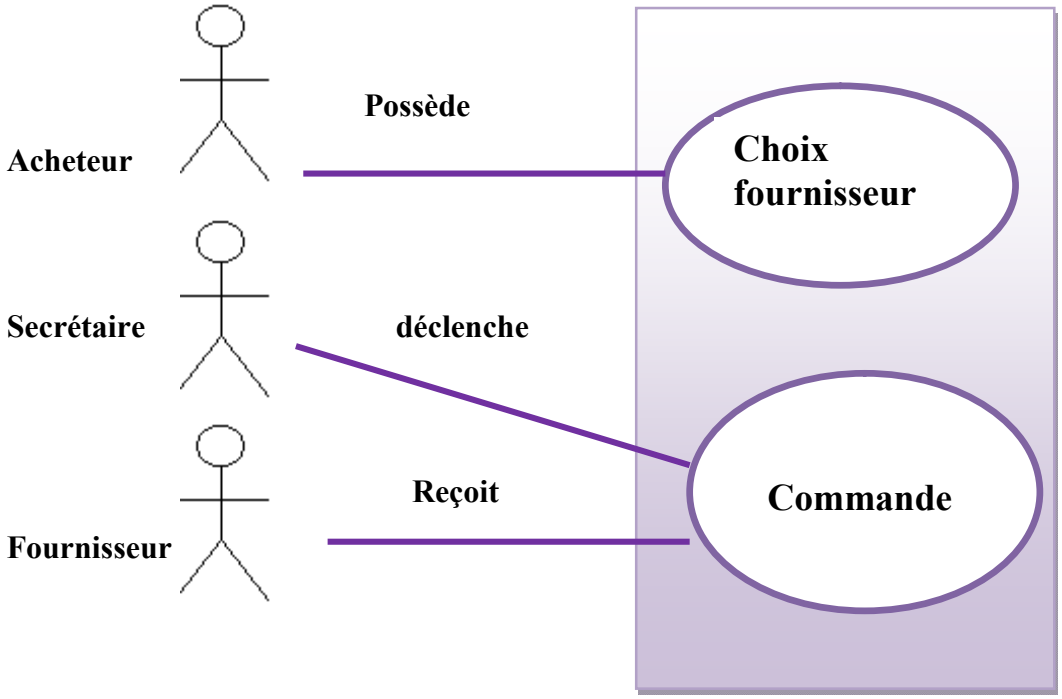
Vues Dynamiques

- Diagramme d'Etats-Transitions
- Diagramme d'Activité
- Diagramme de Séquence
- Diagramme de Collaboration

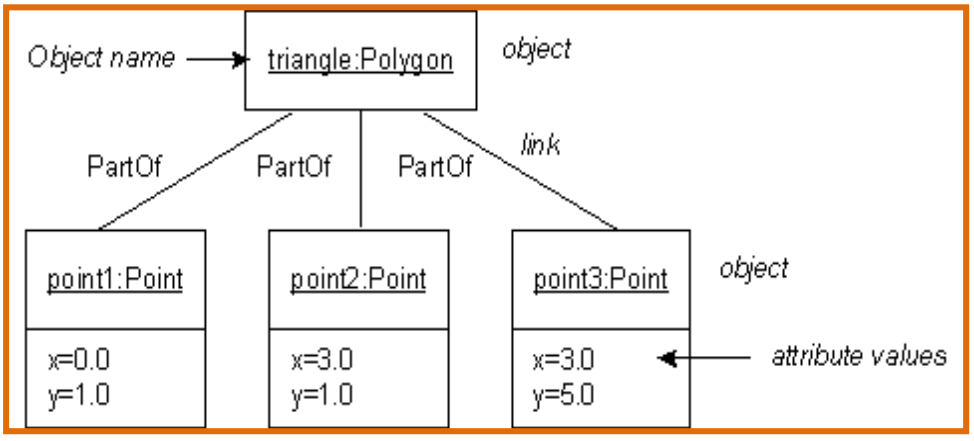
- **Les diagrammes de classes** : expriment de manière générale la structure statique d'un système, en termes de classes et de relations entre ces classes. De plus, ils présentent un ensemble d'interfaces et de paquetages, ainsi que leurs relations.



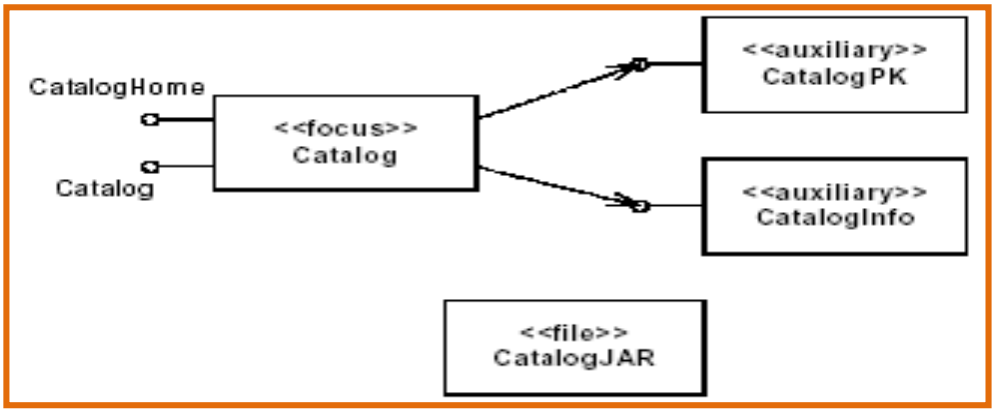
- **les diagrammes de cas d'utilisation** : représentent les cas d'utilisation, les acteurs, et les relations entre le cas d'utilisation et les acteurs. Ils décrivent, sous forme d'actions et de réactions, le comportement d'un système du point de vue d'un utilisateur.



- **les diagrammes d'objets** : sont des graphes d'instances qui incluent les objets et les valeurs de données. Un diagramme d'objets statiques est une instance d'un diagramme de classes, présente un snapshot de l'état détaillé d'un système à un instant donné.



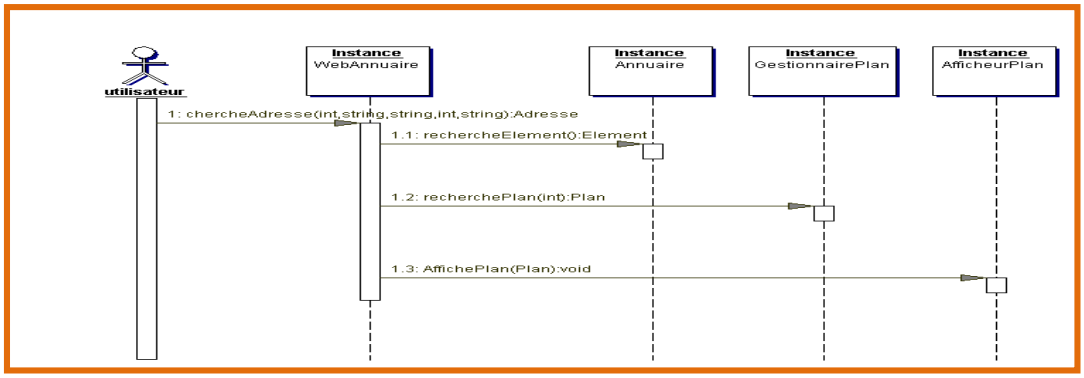
- **les diagrammes de composants** : présentent les dépendances entre les composants logiciels. Ils incluent les classificateurs (2.e. classes) qui spécifient les composants, et les artefacts qui les implémentent, tels que les fichiers de code source, de code binaire, exécutables ou scripts.



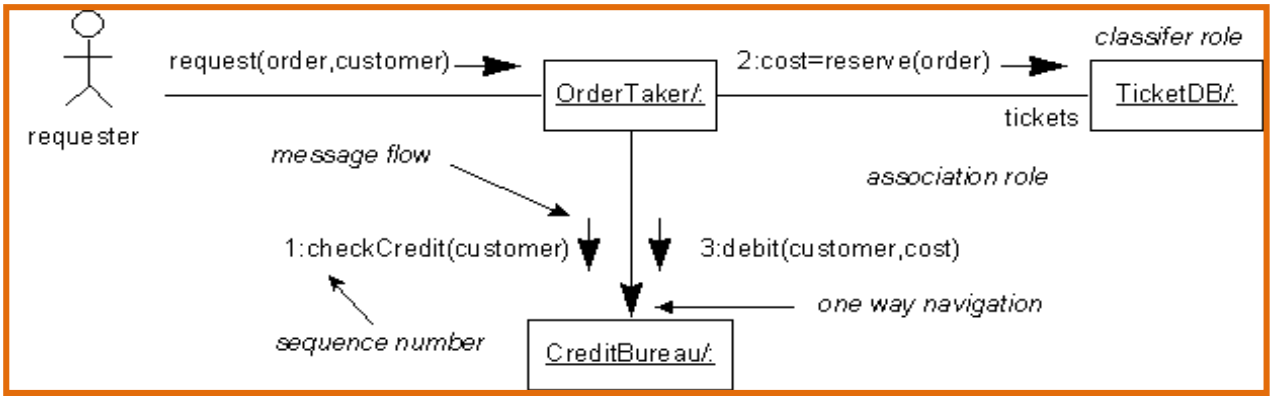
- **les diagrammes de déploiements** : présentent la configuration des éléments de traitement en temps d'exécution, ainsi que les composants logiciels, les processus et les objets qui les exécutent.

Les diagrammes d'UML représentant les caractéristiques dynamiques sont :

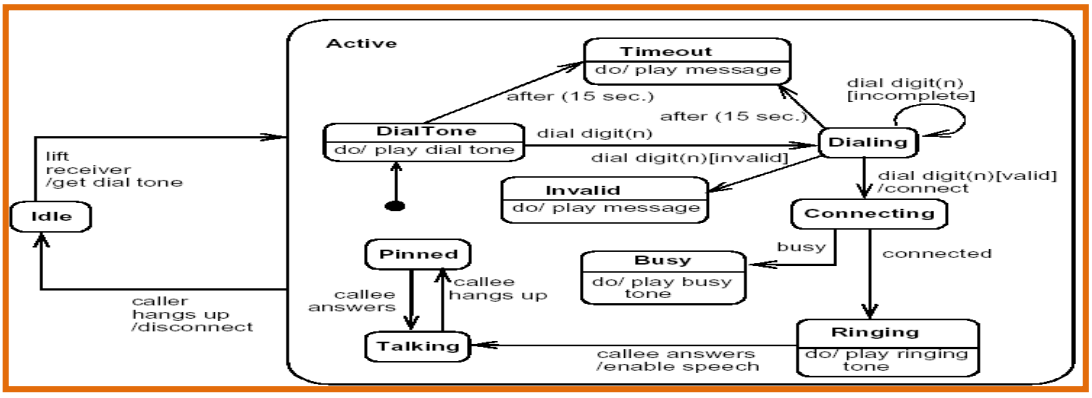
- **les diagrammes de séquence** : montrent les interactions entre les objets dont la représentation se concentre sur la séquence des interactions selon un point de vue temporel.



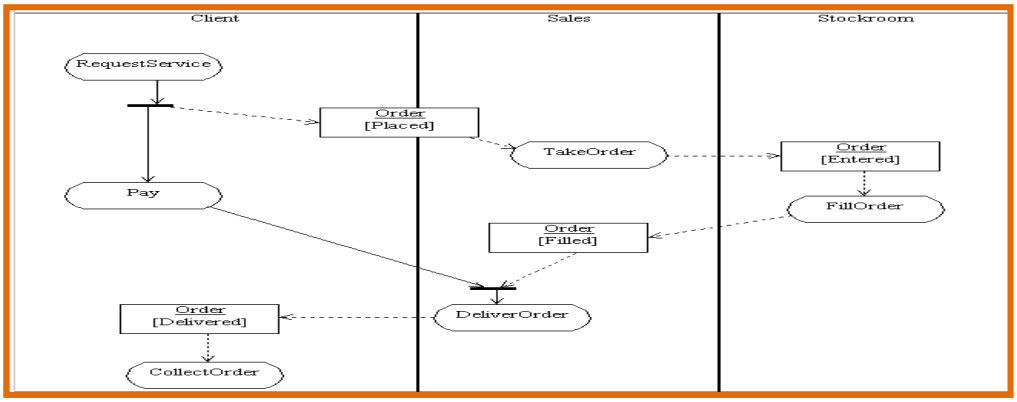
- **les diagrammes de collaboration** : présentent l'ensemble des rôles joués par des objets dans un contexte particulier, ainsi que les liens entre ces objets.



- **les diagrammes d'états-transitions** : représentent les automates d'états finis du point de vue des états et des transitions.



- **les diagrammes d'activités** : sont une variation des diagrammes d'états-transitions dans laquelle les états représentent à la fois la réalisation des actions ou sous activités et à la fois les transitions déclenchées par la finalisation des actions ou sous activités.



4-Insuffisances d'UML1.x et le passage vers UML2

Beaucoup d'utilisateurs d'UML basculeront vers UML2.0 simplement parce que les nouvelles versions des outils UML supporteront UML2.0. Il est évident que ce passage doit être contrôlé, car UML2 introduit de nouveaux concepts de modélisation qu'il faut maîtriser méthodologiquement.

A quelques exceptions près, les diagrammes UML1.4 sont des constructions valides sous UML2. Cette propriété permet ainsi de s'appuyer sur les pratiques actuelles, pour progressivement introduire les apports UML2.

Le passage doit être guidé par le besoin. UML2 n'est pas une révolution. Les méthodologies développées sous UML1.4, utilisant notamment les diagrammes de Use Case, et les diagrammes de classe restent valides. UML2 permet de compléter certains aspects:

· Les diagrammes de flux sont très utiles pour les phases amont,

· Les classes structurées, parts et port sont des outils puissants pour modéliser les architectures,

· Les machines à état "protocole" (Protocol State Machines) peuvent être utilisées extensivement, comme un complément utile à apporter aux diagrammes conceptuels.

Ainsi qu'UML1 est reconnu comme un standard de facto parmi les formalismes Orient´e-Objet au niveau M2. Ses versions précédentes (UML1.x) ne supportaient pas MDA (Model Driven Architecture) non plus que certains de nos besoins pour M2.

5-l'émergence de UML2

Décembre 2006. La commission "UML RTF" de l'OMG1 va terminer ses travaux sur UML2, consacrant ainsi l'aboutissement de ce standard. Adopté en Juin 2004, UML2.0 qui est une révision majeure de UML1.4, est déjà supporté par un nombre important d'ateliers UML, d'autres devant annoncer prochainement leur support. Il reste que UML1.4 demeure très majoritairement utilisé, les nouveaux concepts UML2 n'étant pas encore bien compris, mis en œuvre et méthodologiquement supportés.

La nouvelle version, UML2.0, apporte des points nouveaux et suit l'approche de méta-modélisation supportant MDA.

Le méta-modèle d'UML2.0 vise `à modéliser les structures et les comportements des connaissances dans le champ du développement de logiciels. Il permet `a l'usager de spécifier des modèles au niveau M1. Par exemple, des ´éléments définis au niveau M1

tels que des classes, classes relationnelles et instances sont instanciées des meta-classes concrètes Class, Association, Association Class, Property, InstanceSpecification et Package. UML2.0 remplit assurément nos cinq premiers besoins pour M2 :

Besoin 1 : ***Relation n-aire. Une relation n-aire relie n ´éléments o`u n est un entier positif. Nous avons donc des relations unaires (n=1), binaires (n=2), ternaire (n=3), etc.***

Besoin 2 : ***héritage entre types et héritage entre types relationnels. Parce qu'un type relationnel peut avoir ses propres attributs, comportements et relations avec d'autres ´éléments, l'héritage entre types relationnels est aussi important que celui entre types non-relationnels***.

Besoin 3 : ***Multi-classification et connaissance partielle pour ´éviter l'explosion combinatoire liée `a l'héritage multiple.***

Besoin 4 : ***Catégorie ou Instance Un ´élément peut ˆêtre trait´e comme instance `a un niveau mais comme une catégorie `a un niveau inferieur.***

Besoin 5 : ***Distinction entre l'instanciation globale (traversant deux niveaux de modélisation adjacents) de l'instanciation locale (existant entre types et instances au niveau M1).***

5-1- A qui s'adresse UML2

UML2 adresse fondamentalement la même cible qu'UML1.4. Ses extensions permettent cependant d'élargir la gamme de ses utilisateurs en offrant des outils de modélisation plus adaptés.

Une forte composante liée à l'informatique "technique" (télécoms, temps réel) est intervenue dans les évolutions de ce standard. On peut par exemple citer des sociétés comme MOTOROLA ou Ericson, ou des outilleurs de ce domaine comme Telelogic. L'influence de ces sociétés a renforcé UML dans la précision de certains types de diagrammes (séquence notamment), et les capacités d'UML relativement à la modélisation des architectures logicielles.

Des influences de natures différentes ont été exercées sur d'autres aspects du standard UML.

5-2- UML2 langage universel

Cependant que UML, fort du succès de sa version précédente, consolide ses capacités de modélisation, apparaît un mouvement de contradicteurs prônant la définition de DSL:

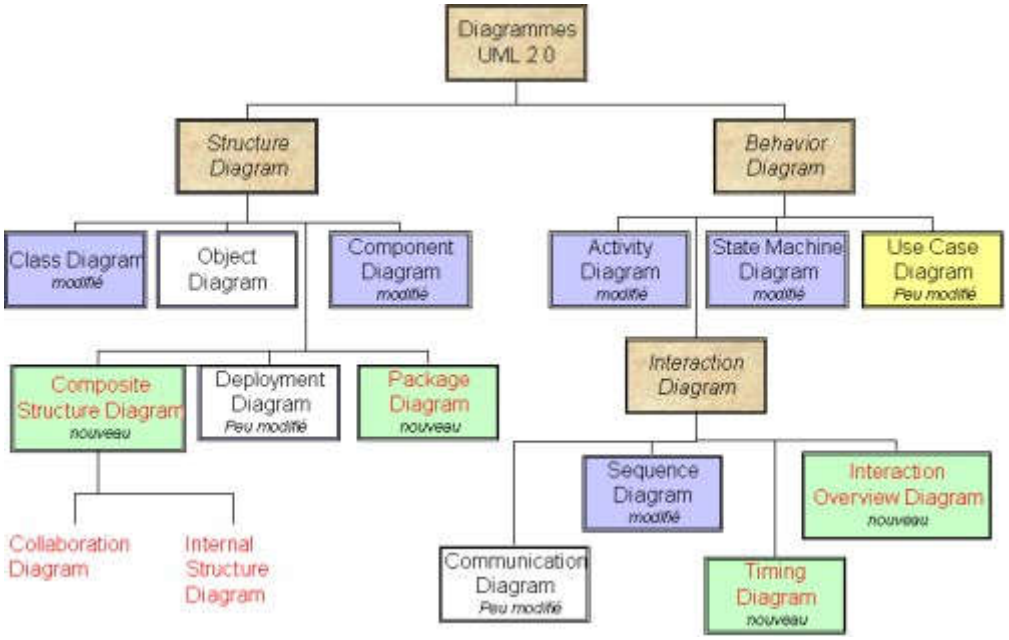
Domain Specific Languages. L'un des participants actifs de ce mouvement n'est autre que Microsoft…

UML "langage universel" cohabite donc avec les "DSL"(Domain Specific Languages). Il y a une zone de compétition lorsqu'en adaptant UML par le mécanisme des profils UML, on rend celui-ci spécifique d'un domaine. Par exemple, des profils UML existent pour modéliser les composants et architectures hardware et la répartition des logiciels sur ces composants.

6-Que nous apporte réellement UML2

6-1- L'impact DUML2.0 sur les diagrammes

La version UML 2.0 permet de structurer les aspects d'un système avec treize diagrammes appartenant à différents niveaux d'abstraction. Par exemple, l'aspect statique d'un système peut être décrit par un diagramme de classes, l'aspect dynamique avec un diagramme de machines d'états et les fonctionnalités générales peuvent être décrites sous forme de diagrammes de cas d'utilisation (à un niveau assez abstrait). En plus, la structuration des aspects d'un système avec des diagrammes UML peut être guidée par la démarche de développement appelée Processus Unifié (Jacobson *et al.*, 1997). Cette démarche organise le travail de développement du système en termes de temps et d'espace et se fait à travers des raffinements itératifs et incrémentaux des différents diagrammes.



Cartographie des diagrammes UML avec l'arrivée du standard UML 2.0

6.2. Les changements des Diagrammes au niveau UML2.0 :

- **d'objets et de packages sont devenus " officiels"**
- **13 diagrammes (au lieu de 9)**
- **Diagramme de structure composite**
- **Diagramme de timing**
- **Diagramme de vue d'ensemble des interactions**
- **Diagramme de collaboration -> diagramme de communication**

les principaux changements a travers diagramme de Classes, de Séquences et diagramme De machines-états.

A. Diagramme de machines-états

Les actions (momentanées) et les activités(continues) sont regroupées sous le même terme : **activité , et** les activités sont alors appelées **activité do.**

B. Diagramme d'activités :

Différenciation forte entre diagramme d'activités et diagramme d'états (À l'inverse de UML 1.x)

Nouvelles notations telles que les signaux temporels, l'acceptation, les paramètres, les spécifications de jonction, les connecteurs, les râteaux de sous-diagrammes, etc **…**

C. Diagramme de classes :

D. Diagramme de séquences :

Notation des cadres d'interaction pour gérer les structures itératives, conditionnelles ou autre (mots-clés alt, opt, par, loop, region, etc …)

6.3. Les nouveaux diagrammes d d'UML2.0

Diagramme de structure composite

Permet de décomposer hiérarchiquement une classe en une structure interne

Diagramme de timing

Représentation des contraintes temporelles entre les changements d'états de différents objets

Diagramme de vue d'ensemble des interactions

Combinaison de diagrammes d'activités et de diagrammes de séquences.

6.4. UML2 pour les phases amont

UML dispose depuis ses débuts des diagrammes de Use Case qui adressent les phases amont, en représentant les modes d'utilisation d'un système sans s'intéresser à son fonctionnement et aux choix d'implémentation. UML2 apporte quelques outils complémentaires pour les phases amont, notamment les flux d'information, utiles pour donner une représentation globale d'un système dès les premières réflexions.

Avec les cas d'utilisation (Use Case), les flux d'information sont des éléments informels, permettant de faire de premiers modèles qui n'induisent pas de décisions sur la manière détaillée de représenter le problème. La notion de modèles informels est très importante pour adresser les besoins des phases amont, comme par exemple les activités de la maîtrise d'ouvrage ou des activités d'analyse préliminaire.

Par ailleurs, les diagrammes de classe, bien connus sous UML1.4 restent utiles pour réaliser des modèles conceptuels, et les diagrammes d'activité offrent tous les moyens de modéliser les processus métier.

6.5. UML2 pour l'architecture logicielle

UML2.0 offre des nouveaux concepts (interfaces offerte et requise, port, structure composite et connecteur) permettant de définir l'architecture de l'application que l'on désire développer. Le méta-modèle UML2.0 considère les deux concepts class et

component comme des classifieurs. De plus, la méta-classe component est une sous-classe de la méta-classe class.

UML2.0 spécifie un composant comme étant une unité modulaire, réutilisable qui interagit avec son environnement par l'intermédiaire de points d'interactions appelés ports. Le comportement d'une interface est décrit par un protocole de type « machine à états » (protocol state machine). Les interfaces peuvent être fournies ou requises. De même, les ports installés sur des composants ou des classes peuvent être fournis ou requis. Le comportement d'un port est issu de la composition des comportements des ces interfaces.

Il existe deux types de modélisation de composants dans UML2.0 : le composant atomique (ou basique) et le composant composite (structure composite). La première catégorie définit le composant comme un élément exécutable du système. La deuxième catégorie étend la première en définissant le composant comme un ensemble cohérent des parties appelées parts. Chaque part représente une instance d'un autre composant. La connexion entre les ports requis et les ports fournis se fait au moyen de connecteurs. Deux types de connecteurs existent : le connecteur de délégation et le connecteur d'assemblage.

Le connecteur de délégation est utilisé pour lier un port du composant composite vers un port d'un composant situé à l.intérieur du composant composite : relier par exemple un port requis à un autre port requis. Le connecteur d'assemblage est utilisé pour les liens de construction : relier un port requis à un port fourni.

Aujourd'hui ; UML est le langage de modélisation orienté objet le plus connu et le plus utilisé au monde.