UML : Langage de modélisation objet unifié

Cours n°1 : Présentation

AVANT PROPOS:

U.M.L. signifie ***Unified Modeling Language*** (C'est-à-dire Langage de Modélisation Unifié).

C'est un **langage graphique** qui permet de créer des **modèles** de systèmes utilisant des **objets**. Avant d'aborder le langage UML, nous définirons d'abord les notions **d'objet** et de **modèle**.

1) L'objet :

Les applications étant de plus en plus importantes et de plus en plus complexes, si nous désirons que le programme soit :

- ✗ le plus facile et le plus rapide à concevoir.
- ✗ évolutif.
- ✗ réutilisable.
- ✗ d'une maintenance aisée.
- ✗ compréhensible et intuitif.

Il est dès lors indispensable d'utiliser un langage de programmation qui facilite l'obtention de ces conditions.

Les styles de développement :

- ✔ ***La programmation procédurale*** ***(Pascal, ADA, C, etc.)*** permet grâce à une analyse fonctionnelle de décomposer le programme en fonctions et sous-fonctions, ce qui permet d'obtenir ***un code structuré*** qui peut remplir les conditions ci-dessus. Mais ce code n'est pas parfaitement modulaire du fait de la séparation des données et des traitements (fonctions).
- ✔ ***La programmation objet*** ***(C++, Java, C#, etc.)*** pallie ce problème en fusionnant dans une même entité (***objet***) les données et les traitements, ***le code devient ainsi très modulaire*** et par conséquent est à même de remplir les conditions ci-dessus.

Structure d'un objet :

- ✔ Un objet est une entité qui regroupe :
  - ✗ des **attributs.** (qui définissent son **état**)
  - ✗ des **méthodes.** (qui définissent son **comportement**)
- ✔ Un objet est une instance de **classe**.
- ✔ Les principales caractéristiques de l'objet sont:
  - ✗ **L'encapsulation.**
  - ✗ **La composition** et **l'agrégation.**
  - ✗ **L'héritage.**
  - ✗ **Le polymorphisme.**

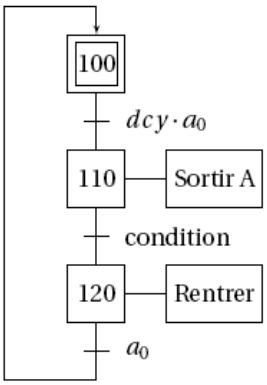
2) La modélisation :

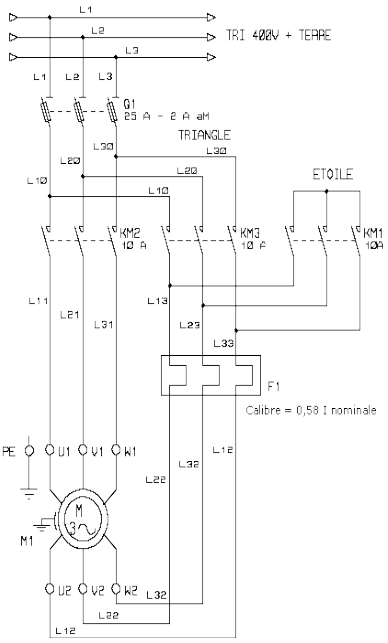
2-1) Qu'est-ce qu'un modèle ?

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

Exemples :

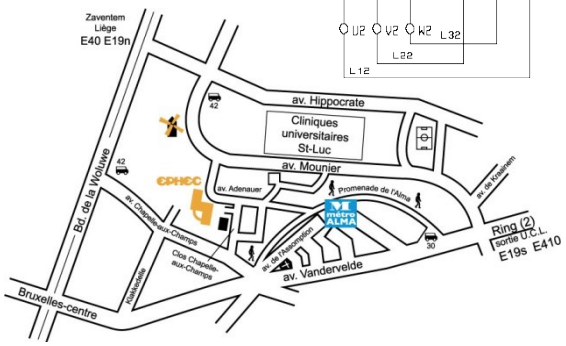
- *Les plans, les cartes les schémas sont des modèles (schémas électriques, cartes topographiques, plans architecturaux, notices de montages, GRAFCET, etc.).*
- *Les calendriers sont une modélisation de la semaine, du mois ou de l'année.*
- *Modèles météorologiques.*
- *Modèles économiques.*
- *Équations physiques et chimiques.*





$$CH_4 + 2\ O_2 \rightarrow CO_2 + 2\ H_2O$$

$$E=m.C^2$$



2-2) Intérêt de la modélisation :

La modélisation permet :

- ✗ De prédire un fonctionnement ou des événements à venir.
- ✗ De mieux comprendre, d'expliquer et de maîtriser le fonctionnement d'un système (un dessin vaut mieux que mille discours) et par conséquent de faciliter sa réalisation puis sa maintenance.
- ✗ Permet dans certains cas d'automatiser certaine tâches (génération automatique du squelette du programme à partir d'un modèle).
- ✗ De disposer d'un langage commun entre les différents intervenants, de la maîtrise d'ouvrage (**MOA**) à la maîtrise d'œuvre (**MOE**).

La MOA : ***est le commanditaire du projet.***
Le chef de projet du coté **MOA** est appelé **« expert métier ».** C'est lui qui connaît exactement les besoins mais il n'a pas les compétences techniques pour la réalisation.
C'est la MOA qui réalise le cahier des charges et dans l'idéal c'est elle qui devrait modéliser.

Le MOE : ***la maîtrise d'œuvre (développeurs)*** s'occupe de la réalisation technique du projet. Soit elle peut réaliser elle-même le projet, soit elle sous-traite une partie ou la totalité du projet.

2-3) Frontière et type d'un modèle :

La plupart du temps, un seul et unique modèle n'est pas suffisant pour décrire et réaliser un projet. Par exemple, pour la construction d'une maison, nous faisons appel à un architecte. Nous sommes la **MOA** et l'architecte est le **MOE**.
Nous définissons avec l'architecte quels sont nos besoins (**cahier des charges**). Et celui-ci va réaliser plusieurs modèles suivant différents points de vue. Du coup :

- ✗ ***Le terrassier dispose d'un plan de masse.***
- ✗ ***Le maçon dispose d'un plan architectural.***
- ✗ ***L'électricien dispose d'un schéma électrique.***

3) UML : Le langage de modélisation objet.

3-1) Historique:

Avant la programmation objet, les méthodes de modélisation avaient une approche structurée, avec une séparation des données et des traitements (comme la méthode **Merise**).
Le succès de la programmation objet a vu l'apparition de plus d'une cinquantaine de méthodes de modélisation objet.
Trois méthodes sortaient du lot (**OMT, OOD, OOSE**) et chose rare, les trois créateurs de ces méthodes se mirent d'accord pour les fusionner en une seule (en prenant à chaque fois le plus intéressant de chaque méthode).

Cette collaboration donnera naissance à **UML** (***Unified Modeling Langage*** → Langage de Modélisation Unifié), le mot unifié étant là pour rappeler cet effort de convergence.

A noter qu'UML n'est pas une méthode mais un **langage** de modélisation. **UML** permet de **créer des modèles** qui serviront de support à la création du programme mais ne donne pas de démarche pour la création de ce programme. Il pourra être complété par une méthode (**RUP, MDA**...).

3-2) Caractéristiques d'UML:

UML est **un langage graphique** qui permet de **modéliser** des systèmes sous la forme d'une collection d'**objets**.

Un seul type de modèle n'étant pas suffisant pour décrire correctement un système, **UML** dispose de plusieurs types de modèles appelés diagrammes (13 en tout), chaque diagramme représentant une vue distincte du système.

Il n'est bien sur pas nécessaire d'utiliser tous les diagrammes pour modéliser un système, il faut choisir parmi tous les diagrammes disponibles ceux qui sont les plus pertinents et les mieux adaptés à la modélisation de notre système.

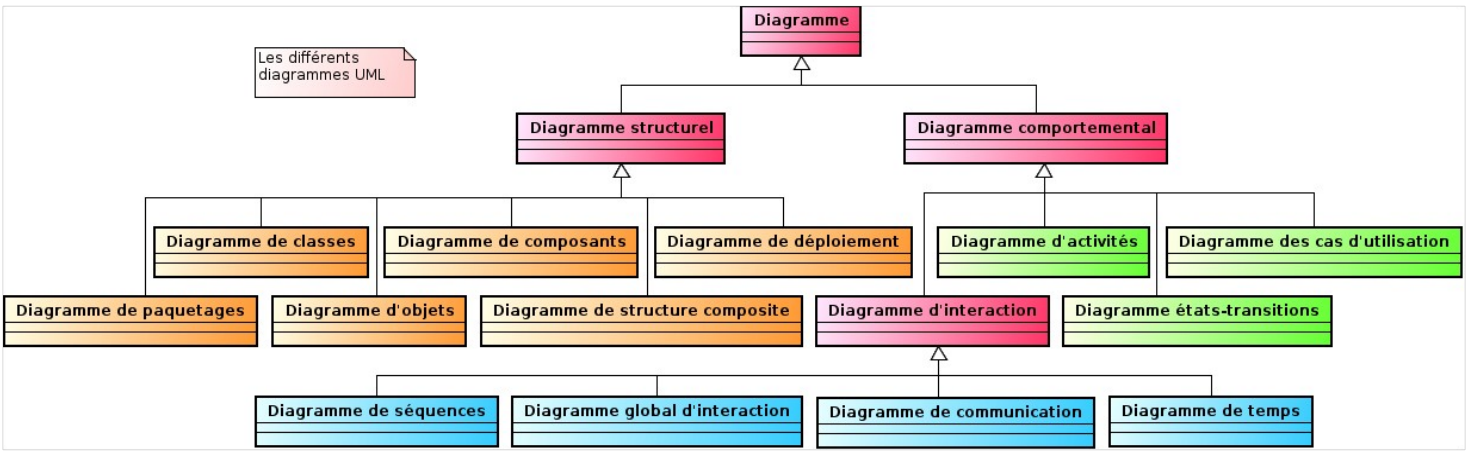
Ces 13 diagrammes sont répartis en 2 groupes :

- **Les diagrammes structurels ou statiques** ***(structure diagrams).***
- **Les diagrammes comportementaux ou dynamiques** ***(behavior diagrams).***

Les diagrammes comportementaux font apparaître un nouveau groupe de diagrammes:

- **Les diagrammes d'interaction** ***(interaction diagrams).***

Certains outils ***(Astah, BOUML, etc.)*** permettent de générer automatiquement le squelette du programme dans n'importe quel langage de programmation objet (C++, java...) à partir des diagrammes **UML**. C'est vers cette orientation que devrait évoluer UML, **la génération automatique de code**. **UML** serait non seulement un langage de modélisation mais aussi un langage de programmation.



- **Les diagrammes structurels ou statiques (Structure Diagrams) :**

Diagramme de classes ***(Class diagram)*** : il représente les classes intervenant dans le système ainsi que les relations entre ces classes. C'est le diagramme le plus important et le plus utilisé.

Diagramme d'objets ***(Object diagram)*** : il sert à représenter les instances de classes (objets) et leurs liens à un instant donné.

Diagramme de composants ***(Component diagram)*** : il permet de montrer les composants du système (éléments logiciels), tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données...).

Diagramme de déploiement ***(Deployment diagram)*** : il sert à représenter les éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage...) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.

Diagramme de paquetages ***(Package Diagram)*** : Il permet de structurer la modélisation de systèmes complexes en décomposant le système en plusieurs parties (paquetages). Chaque paquetage étant constitué des différents diagrammes UML concernant la partie modélisée.

Diagramme de structure composite ***(Composite Structure Diagram)*** : il permet de décrire sous forme de boîte blanche les relations entre composants d'une classe.

- **Diagrammes comportementaux ou dynamique : (Behavior Diagram) :**

Diagramme des cas d'utilisation ***(Use Case Diagram)*** : Il constitue la première étape de l'analyse UML en :

- Modélisant les besoins des utilisateurs.
- Identifiant les grandes fonctionnalités et les limites du système.
- Représentant les interactions entre le systèmes et ses utilisateurs.

Diagramme états-transitions ***(State Machine Diagram)*** : Montre comment l'état du système ou de ses composants est modifié en fonction des différents événements.

Diagramme d'activités ***(Activity Diagram)*** : Variante du diagramme états-transitions. Il permet de décrire sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants.

- **Diagrammes d'interaction (Interaction Diagram) :**

Diagramme de séquences ***(Sequence Diagram)*** : Il permet de représenter les échanges (messages, signaux..) entre les différents objets et acteurs du système en fonction du temps.

Diagramme de communication ***(Communication Diagram)*** : C'est une représentation simplifiée d'un diagramme de séquence, il se concentre sur les échanges de messages entre les objets.

Diagramme global d'interaction (I***nteraction Overview Diagram)*** : Il permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité).

Diagramme de temps ***(Timing Diagram)*** : il permet de décrire les variations d'une donnée au cours du temps.