2022

Analyse et conception Objet, introduction à l'UML

GROUPE 8

Membres du groupe

HOUNMENOU K. MOSES

KOUWONOU K FREDERIC

TABOUKOUNA DIBE REBECCA

ANALYSE ET CONCEPTION OBJET
INTRODUCTION A UML

PLAN

I - Pourquoi modéliser ? Appréhender le spectre de l'analyse et de la conception

II - Domaine métier et modélisation d'une solution informatique. Le modèle, un artefact central du processus projet

III - Analyse et conception d'une solution informatique. Impacts des langages de programmation

IV - Evolution vers l'analyse/conception Objet. Avantages

V - Présentation générale d'UML. Evolution et objectifs. Vues de l'architecte

VI - Le cœur de l'UML : les différents types de diagrammes (statiques et dynamiques)

VII - Présentation de plusieurs démarches de modélisation

VIII - Extensions UML : stéréotype, profils, contraintes

IX - Travaux pratiques : Présentation des études de cas. Analyse des domaines métier

I - Pourquoi modéliser ? Appréhender le spectre de l'analyse et de la conception

❖ Qu'est-ce qu'un modèle ?

Un modèle est une représentation abstraite et simplifiée, 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.

❖ Pourquoi modéliser ?

Modéliser un système avant sa réalisation permet de mieux comprendre le fonctionnement du système. 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.

Cette communication est essentielle pour aboutir à une compréhension commune aux différentes parties prenantes (notamment entre la maîtrise d'ouvrage et la maîtrise d'œuvre informatique) et précise d'un problème donné.

Le choix du modèle a donc une influence capitale sur les solutions obtenues. Les systèmes non triviaux sont mieux modélisés par un ensemble de modèles indépendants. Selon les modèles employés, la démarche de modélisation n'est pas la même.

❖ Appréhender le spectre de l'analyse et de la conception

Une méthode d'analyse et de conception a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client. Pour ce faire, on part d'un énoncé informel (le besoin du client), ainsi que de l'analyse de l'existant éventuel (c'est-à-dire la manière dont les processus est traiter par le système se déroulent actuellement chez le client).

La phase d'analyse permet de lister les résultats attendus, en terme de fonctionnalités, de performance, de robustesse, de maintenance, de sécurité, d'extensibilité, etc.

La phase de conception permet de décrire de manière non ambiguë, le plus souvent en utilisant un langage de modélisation, le fonctionnement futur du système, afin d'en faciliter la réalisation.

II - Domaine métier et modélisation d'une solution informatique. Le modèle, un artefact central du processus projet

La modélisation est utilisée dans plusieurs domaines :

En mathématiques appliquées le modèle mathématique permet d'analyser des phénomènes réels et de prévoir des résultats à partir de l'application d'une ou plusieurs théories à un niveau d'approximation donné.

En ingénierie, la modélisation 3D est un cas particulier du précédent qui consiste à produire des images d'objets réels ou imaginés ;

La modélisation peut etre retrouver en chimie, en physique, en météorologie ou en sciences de la vie et de la terre, en pédagogie, en informatique, en conseil, dans une entreprise, en économie ,en musique, en comportement humain.

En informatique, la modélisation permet de concevoir l'architecture globale d'un système d'information.

La **modélisation** est la conception et l'utilisation d'un modèle. Modèle est un Objet conçu et construit (artefact). D'une manière générale et en informatique, un **artéfact** désigne toute sorte d'information créée, produite, modifiée ou utilisée par un homme dans la mise au point d'un « système » informatique. Tout ce **qu**'un utilisateur réalise à l'aide d'un ordinateur peut être considéré comme un **artéfact**.

III - Analyse et conception d'une solution informatique. Impacts des langages de programmation

❖ Analyse et conception d'une solution informatique

Les méthodes d'analyse et de conception proposent une démarche qui distingue les étapes du développement dans le cycle de vie d'une solution informatique ou d'un logiciel.

- ✓ La phase de pré-analyse
- ✓ La phase de spécification
- ✓ La phase de conception
- ✓ La phase de test
- ✓ La phase de maintenance

Elles s'appuient sur un formalisme de représentation qui facilite la communication, l'organisation et la vérification : Le langage de modélisation. Elles produisent des documents (modèles) qui facilitent les retours sur conception et l'évolution des applications
Les langages de programmation interviennent dans la phase de conception du cycle de vie du logiciel.

❖ Impacts des langages de programmation

L' imapct des langages de programmation est de spécifier les besoins et les exigences des acteurs, le système et l'architecture globale.

De nos jours, les outils de modélisation de processus métier s'étoffent chaque année et les suites logiciels sont de plus en plus nombreuses. L'usage et les fonctionnalités d'UML diffèrent d'un périmètre à un autre, selon les besoins des clients et des fournisseurs d'applications.

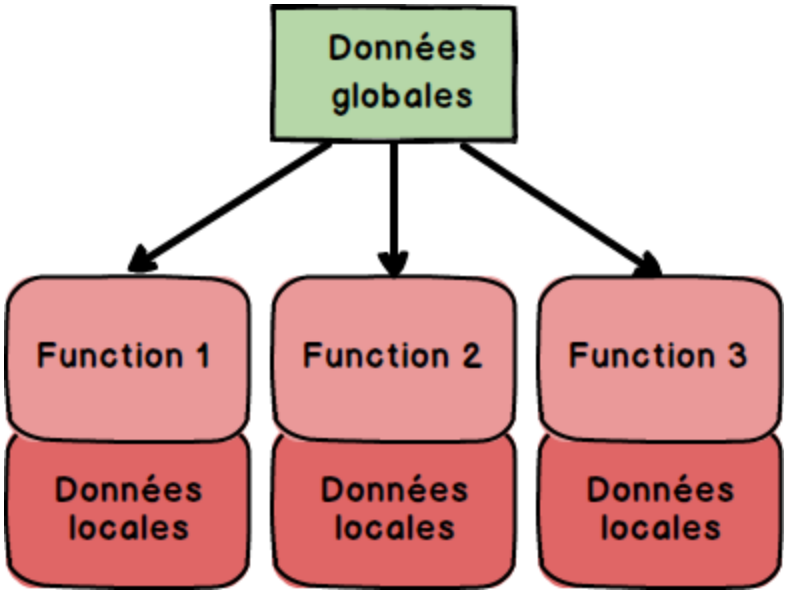
Il permet aussi dans un premier temps de bien définir les besoins clients, et ainsi d'éviter des sur-coûts liés à la livraison d'un logicielle qui ne satisfait pas

le client.

IV- Evolution vers l'analyse/conception Objet. Avantages,

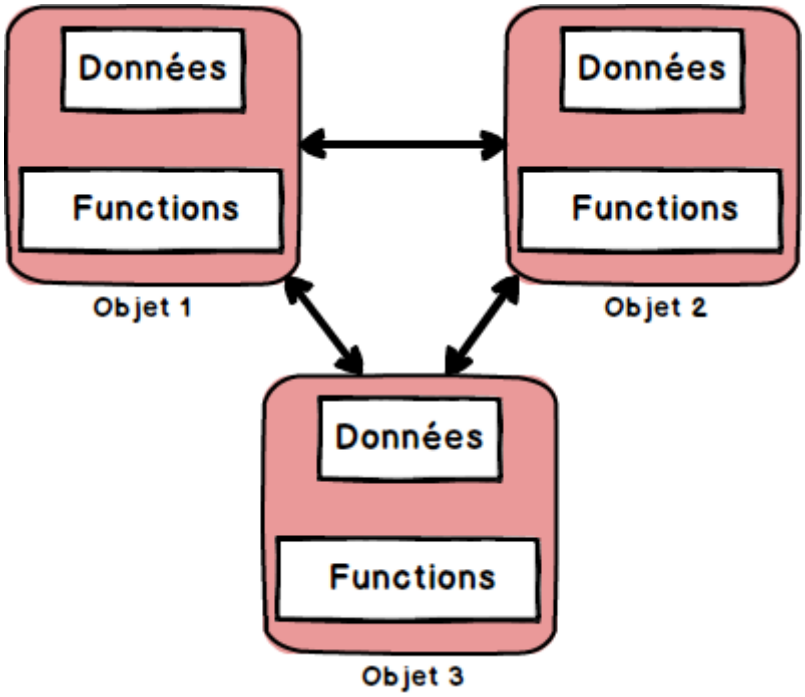
La programmation procédurale, qui est basée sur l'utilisation de procédures, et de la programmation fonctionnelle, qui elle, repose entièrement sur le concept de *fonction*.

C'est quoi la programmation procédurale?



Dans la programmation procédurale le programme est divisé en petites parties appelées procédures ou fonctions. Comme son nom l'indique, la programmation procédurale contient une procédure étape par étape à exécuter. Ici, les problèmes sont décomposés en petites parties et ensuite, pour résoudre chaque partie, une ou plusieurs fonctions sont utilisées.

C'est quoi la Programmation Orientée Objet (POO)?
Les premiers concepts de la programmation orientée objet remontent aux années 1970 avec les les langages *Simula* et *Smalltalk*.



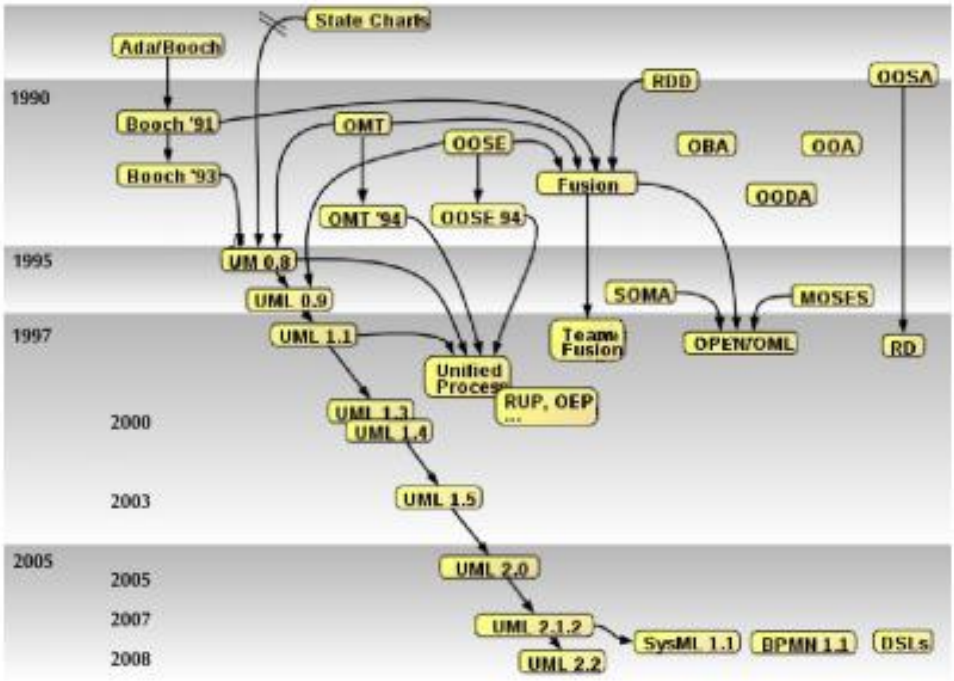
Dans la programmation orientée objet le programme est divisé en parties appelées objets.
La programmation orientée objet est un concept de programmation qui se concentre sur l'objet plutôt que sur les actions et les données plutôt que sur la logique.

	Programmation Procédurale	**Programmation Orientée Objet**
Programmes	Le programme principal est divisé en petites parties selon les fonctions.	Le programme principal est divisé en petit objet en fonction du problème.
Les données	Chaque fonction contient des données différentes.	Les données et les fonctions de chaque objet individuel agissent comme une seule unité.
Permission	Pour ajouter de nouvelles données au programme, l'utilisateur doit s'assurer que la fonction le permet.	Le passage de message garantit l'autorisation d'accéder au membre d'un objet à partir d'un autre objet.
Exemples	Pascal, Fortran	PHP5, C ++, Java.
Accès	Aucun spécificateur d'accès n'est utilisé.	Les spécificateurs d'accès public, private, et protected sont utilisés.
La communication	Les fonctions communiquent avec d'autres fonctions en gardant les règles habituelles.	Un objet communique entre eux via des messages.
Contrôle des données	La plupart des fonctions utilisent des données globales.	Chaque objet contrôle ses propres données.
Importance	Les fonctions ou les algorithmes ont plus d'importance que les données dans le programme.	Les données prennent plus d'importance que les fonctions du programme.
Masquage des données	Il n'y a pas de moyen idéal pour masquer les données.	Le masquage des données est possible, ce qui empêche l'accès illégal de la fonction depuis l'extérieur.

Les failles de la programmation procédurale posent le besoin de la programmation orientée objet. La programmation orientée objet corrige les défauts du programmation procédurale en introduisant le concept «objet» et «classe». Il améliore la sécurité des données, ainsi que l'initialisation et le nettoyage automatiques des objets. La programmation orientée objet permet de créer plusieurs instances de l'objet sans aucune interférence.

V - Présentation générale d'UML. Evolution et objectifs. Vues de l'architecte

❖ Historique d'UML

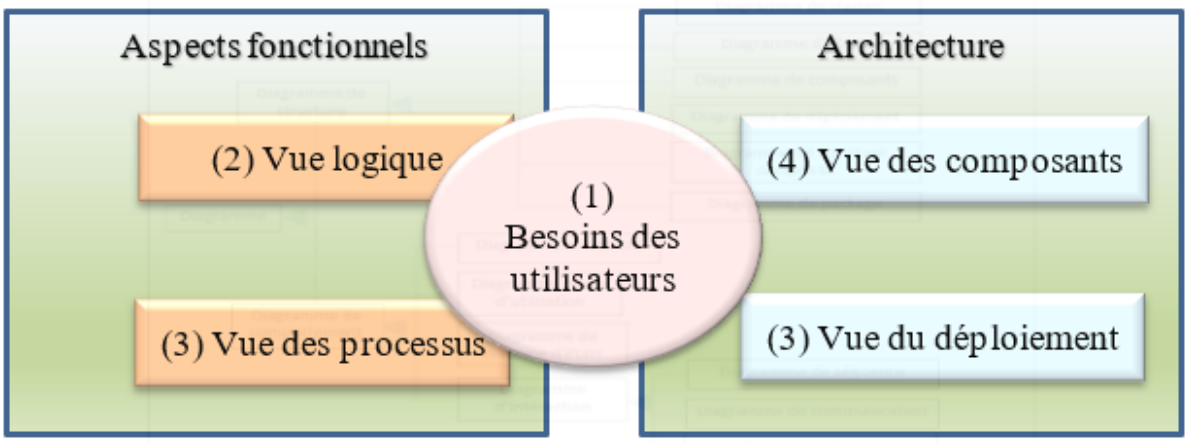


Comme tout langage, UML a connu des évolutions depuis son adoption en novembre 1997 dans sa version 1.1 jusqu'à sa dernière version 2.5.1 diffusé en décembre 2017. A ce jour, cette dernière version de UML, encore appelé UML 2

❖ Présentation de la méthode UML

- UML est la forme contractée de Unified Modeling Language qui peut se traduire en français par langage unifié pour la modélisation
- C'est un langage de modélisation objet l UML se base sur une notation graphique expressive.
- Il permet de modéliser de manière claire et précise la structure et le comportement d'un système indépendamment de toute méthode ou de tout langage de programmation
- La notation UML repose sur plusieurs diagrammes adaptés au développement logiciel pour les phases de spécification, conception et implémentation du cycle de vie.

- UML propose pour l'analyse d'un projet informatique, treize (13) diagrammes « *officiels* » qui peuvent être regroupés suivant deux (02) aspects : l'aspect fonctionnel du logiciel et l'aspect lié à l'architecture du logiciel. Ces deux (02) aspects son axés sur les besoins des utilisateurs, formant un schéma communément appelé **4+1 vues** dont voici une représentation :



1. La vue logique : cette vue exprime la perspective abstraite de la solution en termes de classes, d'objets, de relations, de machine à états de transition etc.. de manière indépendante des langages de programmation et des environnements de développement utilisés pour les mettre en œuvre. Il s'agit d'exprimer le problème de façon abstraite. Les concepts utilisés incluent les concepts de la majorité des méthodes orientés objets existantes. Cette vue concerne l'intégrité de conception

2. La vue des composants : cette vue exprime la perspective physique de l'organisation du code en terme de modules, des composants et surtout des concepts du langage et de l'environnement d'implémentation. Cette perspective dépend du choix du langage utilisé. Dans cette perspective, le concepteur est surtout intéressé par des aspects de gestion de codes, d'ordre de compilation, de réutilisation. UML offre des concepts adaptés comme les modules, l'interface, les composants, les relations de dépendance ect… Malheureusement la plus part des concepteurs ignorent cette vue. Cette vue concerne l'intégrité de gestion

3. La vue des processus : cette vue exprime la perspective sur les activités concurrentes et parallèles (tâches et processus) du système. On y trouve les

activités parallèles, et leur communication et synchronisation. Cette vue concerne l'intégrité d'exécution.

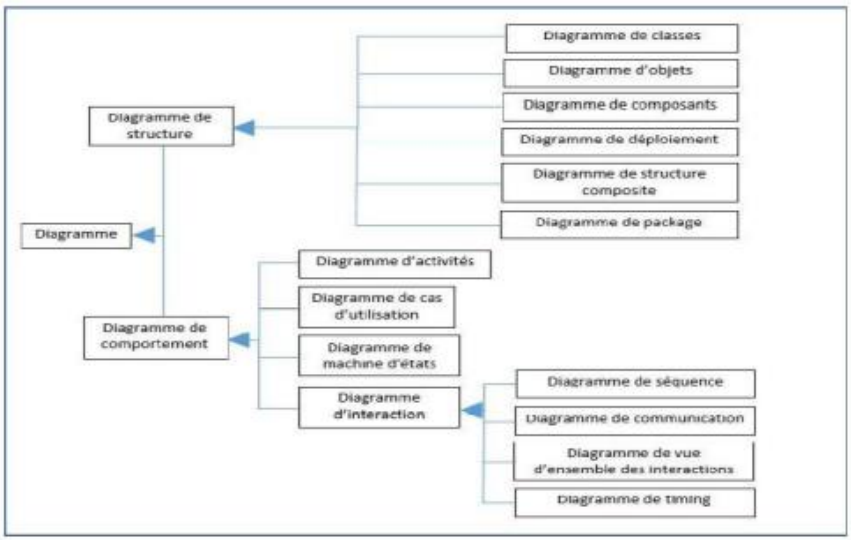
4. La vue de déploiement : elle exprime la répartition du système à travers un réseau de calculateurs et de nœuds logiques de traitement. Cette vue est utile pour décrire la répartition du système réparti, elle concerne l'intégrité de performance 5. La vue des cas d'utilisation : cette vue guide et justifie les autres en ce sens que la modélisation fondée sur les scénarios (cas d'utilisation) constitue ce que l'on fait de mieux aujourd'hui. Cette approche constitue l'unique moyen de guider la modélisation, de trouver le bon modèle

❖ Présentation générale d'UML : objectifs

- UML favorise donc le prototypage, et c'est là une de ses forces. En effet, modéliser une application n'est pas une activité linéaire. Il s'agit d'une tâche très complexe, qui nécessite une approche itérative, car il est plus efficace de construire et valider par étapes, ce qui est difficile à cerner et maîtriser.
- UML permet donc non seulement de représenter et de manipuler les concepts objet, il sous-entend une démarche d'analyse qui permet de concevoir une solution objet de manière itérative, grâce aux diagrammes, qui supportent l'abstraction
- Fournir aux concepteurs de systèmes, ingénieurs logiciels et développeurs de logiciels des outils pour l'analyse, la conception et la mise en œuvre de systèmes logiciels, ainsi que pour la modélisation de processus métier et d'autres processus similaires.
- Faire progresser l'industrie en permettant l'interopérabilité des outils de modélisation visuelle orientés objet. Toutefois, pour permettre un échange significatif d'informations de modèles entre outils, il est nécessaire de trouver un accord sur la sémantique et la notation.
- Fournir une explication détaillée de la sémantique de chaque concept de modélisation
- Spécifier des éléments de notation lisibles par l'homme pour représenter chaque concept de modélisation, ainsi que les règles pour les combiner au sein d'une grande variété de diagrammes correspondant à différents aspects des systèmes modélisés.

VI - Le cœur de l'UML : les différents types de diagrammes (statiques et dynamiques)

UML dispose de différents diagrammes pour modéliser un système. Les diagrammes sont dépendants et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Il en existe treize (13) (depuis UML 2.3 contre 9 pour UML 1.3) que sont :



❖ Les diagrammes de structure ou diagrammes statiques (06)

- **Le diagramme de packages** : il permet de dé composer le logiciel en modules (dits « *packages* ») plus faciles à décrire ; il est possible d'y indiquer également les acteurs interviennent sur chaque package.
- **Le diagramme de classes** : il permet de modéliser les données et les traitements de l'application. Dans la phase d'analyse, il représente les données manipulées par les utilisateurs et dans la phase de conception, il représente la structure « objet » dans un développement orienté objet ;

- **Le diagramme d'objets** : un objet est une instance d'une classe ; ce diagramme sert, ainsi, à modéliser des classes complexes en se servant des instances ou exemple concret (objets) de ces classes.

- **Le diagramme de composants** : il décrit les unités (fichiers, librairies, base de données, etc.) dont l'assemblage permettra l'exécution du système ;

- **Le diagramme de structure composite** : il décrit la structure interne d'un objet pendant son exécution avec les ports et connecteurs utilisés pour communiquer avec d'autres objets ou l'extérieur.
- **Le diagramme de déploiement** : il décrit l'environnement d'exécution du système (matériel, réseau…) et de la façon dont les composants y sont installés.

❖ Les diagrammes comportementaux ou dynamiques (07) ;

- **Le diagramme de cas d'utilisation** : il décrit les fonctionnalités (dit « *cas d'utilisation* ») nécessaires aux utilisateurs ; il peut être présenté pour tout le système ou pour décrire un module donné ;
- **Le diagramme d'activités** : il représente l'enchaînement des actions sans faire intervenir les objets ; il constitue un bon complément pour le diagramme de cas d'utilisation dans le cadre de l'analyse des besoins ;
- **Le diagramme d'états – transitions** : il permet de décrire le cycle de vie d'une instance (objet) d'une classe indépendamment des autres ;
- Les diagrammes d'interactions ou dynamiques (04).
    - **Le diagramme de séquence** : il permet de décrire les aspects dynamiques et les différents scénarii d'utilisation du système en montrant les différentes interactions entre les objets dans un espace temporel ;

    - **Le diagramme de collaboration** : encore appelé diagramme de communication, il permet de mettre en exergues les échanges de

messages entres les différents objets du système ; il complète très bien le diagramme de séquence et de classes ;

- **Le diagramme global d'interaction :** il permet de présenter une vue d'ensemble (globale) des différentes interactions et actions intervenant dans le système et décrites en détail dans le diagramme de séquence et le diagramme d'activités ;

- **Le diagramme de temps :** il modélise les interactions entre différents objets ayant de fortes contraintes de temps ; il est propice à la description des systèmes ayant des contraintes de temps réel.

VII- Présentation de plusieurs démarches de modélisation

UML est un langage qui permet de représenter des modèles, mais il ne définit pas le processus d'élaboration des modèles. Cependant, dans le cadre de la modélisation d'une application informatique, les auteurs d'UML préconisent d'utiliser une des démarches suivantes :

❖ Itérative et incrémentale :

L'idée est simple : pour modéliser (comprendre et représenter) un système complexe, il vaut mieux s'y prendre en plusieurs fois, en affinant son analyse par étapes. Cette démarche devrait aussi s'appliquer au cycle de développement dans son ensemble, en favorisant le prototypage. Le but est de mieux maîtriser la part d'inconnu et d'incertitudes qui caractérisent les systèmes complexes. L'ensemble du problème est décomposé en petites itérations, définies à partir des cas d'utilisation et de l'étude des risques. Les risques majeurs et les cas d'utilisation les plus importants sont traités en priorité. Le développement procède par des itérations qui conduisent à des livraisons incrémentales du système. Nous avons déjà présenté le modèle de cycle de vie par incrément dans la section

❖ Guidée par les besoins des utilisateurs du système :

Avec UML, ce sont les utilisateurs qui guident la définition des modèles :

Le périmètre du système à modéliser est défini par les besoins des utilisateurs (les utilisateurs définissent ce que doit être le système). Le bu

t du système à modéliser est de répondre aux besoins de ses utilisateurs (les utilisateurs sont les clients du système). Les besoins des utilisateurs servent

aussi de fil rouge, tout au long du cycle de développement (itératif et incrémental) :

- A chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs.
- A chaque itération de la phase de conception et de réalisation, on veille à la prise en compte des besoins des utilisateurs.
- A chaque itération de la phase de test, on vérifie que les besoins des utilisateurs sont satisfaits.

la principale qualité d'un logiciel étant son utilité, c'est-à-dire son adéquation avec les besoins des utilisateurs, toutes les étapes, de la spécification des besoins à la maintenance, doivent être guidées par les cas d'utilisation qui modélisent justement les besoins des utilisateurs ;

❖ **centrée sur l'architecture logicielle**.

Elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...). L'architecture est conçue pour satisfaire les besoins exprimés dans les cas d'utilisation, mais aussi pour prendre en compte les évolutions futures et les contraintes de réalisation. La mise en place d'une architecture adaptée conditionne le succès d'un développement. Il est important de la stabiliser le plus tôt possible.

VIII- Extensions UML : stéréotype, profils, contraintes

UML définit trois mécanismes d'extensibilité pour permettre aux modeleurs d'ajouter quelques extensions sans être contraint de modifier le langage de modélisation sous-jacent. Ces trois mécanismes
sont **stéréotypes**, **contraintes** et **profils** Chaque projet est doté d'un noeud Extensions UML. Les Stéréotypes UML et les Contraintes UML sont rassemblés dans ce noeud.

UML permet l'ajout d'une paire de chaînes (une chaîne d'étiquette et une chaîne de valeur) sur chaque élément du modèle. Pour ceux qui considèrent cette fonction trop limitée, les **propriétés d'utilisateurs** permettent les mêmes types de fonctions, mais de surcroît leur valeur peut être de n'importe quel type (chaîne, nombre entier, booléen, et ainsi de suite). Ainsi,
les **propriétés d'utilisateurs** agissent de la même façon que les UML "tagged values" mais de façon plus puissante. Pour plus de détails: voir Propriétés d'utilisateurs .

❖ Les stereotypes

Comme les éléments du modèle n'appartiennent qu'à une seule catégorie, ils ne peuvent pas avoir plus d'un stéréotype. Les stéréotypes sont utilisés pour ajouter de l'information sémantique aux diagrammes de classes ou de données. Ils peuvent être utilisés pour dire aux gabarits ou à la génération de modules de traiter certains objets stéréotypés différemment (par exemple, en générant du code différent).

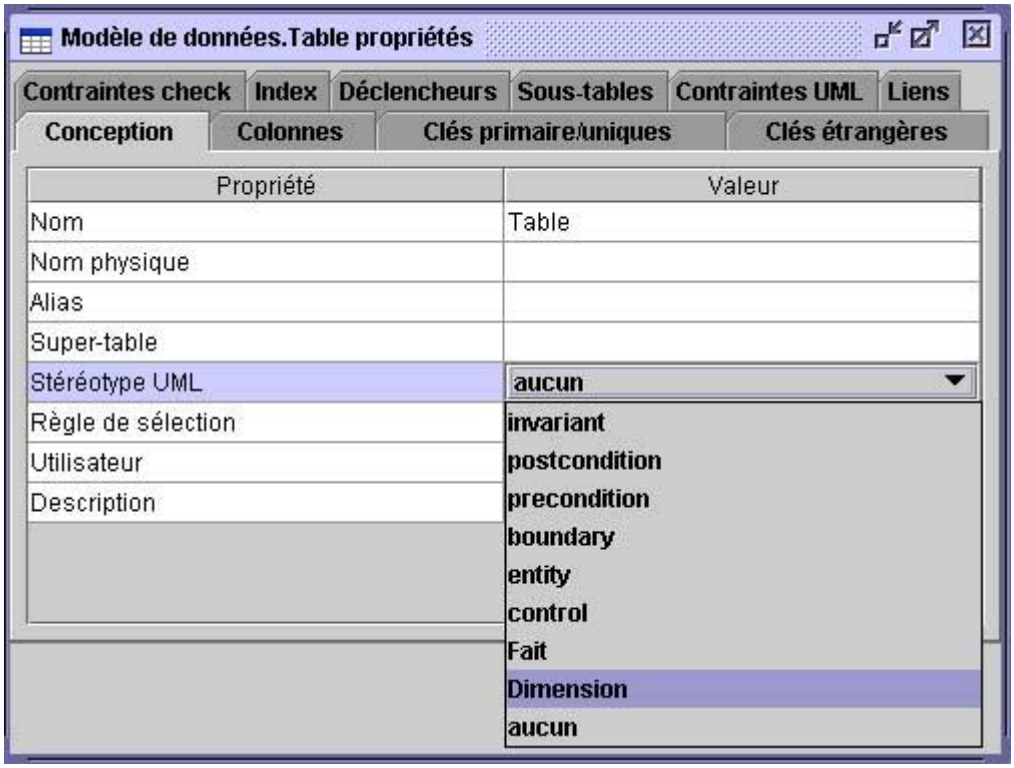
Ajouter de nouveaux stéréotypes

Pour ajouter un stéréotype UML :

- Sélectionnez Extensions UML et de la barre d'outils Édition, cliquez sur **Ajouter > Stéréotype UML**.
- Changez le nom implicite et appuyez sur la touche **Retour**.

Appliquer un stéréotype à un élément du modèle

Dans la fenêtre de propriétés d'un objet (par exemple, celle d'une table ou d'une classe), choisissez un stéréotype UML du menu contextuel des stéréotypes UML.



Afficher les stéréotypes

Même si vous pouvez stéréotyper n'importe quel objet du modèle, seul les objets suivants permettent l'affichage des stéréotypes : les classes, les

paquetages, les tables, les champs, les méthodes, les associations et bouts d'association.
Le nom du stéréotype est affiché à l'intérieur de guillemets. Si un icone est relié à un objet, cet icone sera également affiché pour les classes, les tables et les paquetages.

L'héritage de stéréotype (avancé)

Un stéréotype peut hériter d'un autre stéréotype. Pour qu'un stéréotype puisse hériter d'un super-stéréotype :

1. Affichez la fenêtre de propriétés du stéréotype.
2. Choisissez l'onglet Super-stéréotypes.
3. Cliquez Ajouter.

❖ Les Contraintes UML

Une contrainte est une condition ou restriction sémantique appliquée à un modèle.

Un élément du modèle peut avoir plusieurs contraintes. Une contrainte est déclarée en texte libre; tel que « valeur positive » ou « valeur > 0 ».

Ajouter de nouvelles contraintes

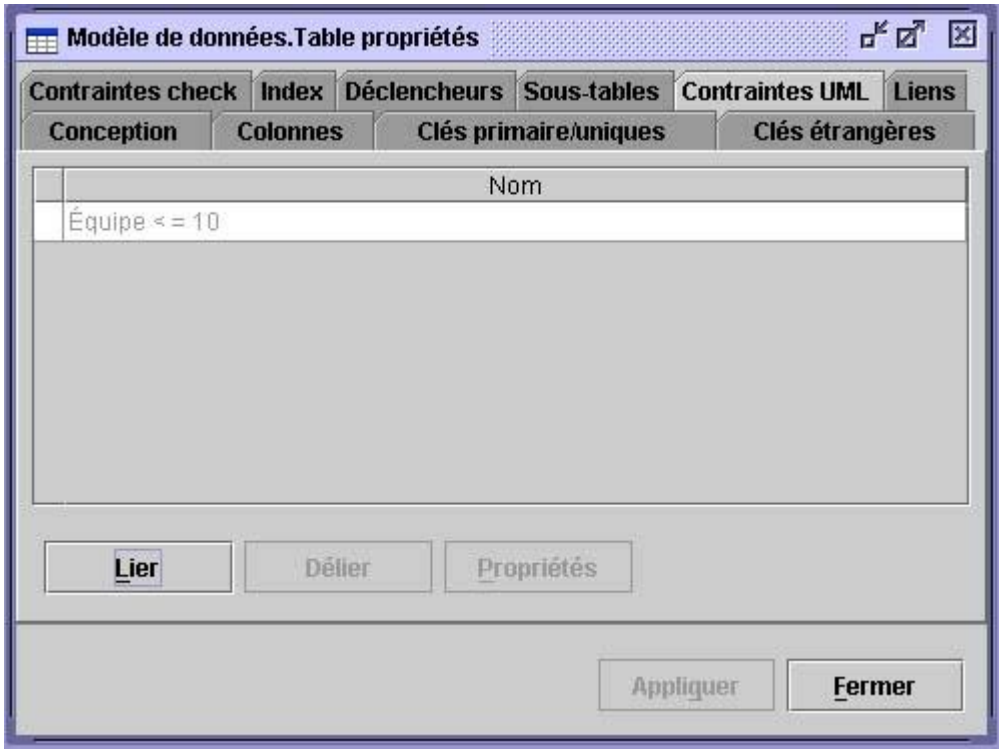
Pour ajouter une contrainte UML :

- Sélectionnez **Extensions UML** et de la barre d'outils Édition faites **Ajouter > Contrainte UML**.
- Changer le nom implicite et appuyez sur **Retour**.

Appliquer une contrainte à un élément du modèle

Pour appliquer une contrainte à un élément du modèle :

1. Dans la fenêtre de propriétés d'un objet (par exemple, celle d'une table ou d'une classe), cliquez sur **contraintes UML**.
2. Cliquez sur **Lier**.
3. De la fenêtre **Contraintes UML**, choisissez une contrainte et cliquez **Sélectionner**.
4. Cliquez sur **Fermer**.



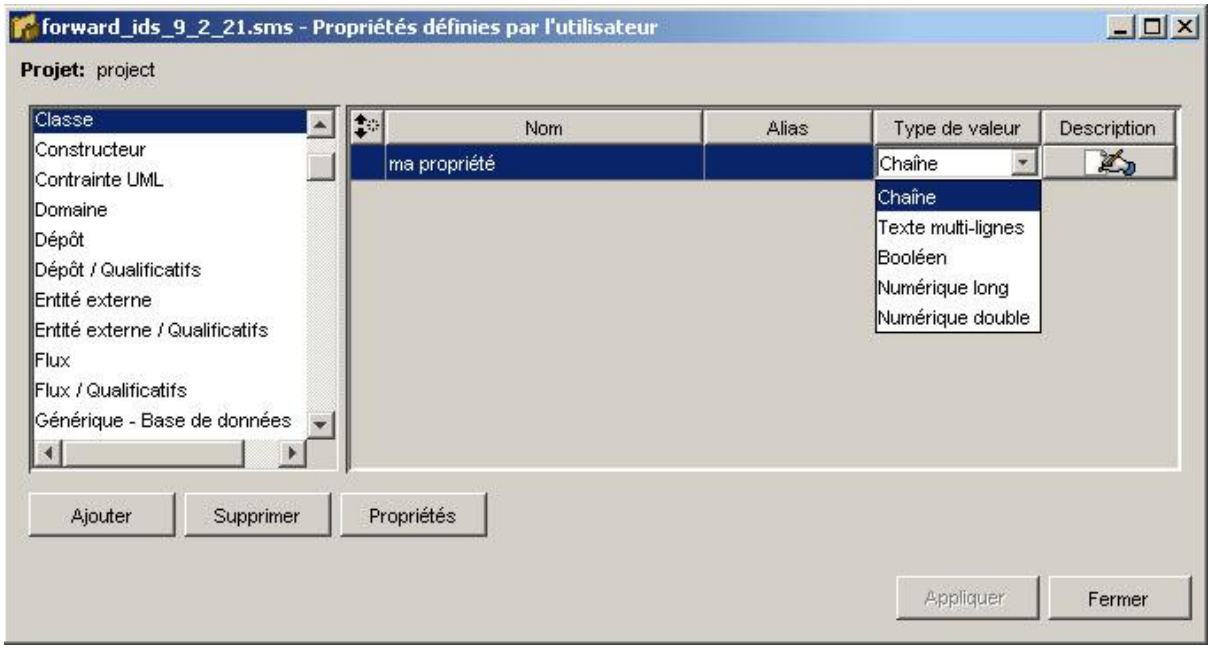
Afficher les contraintes

Même si vous pouvez appliquer une contrainte à n'importe quel objet du modèle, seul les objets suivants permettent l'affichage des stéréotypes : les classes, les paquetages, les tables, les champs, les méthodes, les associations et bouts d'association. Les noms des contraintes sont affichés entre accolades. Pour afficher les contraintes, cliquez **Format > Styles du Projet > l'onglet qui réfère au type d'objet auquel la contrainte est appliquée**. Pour plus amples renseignements sur les Fonctions graphiques,

❖ Créer des propriétés de l'utilisateur

Les propriétés de l'utilisateur sont des attributs avancés pour des concepts du projet. Elles sont définies par l'utilisateur et servent à personnaliser la description de ces concepts. Ainsi, chaque concept peut avoir plusieurs propriétés de l'utilisateur.

Si vous voulez créer vos propres propriétés pour le modèle, vous devez choisir, dans le **menu principal**, **Affichage > Propriétés de l'utilisateurs**.



Dans la fenêtre de propriétés de l'utilisateur, vous devez choisir :

- **Concept** : choisissez le concept du projet pour lequel vous voulez définir une propriété.
- **Nom** : identifiez la propriété en lui donnant un nom.
- **Alias** : identificateur complémentaire pour la propriété (facultatif).
- **Type de valeur** : spécifiez le type de valeur que l'utilisateur entrera à votre propriété. Pour une liste des types de valeurs voir la table à la page suivante.
- **Description** : entrez une description textuelle pour le paquetage (facultatif).

NB: En changeant le Type de valeur, les valeurs déjà entrées seront perdues.

Voici les types de valeurs possibles :

- **Chaîne** : chaîne de caractères de tous genres, avec une longueur illimitée sur une ligne seulement.
- **Texte multi-lignes** : comme la chaîne mais on permet l'utilisation du retour de chariot.
- **Booléen** : vrai ou faux.
- **Numérique et long** : nombres entiers.
- **Numérique et double** : peut comporter des fractions.

IX - Travaux pratiques :

Exercice 1 :

Soient les phrases suivantes :

• Un répertoire contient des fichiers

• Une pièce contient des murs

• Les modems et claviers sont des périphériques d'entrée / sortie

• Une transaction boursière est un achat ou une vente

• Un compte bancaire peut appartenir à une personne physique ou morale

Elaborez les diagrammes de classe correspondants en choisissant le type de relation approprié

Exercice 2 :

1°) Dans un établissement scolaire, on désire gérer la réservation des salles de cours ainsi que du matériel pédagogique (ordinateur portable ou/et Vidéo projecteur).

Seuls les enseignants sont habilités à effectuer des réservations (sous réserve de disponibilité de la salle ou du matériel).

Le planning des salles peut quant à lui être consulté par tout le monde (enseignants et étudiants). Par contre, le récapitulatif horaire par enseignant (calculé à partir du planning des salles) ne peut être consulté que par les enseignants.

Enfin, il existe pour chaque formation un enseignant responsable qui seul peut éditer le récapitulatif horaire pour l'ensemble de la formation.

Modéliser cette situation par un diagramme de cas d'utilisation