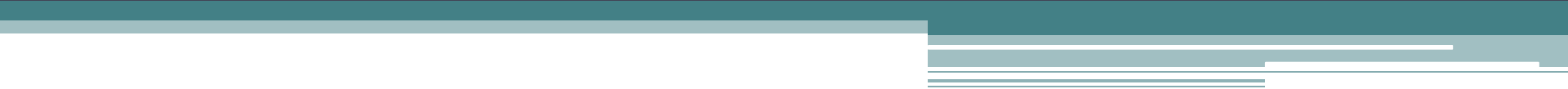


L'analyse et la conception orientées-objet des systèmes d'information.

UML

A decorative graphic element consisting of several horizontal lines of varying lengths and colors (teal, light blue, and white) extending from the left edge of the slide towards the right, positioned below the text.

Analyse

Processus d'examen de l'existant

Conception

Processus de définition de l'application future

M.C.O.O. qu'est-ce que c'est ?

Modélisation et
établir un modèle

Conception
élaborer, représenter quelque chose

Orienté

Objet
vers l'objet

MCOO : modéliser et concevoir en utilisant les objets du monde réel.

Historique des méthodes de développement

Liées aux langages de programmations.

Les méthodes fonctionnelles (1970 –)

Décomposition d'une fonction en plusieurs sous-fonctions.

- Architecture basée sur la décomposition fonctionnelle : peu d'évolutions possibles
- Peu de réutilisation des composants

Les méthodes classiques (1980 –)

Modélisation des données et des traitements.

- JSD : Jackson System Development
- MACH : Méthode d'Analyse et de Conception Hiérarchisée
- MERISE : Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise
- SADT : Structured Analysis and Design Technique
- SASD : Structured Analysis Structured Design

Historique des méthodes de développement

Liées aux langages de programmations.

Les méthodes objet

Objet = abstraction des entités du monde réel

- aide à visualiser les objets tel qu'ils sont ou tels qu'on le souhaite
- permet de spécifier la structure ou le comportement d'un objet
- fournit un guide pour la construction du système
- documente les décisions prises lors de la construction d'un système

Quelques méthodes orientées-objet

- OOA : Object Oriented Analysis, Schlaer/Mellor (1988,91,93)
- OOD : Object Oriented Design, Booch (1991,93)
- OMT : Object Modelling Technique, Rumbaugh (1991,95)
- OOSE : Object Oriented Software Engineering, Jacobson (1992)

Unification des méthodes

Unified Method

- auteurs : Booch, Rumbaugh et Jacobson
- rapprochement de leurs méthodes + cas d'utilisation

Historique de la création d'UML

1995 Début du travail sur la méthode unifiée

1996 Création d'un consortium de partenaires pour travailler sur la définition d'UML dans l'OMG

1997 UML 1.1 (normalisé par l'OMG)

1998 UML 1.2

1999 UML 1.3

2001 UML 1.4

2003 UML 1.5

2005 UML 2.0

Unification des méthodes

Unified Method

- auteurs : Booch, Rumbaugh et Jacobson
- rapprochement de leurs méthodes + cas d'utilisation

Object Management Group – OMG

- Organisation internationale créée en 1989
- Plus de 800 membres (informaticiens et utilisateurs)
- Vise à promouvoir la théorie et la pratique de la technologie objet dans le développement logiciel
- Egalement à la base des standards Meta-Object Facility (MOF), COMmon Request Broker Architecture (CORBA) et Interface Definition Language (IDL)

UML : Unified Modeling Language

- Notation graphique de modélisation à objets
- Permet de visualiser, spécifier, construire et documenter les différentes parties d'un système logiciel
- Langage graphique, basé sur des diagrammes (9 pour UML 1.5, 13 pour UML 2.0)
- **Pas une méthode !** mais utilisable dans tout le cycle de développement d'un logiciel

UML : Unified Modeling Language

- Notation graphique de modélisation à objets
- Permet de visualiser, spécifier, construire et documenter les différentes parties d'un système logiciel
- Langage graphique, basé sur des diagrammes (9 pour UML 1.5, 13 pour UML 2.0)
- **Pas une méthode !** mais utilisable dans tout le cycle de développement d'un logiciel

Buts initiaux des concepteurs d'UML

- représenter des systèmes entiers (pas uniquement logiciels) par des concepts objets ;
- lier explicitement des concepts et le code qui les implantent ;
- pouvoir modéliser des systèmes à différents niveaux de granularité (pour permettre d'appréhender des systèmes complexes) ;
- créer un langage de modélisation utilisable à la fois par les humains et les machines.

Entité du monde réel (ou virtuel) centralisant les données et les traitements associés.

Objet = Identité + Etat + Comportement

Identité d'un objet

Propriété qui permet de distinguer un objet par rapport aux autres sans ambiguïtés.
Distingue deux objets dont les valeurs des différents attributs sont identiques.

Etat d'un objet

Valeur de tous ses attributs à un état donné.
Un attribut est une information qui qualifie l'objet qui la contient.

Comportement d'un objet

Définit l'ensemble des opérations que l'objet peut exécuter en réaction aux messages envoyés par les autres objets.
L'état et le comportement sont liés.

La notion de classe

Une classe est une abstraction qui représente un ensemble d'objets de même nature, c'est-à-dire possédant la même structure statique (attributs) et le même comportement (méthodes).

Exemple : classe Point2D avec les attributs x et y.

Relation entre objet et classe

- Tout objet appartient à une classe et connaît de façon implicite la classe à laquelle il appartient.
- Tout objet est une instance de sa classe.
- Un objet est une instance d'une et une seule classe.

Classe abstraite

Classe conçue sans instances.

Relations entre classes

Association

Relation entre plusieurs classes.

Elles représentent une abstraction des différents liens qui peuvent exister entre les différents objets.

Agrégation et composition

- Forme particulière d'association entre deux classes.
- Exprime le fait qu'une classe est composée d'une ou plusieurs autres classes.

Spécialisation et généralisation

Le mécanisme d'héritage permet de définir de nouvelles classes à partir de classes déjà existantes.

- Généralisation : factorisation dans une classe (appelée super-classe) de propriétés de plusieurs classes
- Spécialisation : inverse de la généralisation, consiste à créer à partir d'une classe plusieurs classes spécialisées.

Notions relatives aux composants des classes

Encapsulation

Possibilité de masquer certains détails de l'implantation. Peut-être réalisé en particuliers par des constituants "privés" des objets.

Polymorphisme

Poly = plusieurs, morphisme = forme

Caractéristique d'un élément qui peut se présenter sous plusieurs formes.

Capacité donnée à une même opération de s'effectuer différemment suivant le contexte de la classe où elle se trouve.

Les diagrammes d'UML

Un diagramme donne à l'utilisateur la possibilité de visualiser et de manipuler des éléments de modélisation.

UML 2.0 comporte 13 diagrammes permettant de modéliser différents aspects d'un système. On se focalisera dans ce cours sur les 9 diagrammes issus d'UML 1.5.

2 catégories de diagrammes UML

- Aspects statiques (structure logicielle et physique) – 5 diagrammes
- Aspects dynamiques (comportements) – 4 diagrammes

Les diagrammes d'UML

Un diagramme donne à l'utilisateur la possibilité de visualiser et de manipuler des éléments de modélisation.

UML 2.0 comporte 13 diagrammes permettant de modéliser différents aspects d'un système. On se focalisera dans ce cours sur les 9 diagrammes issus d'UML 1.5.

Diagrammes des aspects statiques

- 1 Diagramme de cas d'utilisation : représente les fonctions du système du point de vue de l'utilisateur.
- 2 Diagramme de classes : définit les classes et les relations entre les classes.
- 3 Diagramme d'objets : définit les instances des classes, ainsi que les liens entre ces instances.
- 4 Diagramme de composants : permet de décrire les composants (ou modules) d'un système (fichiers sources, fichiers objets, bibliothèques exécutables).
- 5 Diagramme de déploiement : décrit la disposition physique du matériel et la répartition des composants sur le matériel.

Les diagrammes d'UML

Un diagramme donne à l'utilisateur la possibilité de visualiser et de manipuler des éléments de modélisation.

UML 2.0 comporte 13 diagrammes permettant de modéliser différents aspects d'un système. On se focalisera dans ce cours sur les 9 diagrammes issus d'UML 1.5.

Diagrammes des aspects dynamiques

- 6 Diagramme de séquences : représente les interactions d'un point de vue chronologique ; d'une part, les interactions entre le système et les acteurs et, d'autre part, les interactions entre les objets à l'intérieur du système.
- 7 Diagramme de collaboration : représente les interactions entre objets d'un point de vue spatial.
- 8 Diagramme d'états-transitions : automates hiérarchiques permettant représenter les comportements d'un acteur, d'un système ou d'un objet d'une certaine classe.
- 9 Diagramme d'activités : permet de modéliser le comportement d'une méthode en représentant un enchaînement d'activités.

Les diagrammes d'UML

Un diagramme donne à l'utilisateur la possibilité de visualiser et de manipuler des éléments de modélisation.

UML 2.0 comporte 13 diagrammes permettant de modéliser différents aspects d'un système. On se focalisera dans ce cours sur les 9 diagrammes issus d'UML 1.5.

4 diagrammes UML 2.0

- Diagramme de paquetages : permet de décrire les interactions entre les paquetages (paquetage = structure logique de regroupement et d'organisation des éléments du modèle UML).
- Diagramme de structure composite : décrit sous la forme d'une boîte blanche les relations entre les composants d'une classe.
- Diagramme global d'interaction : permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences.
- Diagramme de temps : permet de décrire les variations d'une donnée au cours du temps.

Les aspects statiques

Les diagrammes de cas d'utilisation

Les diagrammes de classes et d'objets

Les diagrammes de composants

Les diagrammes de déploiement

Diagramme des cas d'utilisation

Les cas d'utilisation permettent de décrire le système du point de vue de l'utilisateur. Ils permettent de définir les limites du système et les relations entre le système et son environnement.

Identification des acteurs

Un acteur représente un rôle joué par une personne ou une chose qui interagit avec le système.

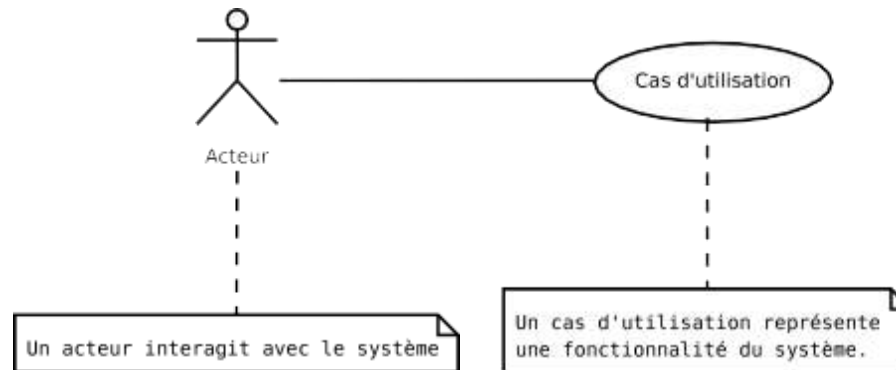
N.B. : Une même personne physique peut correspondre à plusieurs acteurs si celle-ci peut jouer plusieurs rôles.

Acteurs et cas d'utilisation

Cas d'utilisation : fonctionnalité du système utilisée par un acteur.

Un acteur qui interagit avec le système pour utiliser une fonctionnalité est représenté par une relation entre cet acteur et le cas d'utilisation représentant cette fonctionnalité.

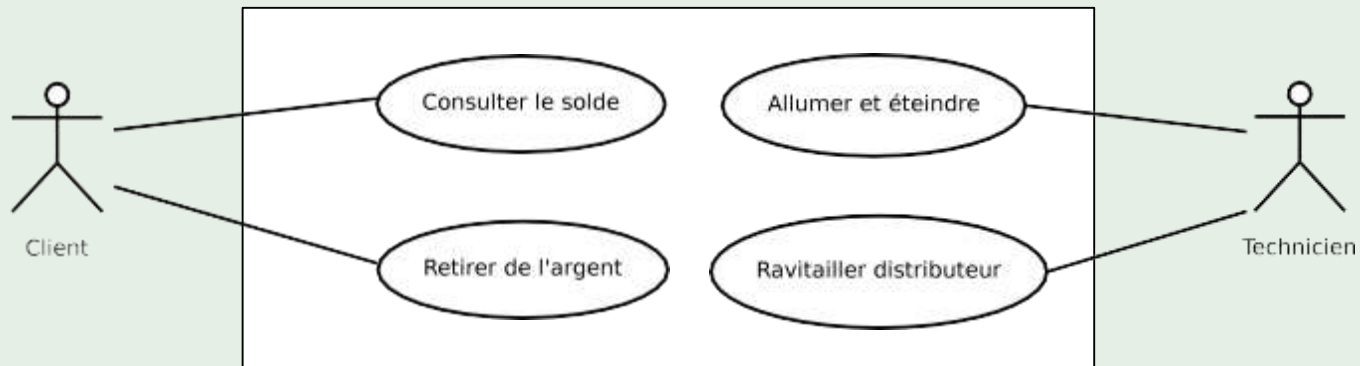
On dit que l'acteur *déclenche* le cas d'utilisation auquel il est lié.



Exemple de cas d'utilisation

Distributeur de billets

- Un premier acteur : "client", consulte le solde de son compte, ou retire de l'argent.
- Un second acteur : "technicien", allume, éteint ou ravitaille le distributeur.

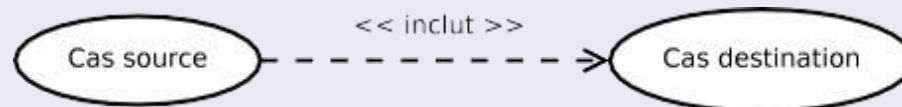


On note qu'un rectangle symbolise les limites du système.

Inclusion entre deux cas d'utilisation

Un cas d'utilisation, nommé *source*, inclut un cas d'utilisation nommé *destination*, si les comportements décrits par le cas source contiennent les comportements décrits par le cas destination.

Notation de l'inclusion

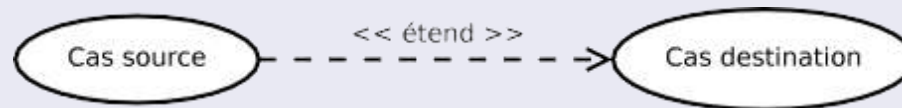


Utilisé en particulier pour factoriser une fonctionnalité partagée par plusieurs cas d'utilisation.

Extension d'un cas d'utilisation

Un cas d'utilisation, nommé *source*, étend un cas d'utilisation nommé *destination*, si les comportements décrits par le cas source étendent les comportements décrits par le cas destination.

Notation de l'extension



L'extension peut être soumise à une condition.

Exemple d'inclusion et d'extension

Système de virements par internet

- Client local : virements (incluant une identification)
- Client distant : virements par internet (étendant le virement standard)

