Chapitre I.
Introduction au module de MCOO

Frédéric DADEAU

Département Informatique des Systèmes Complexes - FEMTO-ST

Bureau 410 C

Email : frederic.dadeau@univ-fcomte.fr

Modélisation et Conception Orientées Objet
Licence 3 – Année 2016-2017

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

Modélisation et

Conception

Orientées

Objet

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

Modélisation et
établir un modèle

Conception
élaborer, se représenter quelque chose

Orientées

Objet
vers l'objet

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

Déroulement du semestre

Sur Moodle : UFR ST > Nouvelle arborescence > Licences > L3 - Informatique > Semestre 6
Clé d'inscription : MCOOL31617

Cours : 5 séances + 1 interrogation écrite

Travaux dirigés : 10 séances (dont 1 interrogation écrite)

Travaux pratiques : 3 séances sur un mini-projet

Historique des méthodes de développement

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 Info. pour les Systèmes d'Entreprise
- SADT : Structured Analysis and Design Technique
- SASD : Structured Analysis Structured Design

Historique des méthodes de développement

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)
2003	UML 1.5
2005	UML 2.0
2009	UML 2.2
2011	UML 2.4

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, 14 pour UML 2.4)
- 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, 14 pour UML 2.4)
- 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.

L'objet

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

Objet = Identité + Etat + Comportement

L'objet

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.

L'objet

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.

L'objet

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 (appellée super-classe) de propriétés de plusieurs classes
- Spécialisation : inverse de le 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.

Plusieurs lectures d'UML

UML peut être vu comme l'aggrégation de plusieurs sous-ensembles :

Plusieurs lectures d'UML

UML peut être vu comme l'aggrégation de plusieurs sous-ensembles :

Les vues

Les vues permettent de décrire le système en fonction d'un point de vue donné : logique, dynamique, architectural, organisationnel, etc.

Plusieurs lectures d'UML

UML peut être vu comme l'aggrégation de plusieurs sous-ensembles :

Les vues

Les vues permettent de décrire le système en fonction d'un point de vue donné : logique, dynamique, architectural, organisationnel, etc.

Les diagrammes

Les diagrammes permettent de décrire les vues (abstraites) sous forme d'éléments graphiques. Un diagramme peut appartenir à plusieurs vues.

Plusieurs lectures d'UML

UML peut être vu comme l'aggrégation de plusieurs sous-ensembles :

Les vues

Les vues permettent de décrire le système en fonction d'un point de vue donné : logique, dynamique, architectural, organisationnel, etc.

Les diagrammes

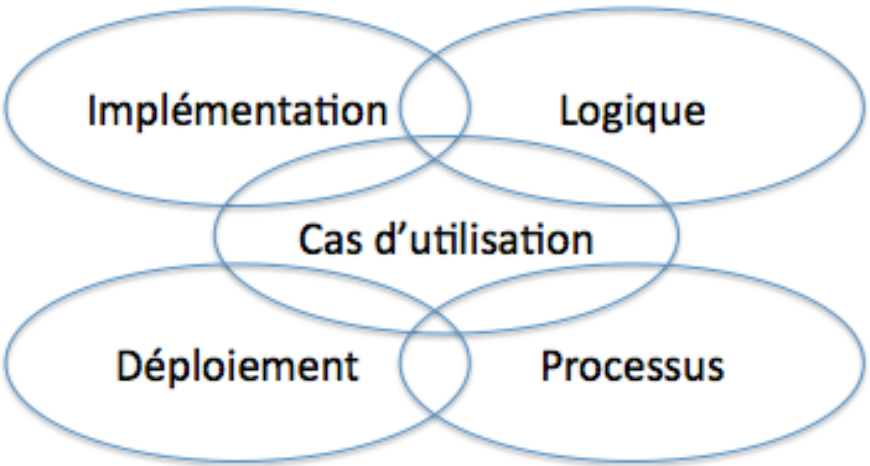
Les diagrammes permettent de décrire les vues (abstraites) sous forme d'éléments graphiques. Un diagramme peut appartenir à plusieurs vues.

Les modèles d'éléments

Les modèles d'éléments sont les composants de bases manipulés dans les diagrammes. Par exemple : les classes, les associations, les états, etc.

Les vues d'UML

UML est communément représenté avec 5 vues qui lui sont associées.

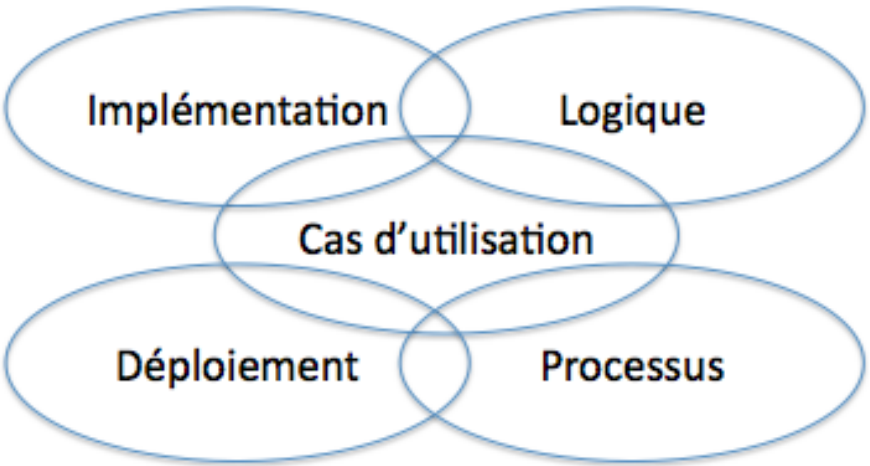


Vue des cas d'utilisation

Répond au QUOI ? et au QUI ?
Vue décrivant le système par rapport aux acteurs (types d'utilisateurs) et aux besoins attendus par ces acteurs.

Les vues d'UML

UML est communément représenté avec 5 vues qui lui sont associées.

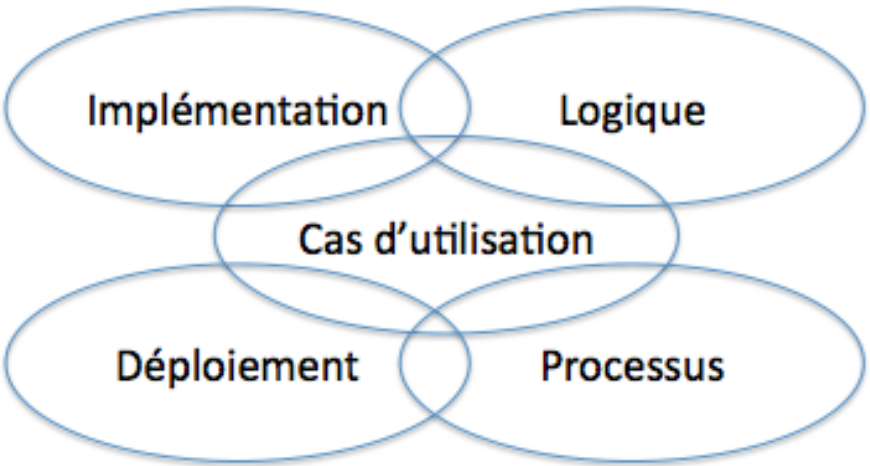


Vue logique

Répond au COMMENT ?
Vue décrivant le système de l'intérieur, expliquant comment celui-ci satisfait les besoins des acteurs.

Les vues d'UML

UML est communément représenté avec 5 vues qui lui sont associées.

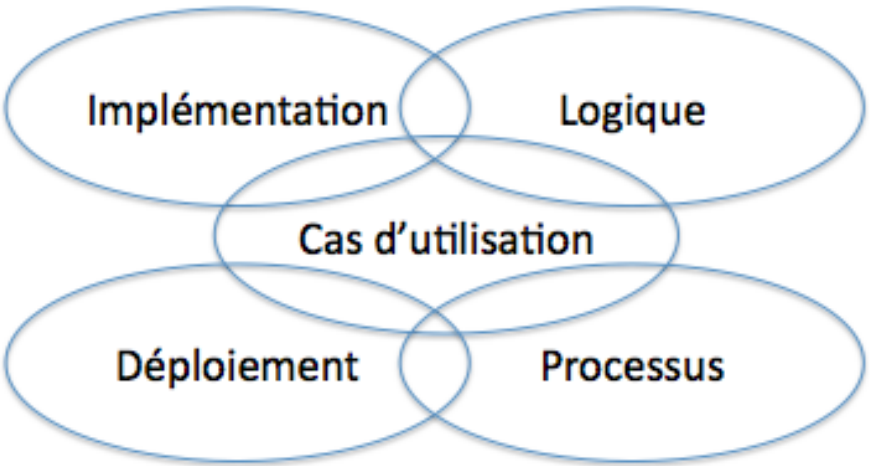


Vue d'implémentation

Vue décrivant les dépendances entre modules.

Les vues d'UML

UML est communément représenté avec 5 vues qui lui sont associées.

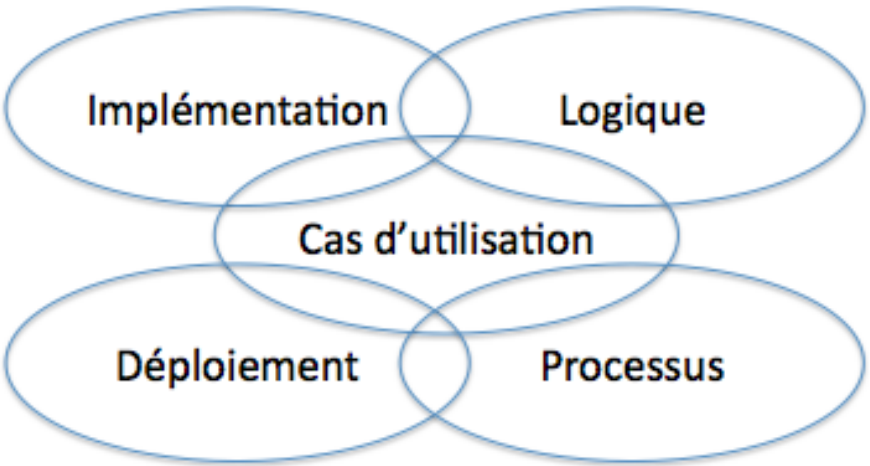


Vue des processus

Vue décrivant les aspects temporels et les processus du système (tâches concurrentes, stimulii, contrôles, synchronisation, etc.)

Les vues d'UML

UML est communément représenté avec 5 vues qui lui sont associées.



Vue de déploiement

Répond au OÙ ?
Vue décrivant l'architecture physique et la disposition du système.

Les diagrammes d'UML

UML 2.4 comporte 14 diagrammes permettant de modéliser différents aspects d'un système. On se focalisera dans ce cours sur les 9 diagrammes issus d'UML 1.5, marqués d'une *.

2 catégories de diagrammes UML

- Aspects statiques (structure logicielle et physique) – 7 diagrammes
- Aspects dynamiques (comportements et interactions) – 7 diagrammes

Les diagrammes d'UML

UML 2.4 comporte 14 diagrammes permettant de modéliser différents aspects d'un système. On se focalisera dans ce cours sur les 9 diagrammes issus d'UML 1.5, marqués d'une *.

Diagrammes des aspects statiques

1. Diagramme de **classes*** : définit les classes et les relations entre les classes.
2. Diagramme d'**objets*** : définit les instances des classes, ainsi que les liens entre ces instances.
3. Diagramme de **paquetages** : permet de décrire les interactions entre les paquetages (paquetage = structure logique de regroupement et d'organisation des éléments du modèle UML).
4. Diagramme de **composants*** : permet de décrire les composants (ou modules) d'un système (fichiers sources, fichiers objets, librairies exécutables).
5. Diagramme de **structure composite** : décrit sous la forme d'une boîte blanche les relations entre les composants d'une classe.
6. Diagramme de **déploiement*** : décrit la disposition physique du matériel et la répartition des composants sur le matériel.
7. Diagramme de **profil** : permet la description de "profils UML", qui sont des stéréotypes et profils s'appliquant sur les modèles d'éléments des diagrammes (par ex. diagramme de classe) et leur donnant une sémantique particulière.

Les diagrammes d'UML

UML 2.4 comporte 14 diagrammes permettant de modéliser différents aspects d'un système. On se focalisera dans ce cours sur les 9 diagrammes issus d'UML 1.5, marqués d'une *.

Diagrammes des aspects dynamiques

8 Diagramme de **cas d'utilisation*** : représente les fonctions du système du point de vue de l'utilisateur.

9 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.

10 Diagramme **global d'interaction** : permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences.

11 Diagramme de **collaboration*** : représente les interactions entre objets d'un point de vue spacial.

12 Diagramme d'**états-transitions*** : automates hiérarchiques permettant représenter les comportements d'un acteur, d'un système ou d'un objet d'une certaine classe.

13 Diagramme d'**activités*** : permet de modéliser le comportement d'une méthode en représentant un enchaînement d'activités.

14 Diagramme de **temps** : permet de décrire les variations d'une donnée au cours du temps.

Plan du cours

- Introduction au module MCOO
- Diagrammes aspects statiques : classe, objet, composants, déploiement
- Diagrammes aspects dynamiques : cas d'utilisation, séquence, collaboration, états-transitions, activité
- Patrons de conceptions (Design Patterns)