Bases de la conception « orientée objet »

DUT Informatique
Semestres 2

Mourad Ouziri
mourad.ouziri@parisdescartes.fr

IUT de Paris Descartes

OMGL
Quelques informations générales

- Contrôles des connaissances
  - 2 DST : 1er DST en mi-semestre et 2ème DST en fin de semestre
  - 1 projet
  - Coefficients : DST1 x 2, DST2 x 2, Projet x 1
- Bibliographie
  - UML 2 : Guide de référence. Booch, Jacobson, Rumbaugh (CampusPress 2004)
  - UML 2. Roques (Eyrolles 2008)

2

OMGL
Aperçu de la matière

- Introduction à la conception d'applications à objets
- Analyse et conception d'application à objets
  - Le langage UML : cas d'utilisation, classes, séquence, états-transitions
  - Le langage de contraintes OCL – *Object Constraint Language*
- Implémentation des modèles de conception en Java
- Tests et validation

3

Introduction à la conception

« orientée objet »

4

Développement logiciel

Motivation (1) – du S1 au S2

- Développer un petit programme
  - Écrire un algorithme/programme : expression du *COMMENT*
  - Travail à l'échelle d'un individu : *Programming-in-the-small*
- Ne convient pas pour aux développement logiciel !
  - Le *QUOI* n'est pas indiqué : que font les résultats ?
  - Pose problème pour le travail en équipe : faible lisibilité et peu transmissible !
  - Résultats produits qu'à la fin du développement !
  - Maintenance difficile et coûteuse !
- Production du logiciel : *Programming-in-the-large*

5

Développement logiciel

Motivation (2) – crise de l'industrie du logiciel

- Enquête sur 365 entreprises PME totalisant 8380 logiciels
  - 31% Abandonnés durant leur développement
  - 52% Dépassement du budget d'un facteur 2 à 3
  - 16% Logiciels conformes aux prévisions initiales
- Enquête sur les grandes entreprises
  - 50% Abandonnés durant leur développement
  - 37% Dépassement du budget d'un facteur 2 à 3
  - 9% Logiciels conformes aux prévisions initiales
- TAUREAUS (Informatisation de la bourse londonienne) abandonné
  Coûts : 4 années de travail et 100.000.000£
- Répartition des coûts du logiciel

6

Développement logiciel

Motivation (3) – crise de l'industrie du logiciel

- Quelques raisons d'échec des logiciels
  - **Mauvaise compréhension des besoins des utilisateurs**
  - **Découverte tardive de problèmes**
  - Logiciel difficile à maintenir ou à faire évoluer
  - Des modules qui ne fonctionnent pas ensemble
  - Problème de coordination dans les équipes
  - Procédures de tests coûteuses
  - …

7

Développement logiciel

Définitions

- Nécessité de surmonter la crise de l'industrie du logiciel
- Naissance du domaine du Génie Logiciel (1968)
- Génie logiciel – *Software Engineering*

  Le génie logiciel désigne l'ensemble des méthodes, des techniques et outils concourant à la **production de logiciels de qualité**
- Logiciel – *Software*

  Ensemble des programmes, procédés et règles (et éventuellement de la documentation) relatifs au fonctionnement d'un ensemble de traitements de l'information *(IEEE Std 729)*

8

Développement logiciel

Qualité du logiciel (1)

- Quelques facteurs de qualité *(McCall – US Air Force, FURPS – HP)*
  - **Conformité fonctionnelle** : satisfaire les besoins fonctionnels des utilisateurs
  - **Maintenabilité** : minimiser l'effort pour localiser et corriger les erreurs
  - **Evolutivité** : minimiser l'effort nécessaire pour le modifier par suite d'évolution des spécifications
  - **Maniabilité** : minimiser l'effort nécessaire pour son apprentissage et son utilisation
  - **Portabilité** : facilité de le transférer d'une plateforme/environnement à un autre
  - **Efficacité** : se limiter à l'utilisation des ressources strictement nécessaires à l'accomplissement de ses fonctions
- CISQ – Consortium for IT Software Quality (créé en 2009 aux USA) pour établir un standard mondial de la qualité du logiciel

9

Développement logiciel

Qualité du logiciel (2)

- Que faire pour obtenir cette qualité du logiciel ?
  - **Modéliser avant de programmer**
  - Structurer : définir une architecture de maintenabilité pour le logiciel
  - Suivre une méthode (processus) de développement : définir des étapes à suivre pour la production du logiciel
  - Répartir : répartir les modules du logiciels pour augmenter ses performances

10

Développement logiciel

Modélisation

- Modèle
  - Représentation abstraite et facilement compréhensible de la réalité
  - Schématisation de la solution avec des symboles (graphiques) intuitifs
  - Vue subjective mais pertinente de la réalité
- Pourquoi modéliser ?
  - Comprendre les besoins ainsi que le monde réel avant de réaliser le logiciel
  - Concevoir le logiciel indépendamment des langages de programmation objets
  - Faciliter la communication entre les contractants du projet
  - Répartir efficacement les tâches de développement
  - Réduction des coûts et des délais
  - Préparer la documentation

11

- Quel langage ? UML : langage de modélisation d'applications objets

Développement logiciel

Étapes de développement et diagrammes UML (1)

- Modélisation des besoins fonctionnels des utilisateurs
  - Les modèles exprimant les besoins des utilisateurs sont une description précise de *ce que le client demande*, et non de comment on peut les réaliser
  - Compréhensible par les experts du métier qui ne sont pas informaticiens
  - Ne comporte aucune décision d'implantation
- Diagramme UML correspondant
  - **Diagramme de cas d'utilisation – DCU**
    + Fiches descriptives des CU

} 1 cours

12

Développement logiciel

Étapes de développement et diagrammes UML (2)

- Modéliser en objet les besoins fonctionnels
  - Identifier les traitements et les données nécessaires au fonctionnement de l'application
  - Analyser le comportement du logiciel face à un besoin
  - Comprendre et tester le comportement des objets constituant le logiciel
- Diagrammes UML correspondants
  - **Diagramme de classes – DC** 3 cours
  - **Diagramme de séquences – DS** 2 cours
  - **Diagramme d'états-transitions – DE** 1 cours

13

En conclusion

- Modéliser avant de réaliser (programmer)
- Le modèle objet est assez intuitif
- A noter que la programmation ne représente qu'une infime part dans le développement logiciel !

14

Le langage UML

15

UML

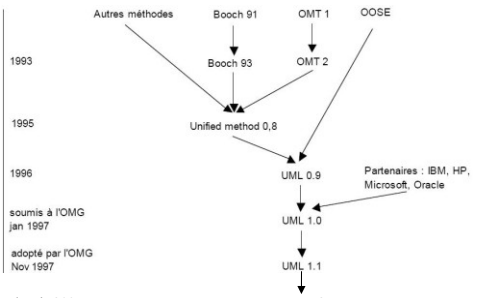
Quelques caractéristiques

- UML : *Unified Modeling Language*
  - Norme de l'OMG (Object Managment Group) depuis 1996
  - Langage de modélisation objet et pas une méthode
  - Universel : indépendant des méthodes et langages de programmation
  - Support de discussion, de communication et de répartition des tâches
  - Intervient à toutes les étapes du développement logiciel : les diagrammes UML
  - Diagramme UML : notation graphique représentant un aspect précis du logiciel

16

UML

Genèse



17

UML

Objectif

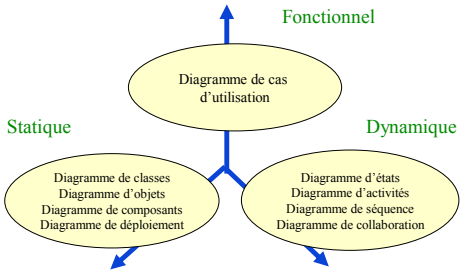
- Objectifs de UML
  - Bien formaliser les besoins utilisateurs et montrer les frontières du projet de développement logiciel
  - Illustrer les réalisations (déroulements) des fonctions principales du logiciel
  - Représenter la structure statique (squelette) du logiciel « orienté objet »
  - Modéliser la dynamique et le comportement des objets
  - Révéler l'implantation physique de l'architecture
- Un langage compréhensible par l'homme et la machine
  - Conception (intellectuelle) humaine et manipulation par la machine
  - Atelier de génie logiciel – AGL
  - Ce qui permet la génération automatique du code

18

UML

Les diagrammes UML

☞ UML : les 9 diagrammes



19

UML

Quelques diagrammes

☞ UML : les quatre diagrammes que nous étudierons

- Diagrammes des cas d'utilisation (DCU) : fonctions de l'application du point de vue des utilisateurs
- Diagrammes de classes (DC) : structure statique d'une application objet
- Diagrammes de séquence (DS) : modèle dynamique représentant les interactions entre les objets d'une application
- Diagrammes d'états-transitions (DE) : comportement des objets d'une classe en terme d'états et de transitions possibles

20

UML

Diagramme de Cas d'Utilisation

21

UML

DCU – Diagramme de Cas d'Utilisation

- Diagramme de cas d'utilisation
  - But : recueillir, comprendre et structurer les besoins utilisateurs
  - DCU : Représentation graphique simple et compréhensible des besoins utilisateurs
  - Définir les frontières du système à modéliser (préciser le but à atteindre)
  - Identifier les besoins fonctionnels requis par les utilisateurs potentiels du logiciel
  - Fournir un document de discussion entre la maîtrise d'ouvrage et la maîtrise d'œuvre

22

UML

DCU – Diagramme de Cas d'Utilisation

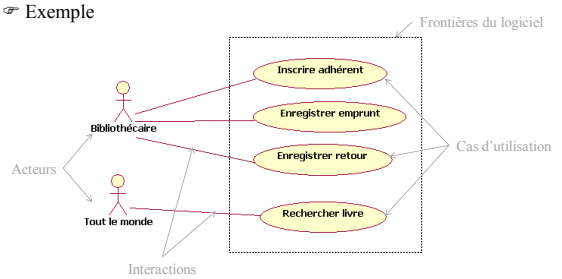
- DCU : **Qui** utilise le logiciel et **Quels** sont ses besoins fonctionnels ?
  - **Qui** : utilisateurs potentiels du logiciel analysé
  - **Quoi** : fonctions du logiciel analysé
- Éléments de modélisation du diagramme de cas d'utilisation
  - **Acteur** (Qui?) : utilisateur potentiel du logiciel
  - **Cas d'utilisation** (Quoi?) : fonctionnalité utile (requise) pour au moins un acteur

23

UML

DCU – Diagramme de Cas d'Utilisation

- Exemple



24

UML

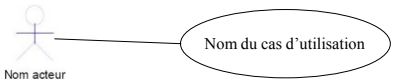
DCU – Diagramme de Cas d'Utilisation

- Acteur
  - Toute entité externe au logiciel et qui interagit avec lui (qui l'utilise)
  - Représente un rôle joué par plusieurs personnes (ou systèmes)
  - La même personne physique peut être représentée par plusieurs acteurs en fonction des rôles qu'elle joue
  - Un acteur n'est pas nécessairement une personne physique : il peut être un service administratif, une société, un système informatique, …
- Représentation de l'acteur

UML

DCU – Diagramme de Cas d'Utilisation

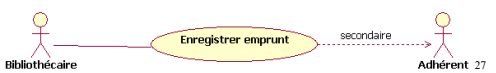
- Cas d'utilisation
  - Comportement du logiciel du point de vue des Acteurs
  - Fonction métier (besoin fonctionnel) requise par au moins un acteur
  - Une fonction métier déclenchée par un acteur (directement ou indirectement)
  - Il modélise à la fois des activités (fonctions) et des communications
- Représentation d'un cas d'utilisation



UML

DCU – Diagramme de Cas d'Utilisation

- Deux catégories d'acteurs pour un CU
  - Acteur principal : toute entité qui utilisent directement un cas d'utilisation
  - Acteur secondaire : une entité qui bénéficie d'un résultat produit par le CU ou celle sollicitée par le logiciel
- Exemple
  - Bibliothécaire : voudrait enregistrer les emprunts pour les adhérents
  - Adhérent : informé de la date limite de remise



UML

DCU – Identifier les acteurs

- Qui utilisera directement les fonctionnalités du logiciel ?
- Qui a besoin du support du logiciel pour effectuer ses tâches quotidiennes ?
- Avec quels autres systèmes le logiciel interagit-il ?
- Qui a un intérêt pour les résultats produits ?
- …

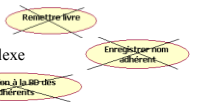
28

UML

DCU – Cas d'utilisation

- Rappel : un modèle (DCU) est un schéma subjectif !
- Les CU peuvent être conçus à différents niveaux de détail !

- Quelques caractéristiques des CU:
  - Fonction informatisable
  - Fonction suffisamment complexe
  - Fonction métier



29

DCU

Fiches descriptives de Cas d'Utilisation

30

UML

DCU – Fiche descriptive des CU

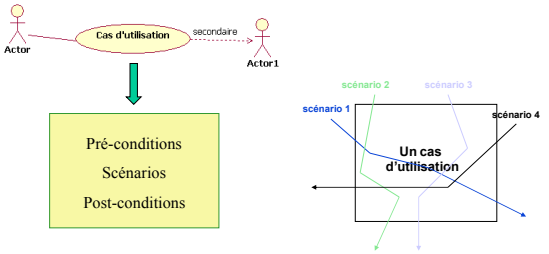
- Limites du DCU
  - Le DCU n'est qu'un sommaire des besoins fonctionnels
  - Nom des CU pas assez explicatif → fonctions pas assez précises
- D'où la nécessité de détailler un CU par une description textuelle
- But des fiches descriptives des CU
  - Donner les scénarios principaux des CU

31

UML

DCU – Fiche descriptive des CU

- Contenu d'une fiche descriptive d'un CU



32

UML

DCU – Fiche descriptive des CU

- Contenu d'une fiche descriptive d'un CU

1. Identification du CU

Nom du CU, date de création, résumé , version, acteurs impliqués

2. Séquences : scénarios possibles du CU

2.1 Pré-conditions : conditions devant être satisfaites pour que le CU puisse se dérouler

2.2 **Enchaînements** : scénarios du CU

Nominal (obligatoire) : scénario le plus probable (favorable)

Alternatifs (optionnel) : scénarios alternatifs au scénario nominal

Exceptionnels (optionnel) : traitement des exceptions (erreurs)

2.3 Post-conditions : conditions devant être satisfaites à la fin du CU et ce quelque soit le scénario

33

UML

Fiche descriptive – Exemple

☞ Fiche descriptive du CU « *Valider la création d'un compte bancaire*»

Nom du cas : Valider la création d'un compte bancaire
Acteur principal : Directeur d'agence
Date : 15/01/2014 **Version** : 1.0
Pré-conditions : des comptes bancaires en attente de validation
Enchaînement nominal

Le CU commence lorsque le client demande un retrait d'espèces

Directeur d'agence	CU – Valider des comptes
1. Demander à valider des comptes	2. Afficher la liste des comptes en attente de validation
3. Choisir un compte	4. Afficher les informations du comptes (propriétaire, justif. de ressources, date de création)
5. Valider le compte	6. Enregistrer la validation + date de validation
	7. Retour au point 2

Post-conditions : Le compte concerné est débité du montant du retrait

34

UML

Fiche descriptive – Exemple

Enchaînement alternatif 1 – Non validation d'un compte

Le cas continue au point 5

5. Annuler la création du compte

6. Enregistrer la non validation du compte et le motif

Enchaînement alternatif 2 – Aucun compte en attente de validation

Le cas continue au point 2

2. Afficher le message « Pas de comptes à valider »

3. Fin de traitement

Enchaînement d'exception – Motif de non validation de compte non renseigné

Le cas continue au point 6 du cas alternatif 1

6. Afficher le message « Motif de non validation obligatoire ! »

7. Retour au point 4

35

UML

DCU – Fiche descriptive des CU

☞ Avantages

– Décrire les CU : meilleure compréhension des fonctions du logiciel

– Faciliter la conception des diagrammes UML (notamment les diagrammes de séquence et de collaboration)

– Préparer les cas de tests

– Préparer la documentation finale du logiciel

36

DCU

Relations entre Cas d'Utilisation

37

UML

DCU – Relations dans le DCU

- Modulariser pour mieux maintenir
- Modulariser pour réutiliser
  - Constat : comportements (traitements) communs à plusieurs CU
  - Conséquence : redondance du travail de développement
  - Modulariser pour éviter les développements redondants
- Types de relations dans le DCU
  - 3 types de relation entre CU : Inclusion, Extension et Héritage
  - 1 type de relation entre acteurs : Héritage

38

UML

DCU – Relations dans le DCU

- **Inclusion de CU**
  - Sémantique : CU1 *inclut* CU2 ssi CU1 fait appel à CU2 dans tous les scénarios possibles
  - CU1 ne peut se terminer avec succès si CU2 ne s'est pas terminé avec succès !
  - Représentation graphique

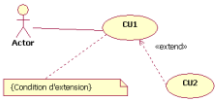
  - Exemple : la bibliothécaire doit être authentifiée pour enregistrer un emprunt ou un retour

39

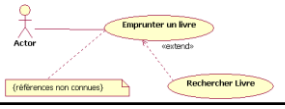
UML

DCU – Relations dans le DCU

- **Extension de CU**
  - Sémantique : CU1 *étend* CU2 ssi CU1 fait appel à CU2 dans certains scénarios
  - L'extension est conditionnelle
  - Représentation graphique



  - Exemple : lorsqu'on emprunte un livre, on peut le rechercher si on ne connait pas ses références exactes



40

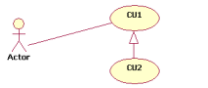
UML

DCU – Relations dans le DCU

- **Héritage entre CU**
  - Sémantique : CU2 *hérite de* CU1 signifie que :

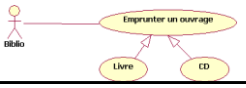
    CU2 est une spécialisation de CU1 (CU1 et une généralisation de CU2)

    CU1 spécialise CU2 en y rajoutant des spécificités
  - Traitements de même nature et substituables
  - Représentation graphique



  - Exemple

    On peut emprunter soit des livres soit des CD



41

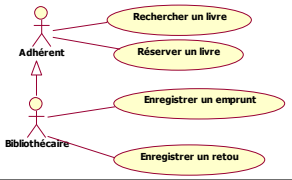
UML

DCU – Relations dans le DCU

- **Héritage entre acteurs**
  - Sémantique :

    Acteur1 *hérite de* Acteur2 si Acteur1 utilise tous les CU de Acteur2
  - Exemple

    La bibliothécaire utilise (a besoin de) tous les CU de l'adhérent



42

UML

DCU – Résumé

- Le DCU exprime les attentes des utilisateurs
  - Fonctions métiers exprimées du point de vue de l'utilisateur
- Il doit être lisible
  - C'est un support de discussion permettant de valider les besoins fonctionnels
- Il inclut des relations
  - Ne pas trop détailler les CU en voulant utiliser systématiquement ces relations
  - Attention ! Ne pas confondre les relations d'héritage et d'extension
- Le DCU est indépendant de l'approche « objet »
  - Peut être utilisé dans n'importe quel processus de développement

43