Analyse et Conception des Systèmes d'Information Orientés Objet

M. BOBO DIALLO

SPÉCIALISTE DE SYSTÈMES D'INFORMATION

Objectifs du cours

- Ingénierie du logiciel
  - Principes et objectifs
- Modélisation avec UML(Unified Modeling Language)
  - Comprendre les concepts clés de l'approche orientée objet.
  - Savoir aborder un problème en se basant sur une approche orientée objet.
  - Maîtriser les principaux diagrammes UML qui permettent de concevoir un système orienté objet.

Hardware et Software

- Systèmes informatiques
  - 80 % de logiciel
  - 20 % de matériel
- Depuis quelques années, la fabrication du matériel est assurée par quelques fabricants seulement
  - Le matériel est relativement fiable
  - Le marché est standardisé
- Les problèmes liés à l'informatique sont essentiellement des problèmes de **Logiciel**

Pourquoi utiliser une modélisation virtuelle?

- Pour enregistrer nos pensées et communiquer en utilisant des langages visuels et schématiques (par ex. UML).
- Parce que :
  - On estime qu'au moins 50% de notre cerveau est impliqué dans le processus visuel.
  - Les langages visuels sont naturels et faciles pour notre cerveau.

Objectifs du GL

- Comment faire des logiciels de qualité?
- Quelles métriques pour jauger la qualité d'un logiciel?

Qualités d'un logiciel (1)

- **Utilité**
  - Adéquation entre le besoin effectif de l'utilisateur et les fonctionnalités offertes par le logiciel

Comment y parvenir?

- Communication avec l'utilisateur: en amont, durant et en aval
  - –Via un langage compréhensible
- Travail avec rigueur et avec abstraction

Qualités d'un logiciel (2)

- **Utilisabilité**

« Efficacité, convivialité et satisfaction avec laquelle les utilisateurs accomplissent des tâches dans un environnement particulier »

  - **Facilité d'apprentissage**
    - Comprendre ce que l'on peut faire avec le logiciel et savoir le faire
    - Documentation bien rédigée
  - **Facilité d'utilisation**
    - Importance de l'effort nécessaire pour interagir avec le logiciel
  - **Comment atteindre ce but?**
    - Analyser le mode opératoire des utilisateurs
    - Adapter l'ergonomie du logiciel aux utilisateurs.

Qualités d'un logiciel (3)

- **Fiabilité**
  - **–Justesse, conformité**
    - •Le logiciel est conforme à sa spécification, les résultats sont ceux escomptés
  - **–Robustesse, sûreté**
    - •Le logiciel fonctionne convenablement en toutes circonstances, rien de catastrophique ne peut survenir même en dehors des conditions d'utilisations prévues.
  - **–Solutions**
    - •Utiliser des méthodes formelles, langage de programmation de haut niveau
    - •Équipement matériel de qualité
    - •Vérification et tests (version alpha ou bêta)

Qualités d'un logiciel (4)

- **Interopérabilité et couplabilité**
  - –Un logiciel doit pouvoir interagir en synergie avec d'autres logiciels
  - –**Solutions**
    - •Bases de données (indépendance données/traitements)
    - •Externaliser certaines tâches en utilisant des middlewares
    - •Utiliser des formats standards (XML) et des protocoles de communication normalisés (Corba)

Qualités d'un logiciel (5)

- **Performance**

–Le logiciel doit satisfaire aux contraintes de temps d'exécution, de débit, d'exactitude et de complétude des réponses

–**Solutions**

- Logiciels simples
- Veiller à la complexité des algorithmes et à leur validité
- Machines plus performantes

Qualités d'un logiciel (6)

- **Portabilité**
  - **Un logiciel doit pouvoir fonctionner sur plusieurs plateformes**
    - Indépendance avec le système d'exploitation
    - Indépendance avec le matériel (plus difficile)
  - **Compiler avec 32 bits et exécuter sur 64 bits n'est pas trivial**
  - **Solutions**
    - Rendre le logiciel le plus possible indépendant de son environnement d'exécution
    - Machines virtuelles

Qualités d'un logiciel (7)

- **Réutilisabilité**
  - **–Réutiliser du code existant**
    - Applications, base de données, …
  - **–Rendre le code réutilisable**
    - Modulaire, échangeable,
  - **–Solutions**
    - Abstraction, généricité (ex: MCD générique de gestion stock)
    - Construire des logiciels à partir de composants

Qualités d'un logiciel (8)

- **Facilité de maintenance**
  - **–La maintenance absorbe une très grande partie des efforts**
    - •Maintenance corrective: corriger les erreurs
    - •Maintenance adaptative: ajuster le logiciel en fonction de l'évolution de son environnement
    - •Maintenance perfective: accroître/améliorer les possibilités du logiciel
  - **–Objectifs**
    - •Réduire la quantité de maintenance corrective
    - •Rendre moins coûteuse les autres maintenances
  - **–Solutions**
    - •Réutilisabilité
    - •Vérification, tests
    - •Structures de données complexes et algorithmes simples
    - •Anticiper les changements à venir

Quelques principes du GL

- **Généralisation**

  –regroupement d'un ensemble de fonctionnalités semblables en une fonctionnalité paramétrable (généricité, héritage)

- **Structuration**

  –façon de décomposer un logiciel (utilisation d'une méthode bottom-up ou top-down)

- **Abstraction**

  –mécanisme qui permet de présenter un contexte en exprimant les éléments pertinents et en omettant ceux qui ne le sont pas.

Phases de réalisation d'un système d'information

M. BOBO DIALLO

SPÉCIALISTE DE SYSTÈMES D'INFORMATION

Phases de développement

- Le cycle de développement classique comporte cinq étapes:
  - **Analyse des besoins:** Déterminer les fonctionnalités que doivent posséder le logiciel
  - **Analyse:** déterminer les tâches et les structures qui se répètent dans le problème
  - **Conception:** s'accorder sur la manière dont le système doit être construit.
  - **Implémentation**: Codage du résultat de la conception.
  - **Test :** Le système est-il conforme au cahier des charges

Analyse des besoins

- Capturer les besoins des clients.
  - –clarifier, filtrer et organiser les besoins, ne pas chercher l'exhaustivité.
- Délimiter les frontières du système.
  - –Spécifier le «quoi» fait par le logiciel.
- Étudier la faisabilité du projet
  - –Faisabilité organisationnelle.
  - –Faisabilité technique.
  - –Faisabilité temporelle
  - –Faisabilité financière

Analyse

- **Analyse du domaine**: identifier les éléments du domaine ainsi que les relations et interactions entre ces éléments.
- **Analyse de l'existant:** déterminer les fonctions principales et réutilisables du système existant.
- **Analyse organisationnelle:** déterminer la structure de l'organisation actuelle.
- **Analyse technique:** recenser les équipements informatiques en place.

Conception

- Définir l'architecture du logiciel
- Définir chaque constituant du logiciel
  - –Informations traitées
  - –Opérations effectuées
  - –Résultats fournis
  - –Contraintes à respecter
- Optimiser les modèles
- Choisir un langage de programmation

Implémentation

- Création des modules et des bases de données
  - –Un module pour réaliser une fonctionnalité donnée
  - –Adaptation et/ou modification des modules existants
- Intégration des différents modules.

Tests

- **Tests unitaires:** permettent de vérifier que chaque module fonctionne correctement indépendamment des autres.
- **Tests d'intégration:** permettent de vérifier que tous les programmes testés individuellement fonctionnent bien ensemble.
- **Tests systèmes**: permettent de vérifier que le système fonctionne correctement dans les conditions réelles d'utilisation.
  - –**Tests Alpha**: faire tester le logiciel par le client sur le site de développement
  - –**Tests Bêta**: faire tester le logiciel par le client sur le site de production

Comment réussir un logiciel?

M. BOBO DIALLO

SPÉCIALISTE DE SYSTÈMES D'INFORMATION

Utiliser des méthodes

- Objectifs
  - –Spécifier et planifier les étapes de l'analyse et de la conception.
- Composition:
  - –**une démarche:** explique la procédure à suivre en exploitant au mieux les principes de modularité, d'abstraction, de réutilisation, etc.
  - –**un formalisme de représentation**: facilite la communication, l'organisation et la vérification.
  - –**des modèles**: facilitent les retours sur la conception et l'évolution des applications.

Des Modèles plutôt que du Code

- Un modèle est la simplification/abstraction de la réalité
- Nous construisons donc des modèles afin de mieux comprendre les systèmes que nous développons
- Nous modélisons des systèmes complexes parce que nous somme incapables de les comprendre dans leur totalité
- Le code ne permet pas de simplifier/abstraire la réalité

Méthodes d'analyse et de conception

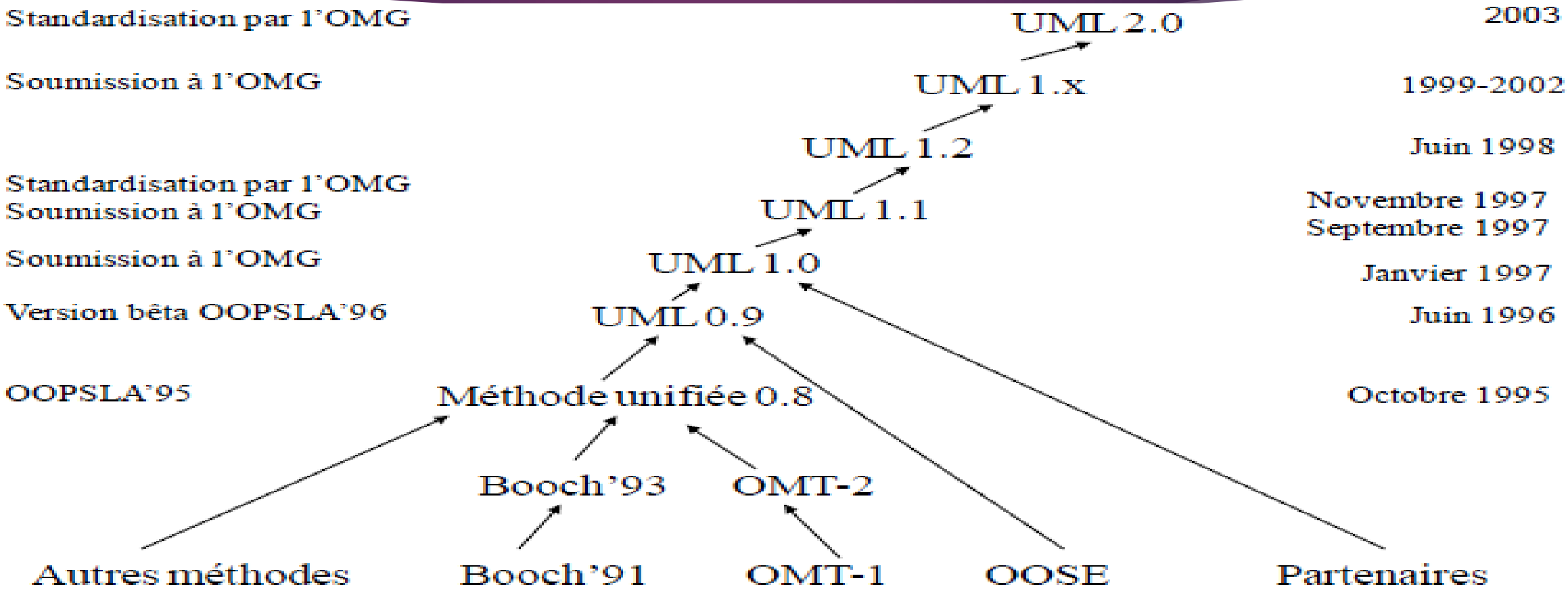
Il existe de nombreuses méthodes:

- **Méthodes fonctionnelles**: hiérarchie de fonction
  - SADT, SA-SD, etc.
- **Méthodes systémiques**: séparation des données et des traitements.
  - Merise, Entité Relation, etc.
- **Méthodes objets**: intégration des données et des traitements dans un objet unique.
  - OMT, OOSE, etc.

Trop de Méthodes orientées objets

- Entre 89 et 94 : le nombre de méthodes orientées objet est passé de 10 à plus de 50
- Toutes les méthodes avaient pourtant d'énormes points communs (objets, méthode, paramètres, …)
- Au milieu des années 90, G. Booch, I. Jacobson et J. Rumbaugh ont chacun commencé à adopter les idées des autres. Les 3 auteurs ont souhaité créer un langage de modélisation unifié

Historique


Standardisation par l'OMG
UML 2.0
2003
Soumission à l'OMG
UML 1.x
1999-2002
UML 1.2
Juin 1998
Standardisation par l'OMG
Soumission à l'OMG
UML 1.1
Novembre 1997
Septembre 1997
Soumission à l'OMG
UML 1.0
Janvier 1997
Version bêta OOPSLA'96
UML 0.9
Juin 1996
OOPSLA'95
Méthode unifiée 0.8
Octobre 1995
Booch'93
OMT-2
Autres méthodes
Booch'91
OMT-1
OOSE
Partenaires

Aujourd'hui

- UML est le langage de modélisation orienté objet le plus connu et le plus utilisé au monde
- UML s'applique à plusieurs domaines
- UML n'est pas une méthode
- Peu d'utilisateurs connaissent le standard, ils ont une vision outillée d'UML (Vision Utilisateur)
- UML est fortement critiqué car pas assez formel
- Le marché UML est important et s'accroît