Analyse, Conception des Systèmes Informatiques

Méthode Analyse Conception
Introduction à UML

O. Boissier, SMA/G2I/ENS Mines Saint-Etienne, Olivier.Boissier@emse.fr, **Septembre 2004**

Génie logiciel

- Définition
  - « Ensemble de méthodes, techniques et outils pour la production et la maintenance de composants logiciels de qualité. »
- Justifications :
  - Evolution des techniques de programmation, du matériel, des besoins

Des problèmes

- Croissance de la taille et de la complexité des systèmes
  - besoins et fonctionnalités augmentent, évoluent
  - technologies en perpétuelle évolution
  - diversification des architectures
- Avec des délais de plus en plus courts,
- et des équipes de plus en plus grosses, avec des compétences multiples

Des principes

- Rigueur et formalisation
- Séparation des problèmes
- Modularité, Abstraction
- Prévision de changements
- Généricité
- Incrémentalité

Qualité du logiciel

- Facteurs externes (cf. utilisateur)
  - Correction (validité)
    - aptitude à répondre aux besoins et à remplir les fonctions définies dans le cahier des charges
  - Robustesse (fiabilité)
    - aptitude à fonctionner dans des conditions non prévues au cahier des charges, éventuellement anormales
  - Extensibilité
    - facilité avec laquelle de nouvelles fonctionnalités peuvent être ajoutées à un logiciel

Qualité du logiciel (suite)

- Compatibilité
  - facilité avec laquelle un logiciel peut être combiné avec d 'autres
- Efficacité
  - utilisation optimale des ressources matérielles (processeur, mémoires, réseau, …)
- Convivialité
  - facilité d 'apprentissage et d 'utilisation
  - facilité de préparation des données
  - facilité de correction des erreurs d 'utilisation
  - facilité d 'interprétation des résultats

Qualité du logiciel (suite)

  - Intégrité (sécurité)
    - aptitude d 'un logiciel à protéger son code contre des accès non autorisés.
- Facteurs internes (cf. concepteur)
  - Ré-utilisabilité
    - Aptitude d 'un logiciel à être réutilisé, en tout ou en partie, pour d 'autres applications
  - Vérifiabilité
    - aptitude d 'un logiciel à être testé (optimisation de la préparation et de la vérification des jeux d 'essai)

Qualité du logiciel (suite)

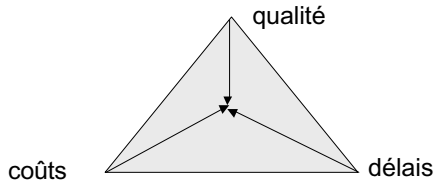
- Portabilité
  - aptitude d 'un logiciel à être transféré dans des environnements logiciels et matériels différents
- Lisibilité,
- Modularité.

Sommaire

- Génie logiciel
- **Conduite de projet informatique**
- Phases de développement
- Modèles de développement
- Méthodes d 'analyse et de conception
- Unification des méthodes objet : UML

Projet

- Ensemble d'actions à entreprendre afin de répondre à un besoin défini dans des délais fixés, mobilisant des ressources humaines et matérielles, possédant un coût.

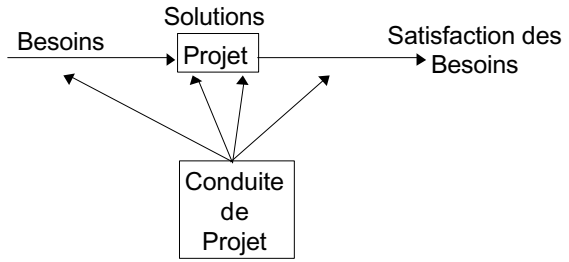


Acteurs d'un projet

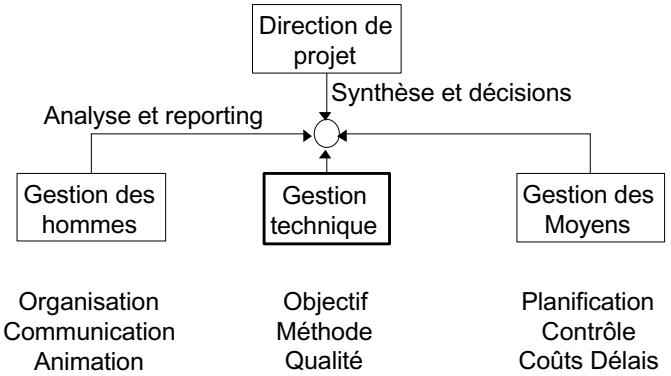
- **Maître d'ouvrage** personne physique ou morale propriétaire de l'ouvrage. Il détermine les objectifs, le budget et les délais de réalisation.
- **Maître d'œuvre** personne physique ou morale qui reçoit mission du maître d'ouvrage pour assurer la conception et la réalisation de l'ouvrage.

Conduite de projet

Organisation méthodologique mise en œuvre pour faire en sorte que l'ouvrage réalisé par le maître d'œuvre réponde aux attentes du maître d'ouvrage dans les contraintes de délai, coût et qualité.



Conduite de projet



Sommaire

- Génie logiciel
- Conduite de projet informatique
- **Phases de développement**
- Modèles de développement
- Méthodes d 'analyse et de conception
- Unification des méthodes objet : UML

Une partie du matériau de ce cours est issue du cours de D. Bardou, UJF, INRIALPES

Phases de développement

- 7 étapes dans la vie d'un logiciel:
  - Planification (Étude de la faisabilité)
  - Spécification des besoins (Requirement analysis)
  - Analyse (Spécification formelle)
  - Conception (Spécification technique)
  - Implémentation (Codage)
  - Tests unitaires
  - Intégration et tests
  - Livraison
  - Maintenance

Planification

- **Objectifs** : identification de plusieurs solutions et évaluation des coûts et bénéfices de chacune d'elles
- **Activités** : simulation de futurs scénarios de développement
- **Sortie** : un schéma directeur contenant
  - la définition du problème
  - les différentes solutions avec les bénéfices attendus
  - les ressources requises pour chacune d'elles (délais, livraison, etc.)

Spécification des besoins

- **Objectifs** : À partir du cahier des charges, description du problème à traiter
  - identification des besoins de l'utilisateur
  - spécification du "quoi" fait par le logiciel : informations manipulées, services rendus, interfaces, contraintes

Spécification des besoins (suite)

- **Activités** :
  - Abstraction et séparation des problèmes
  - Modularisation : séparation des besoins fonctionnels
- **Sorties** :
  - Modèle conceptuel
  - Manuel utilisateur provisoire pour les non informaticiens
  - Plans de tests du système futur (cahier de validation)

Analyse

- **Objectifs :**
  - Répondre au « Que fait le système ? »
  - modélisation du domaine d 'application
  - analyse de l 'existant et des contraintes de réalisation
- **Activités :**
  - Abstraction et séparation des problèmes
- **Sorties :** Modèle conceptuel
  - + dossier de tests d 'intégration

Conception

- **Objectifs :**
  - Répondre au « Comment faire le système ?
  - Décomposition modulaire
- **Activités :**
  - Définition de l 'architecture du logiciel
  - Définition de chaque constituant du logiciel : informations traitées, traitements effectués, résultats fournis, contraintes à respecter
- **Sorties :** Modèle logique
  - + dossier de tests unitaires

Implémentation

- **Objectifs** :
  - Réalisation des programmes dans un (des) langage(s) de programmation
  - Tests selon les plans définis lors de la conception
- **Activités** :
  - Écriture des programmes
  - Tests
  - Mise au point (déboguage)
- **Sorties** : Modèle physique
  - Collection de modules implémentés, non testés
  - Documentation de programmation qui explique le code

Tests unitaires

- **Objectifs** : test séparé de chacun des composants du logiciel en vue de leur intégration
- **Activités** :
  - réalisation des tests prévus pour chaque module
  - les tests sont à faire par un membre de l'équipe n'ayant pas participé à la fabrication du module
- **Sorties** : Rapport de cohérence logique

Intégration et test du système

- **Objectifs** :
  - Intégration des modules et test de tout le système
- **Activités** :
  - Assemblage de composants testés séparément
  - Tests Alpha : l'application est mise dans des conditions réelles d'utilisation, au sein de l'équipe de développement (simulation de l'utilisateur final)
- **Sorties** :
  - Rapport de conformité
  - Démarche d'intégration (ascendante, descendante ou les deux)
  - Conception des données de tests
  - Documentation des éléments logiciels

Livraison, maintenance, évolution

- **Objectifs** :
  - Livraison du produit final à l'utilisateur,
  - Suivi, modifications, améliorations après livraison.
- **Activités** :
  - Tests Bêta : distribution du produit sur un groupe de clients avant la version officielle,
  - Livraison à tous les clients,
  - Maintenance : corrective, adaptative, perfective.
- **Sorties** :
  - Produit et sa documentation
  - Trace d'exploitation et d'évolution

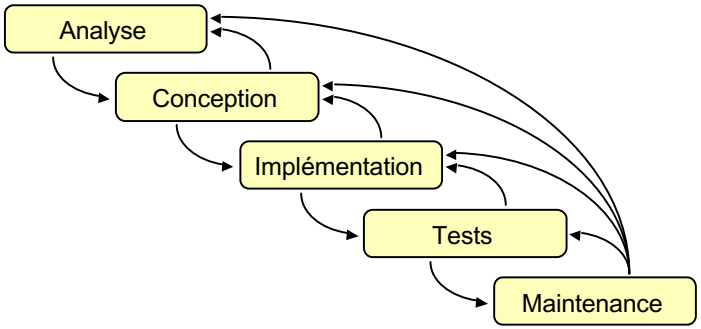
Modèles de développement

- Objectifs :
  - Organiser les différentes phases du cycle de vie pour l'obtention d'un logiciel fiable, adaptable et efficace
  - Guider le développeur dans ses activités techniques
  - Fournir des moyens pour gérer le développement et la maintenance (ressources, délais, avancement, etc.)

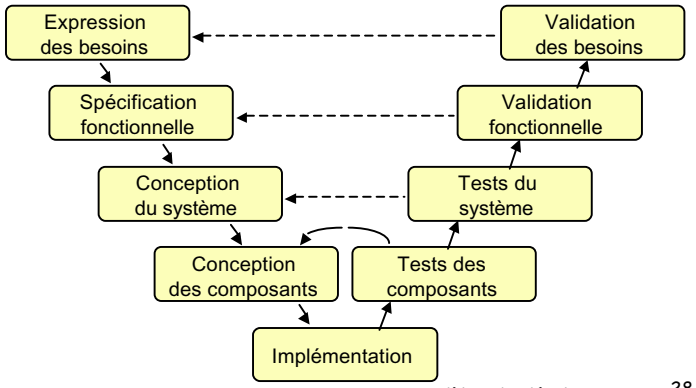
Modèles de développement (suite)

- Modèle "*code-and-fix*"
- Modèle (linéaire) en cascade
- Modèle en V
- Modèle en spirale
- ...

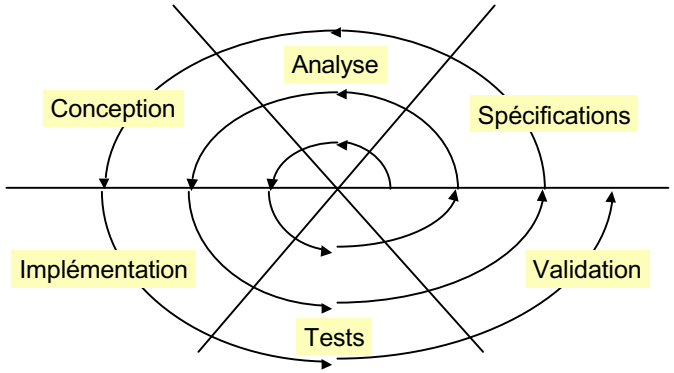
Modèle en cascade



Modèle en V



Modèle en spirale



Sommaire

- Génie logiciel
- Conduite de projet informatique
- Phases de développement
- Modèles de développement
- **Méthodes d 'analyse et de conception**
- Unification des méthodes objet : UML

Méthode d 'analyse et de conception

- Proposition d 'une démarche distinguant les étapes du développement dans le cycle de vie du logiciel (modularité, réduction de la complexité, réutilisabilité éventuelle, abstraction)
- Utilisation d 'un formalisme de représentation qui facilite la communication, l'organisation et la vérification
- production de documents (modèles) qui facilitent les retours sur conception et l'évolution des applications

De nombreuses méthodes ...

- Méthodes fonctionnelles hiérarchiques
  - Data-Flow/SADT/SA-SD, Structure-Chart, ...
- Méthodes données
  - Entité-Relation, MERISE, ...
- Méthodes comportements
  - SA-RT, Réseaux de Pétri, ...
- Méthodes objets
  - OMT, OOA, Classe-Relation, OOD, ...

Unification des méthodes objet

- Constat : au début des années 90, existe une cinquantaine de méthodes objet, liées uniquement par un consensus autour d 'idées communes (objet, classe, sous-systèmes, …) MAIS avec chacune sa propre notation, SANS arriver à remplir tous les besoins et à modéliser correctement les divers domaines d 'application.
- **Recherche d 'un langage commun unique**
  - utilisable par toute méthode objet,
  - dans toutes les phases du cycle de vie,
  - compatible avec les techniques de réalisation actuelles.

  → **UML**
- **Recherche d'un processus de développement commun, unique, … ???**

  → **Unified Process**

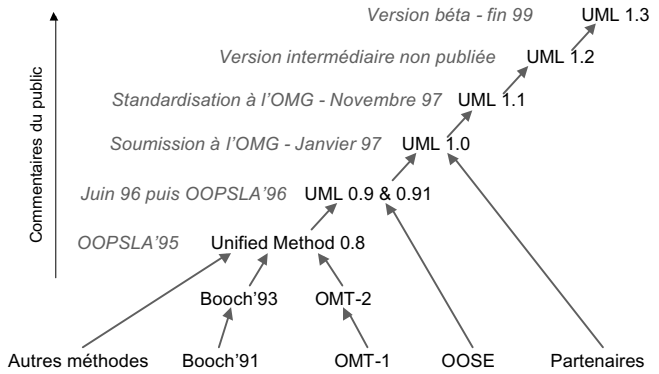
UML

- signifie **U**nified **M**odeling **L**anguage
- est une **notation** basée sur les méthodes Booch, OMT (Rumbaugh), OOSE (Jacobson),
- a été construite afin de standardiser les artéfacts de développement (modèles, notation, diagrammes) SANS standardiser le processus de développement,
- Rôle important de RATIONAL et de l 'OMG

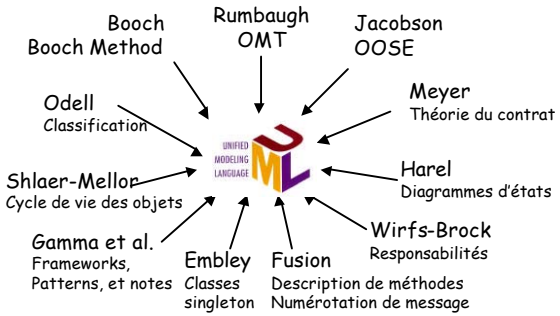
UML (suite) Points forts des sources

- OMT : expressive pour l 'analyse et la conception de systèmes d 'information à base de données,
- BOOCH : expressive durant les phases de design et d 'implantation des projets,
- OOSE : expressive pour l 'analyse des besoins grâce aux « cas d 'utilisation ».

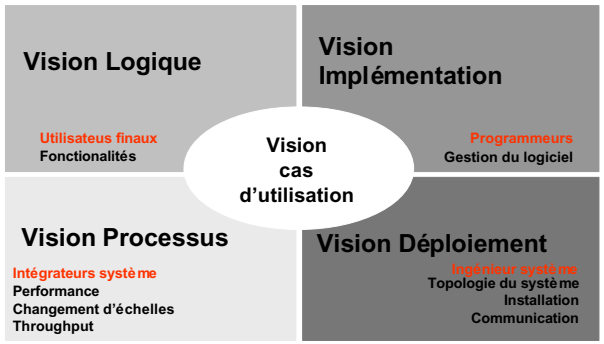
UML (suite) Historique



UML (suite)
Contributions

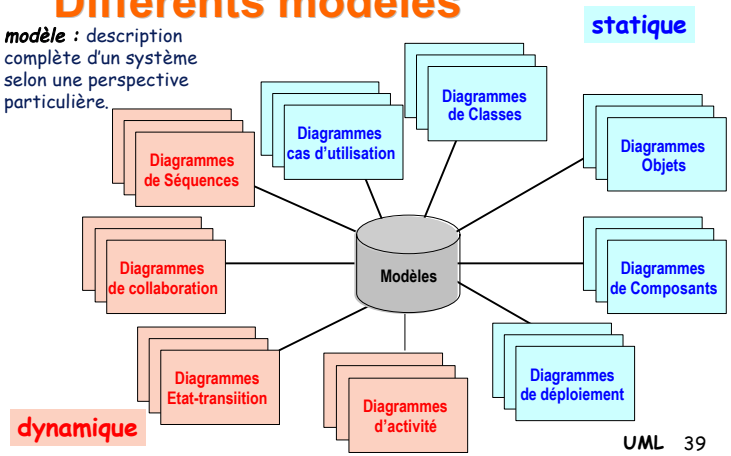


UML (suite)
Différentes vues

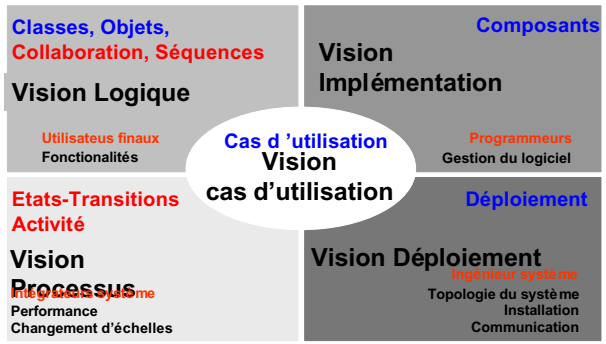


UML (suite)
Différents modèles

modèle : description complète d'un système selon une perspective particulière.



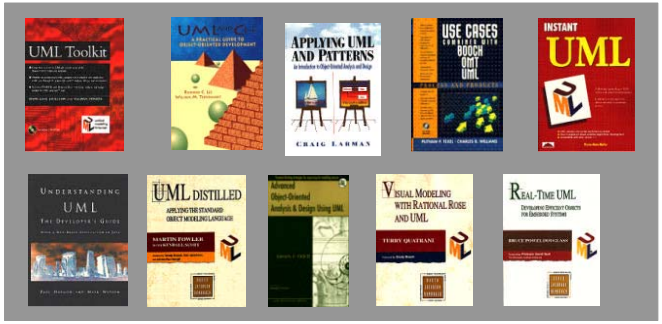
UML (suite)
Différentes vues



UML (suite)
Pour en savoir plus ...

- **The Objectory Software Development Process** Ivar Jacobson, Grady Booch, Jim Rumbaugh, all of Rational Software Corp.
- **Unified Modeling Language Reference Manual** Jim Rumbaugh, Ivar Jacobson, Grady Booch
- **Unified Modeling Language User Guide** Grady Booch, Jim Rumbaugh, Ivar Jacobson -all of Rational Software Corp.

UML (suite)
Pour en savoir plus ...



+ Documentation : http://www.rational.com