UML 2 – Introduction à la modélisation objet

Laurent Audibert

Institut Universitaire de Technologie de Villetaneuse
Département Informatique

23 mars 2010

L'informatisation

- Phénomène le plus important de notre époque
- S'immisce partout
- Au cœur de toutes les grandes entreprises
- Répartissement des investissements dans un système d'information :
  - 20% pour le matériel
  - 80% pour le logiciel
- $\longrightarrow$ La problématique est essentiellement logicielle

Le logiciel

- Qu'est-ce qu'un logiciel ou une application ?

Le logiciel

- Qu'est-ce qu'un logiciel ou une application ?

  ensemble des programmes nécessaires au fonctionnement d'un ensemble de traitements de l'information

Le logiciel

- Qu'est-ce qu'un logiciel ou une application ?

  ensemble des programmes nécessaires au fonctionnement d'un ensemble de traitements de l'information
- Combien de personnes pour développer un logiciel ?

Le logiciel

- Qu'est-ce qu'un logiciel ou une application ?
  ensemble des programmes nécessaires au fonctionnement d'un ensemble de traitements de l'information
- Combien de personnes pour développer un logiciel ?
- Le développement de grands logiciels par de grandes équipes pose d'importants problèmes de conception et de coordination

Les logiciels

Étude du *Standish Group* (1995)

Échantillon représentatif de 365 entreprises,
totalisant 8380 applications

- $16,2\%$ conformes aux prévisions initiales
- $52,7\%$ dépassements en coût et délai d'un facteur 2 à 3 avec diminution des fonctionnalités
- $31,1\%$ abandonnés en cours de développement

Le taux de succès décroît avec la taille des projets et la taille des entreprises.

Les logiciels

Étude du *Standish Group* (1995)

Échantillon représentatif de 365 entreprises,
totalisant 8380 applications

- 16, 2% conformes aux prévisions initiales
- 52, 7% dépassements en coût et délai d'un facteur 2 à 3 avec diminution des fonctionnalités
- 31, 1% abandonnés en cours de développement

Le taux de succès décroît avec la taille des projets et la taille des entreprises.
Causes des échecs : maîtrise d'ouvrage.

Le génie logiciel

Le génie logiciel à pour but de faire face à ces problèmes.

Génie logiciel

ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi

Le génie logiciel

Le génie logiciel à pour but de faire face à ces problèmes.

Génie logiciel

ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi

- La notion de suivi (c.-à-d. de maintenance) occupe une place importante et représente 53% du budget total d'un logiciel.

Le génie logiciel

Le génie logiciel à pour but de faire face à ces problèmes.

Génie logiciel

ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi

- La notion de suivi (c.-à-d. de maintenance) occupe une place importante et représente 53% du budget total d'un logiciel.
- L'une des techniques indispensables en Génie Logiciel est : **La Modélisation !**

Qu'est-ce qu'un modèle ?

Modèle

Représentation abstraite et simplifiée d'une entité du monde réel en vue de le décrire, de l'expliquer ou de le prévoir.

Qu'est-ce qu'un modèle ?

Modèle

Représentation abstraite et simplifiée d'une entité du monde réel en vue de le décrire, de l'expliquer ou de le prévoir.

Modèle météorologique – permet de prévoir les conditions climatiques (modèle prédictif)

Plans en génie civil – pour construire un immeuble, il faut préa-lablement élaborer de nombreux plans (modèle conceptuel) :

- plans d'implantation du bâtiment ;
- plans généraux du bâtiment et de sa structure ;
- plans détaillées des locaux, burreaux, appartements, ...
- plans des cablages électriques ;
- plans d'écoulements des eaux, etc.

Pourquoi modéliser ?

- Mieux comprendre le fonctionnement du système
- Maîtriser la complexité et assurer la cohérence
- Vecteur privilégié pour communiquer
- Mieux répartir les tâches, automatiser certaines d'entre elles
- Réduction des coût et des délais (ex : génération automatique de code)
- Assurer un bon niveau de qualité et une maintenance efficace

Qui doit modéliser ?

Maîtrise d'ouvrage (MOE) : le client qui passe commande d'un produit nécessaire à son activité

Maîtrise d'œuvre informatique (MOA) : personne morale garante de la bonne réalisation technique du produit commandé par la MOA

Qui doit modéliser ?

- Il est préférable que ce soit la MOE.
- Dans la pratique, c'est souvent la maîtrise d'œuvre par manque de professionnalisation de la MOE
  $\longrightarrow$ dans ce cas, mieux vaut privilégier une *méthode de développement Agile*

Le cycle de vie d'un logiciel

Le cycle de vie d'un logiciel

Désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition

Les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement

$\Rightarrow$ il faut définir des jalons intermédiaires permettant la validation du développement logiciel

Étapes du cycle de vie d'un logiciel I

Définition des objectifs : définir la finalité du projet

Analyse des besoins et faisabilité : formalisation des besoins et estimation de la faisabilité

Spécifications ou conception générale : spécification de l'architecture générale du logiciel

Conception détaillée : définition précise de chacun des sous-ensembles du logiciel

Codage (Implémentation ou programmation) : traduction dans un langage de programmation des fonctionnalités

Tests unitaires : validation de l'implémentation de chacun des modules du logiciel

Intégration : validation de l'interfaçage des différents modules

Étapes du cycle de vie d'un logiciel II

Qualification (ou recette) : vérification de la conformité aux spécifications initiales

Documentation : informations nécessaires à l'utilisation et aux développements ultérieurs

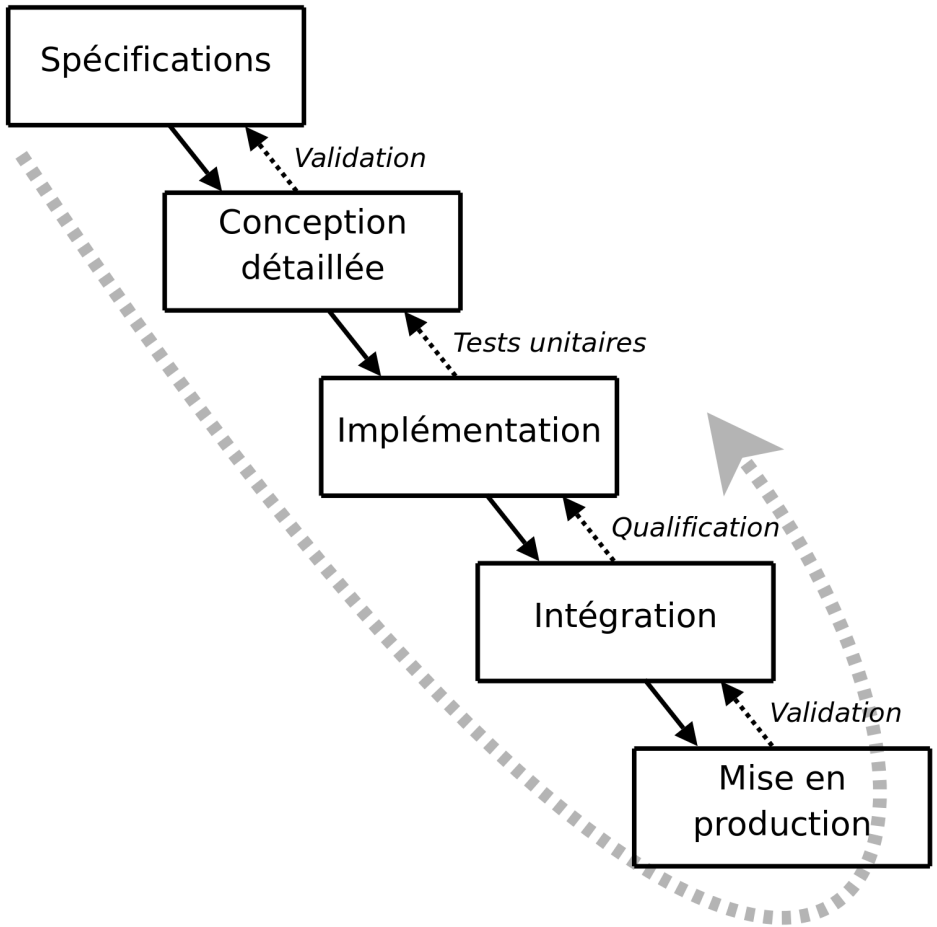
Mise en production : déploiement sur site du logiciel

Maintenance : actions correctives et évolutives du logiciel

Cycle de vie en tunnel

- Ce modèle de cycle de vie cache l'absence de modèle de développement

Cycle de vie en cascade

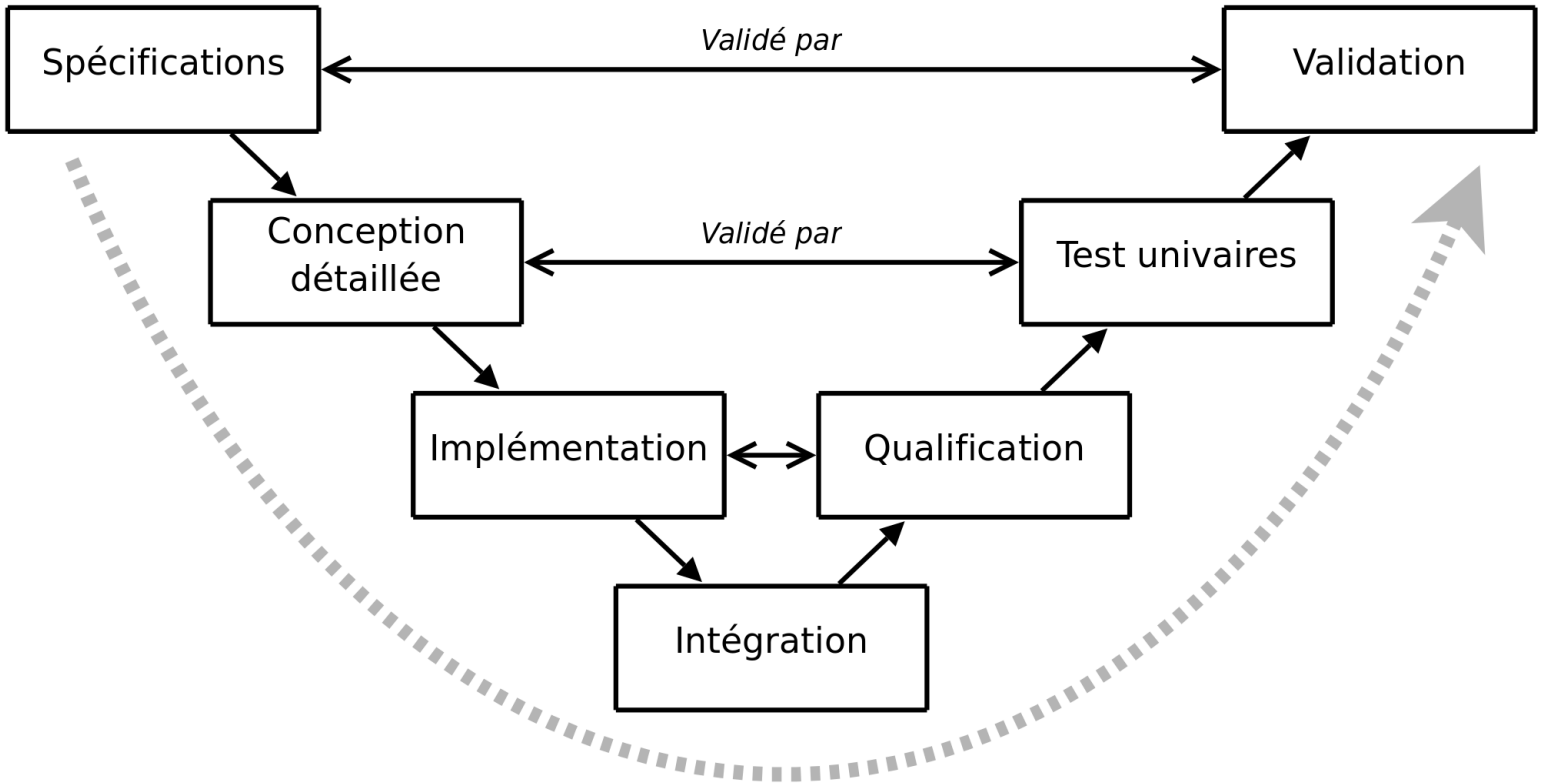


Inconvénients

- Ne supporte aucune erreur sur les étapes passées
- Incapable de prendre en compte des évolutions au cours de son cycle
- Vérification bien trop tardive du bon fonctionnement du système

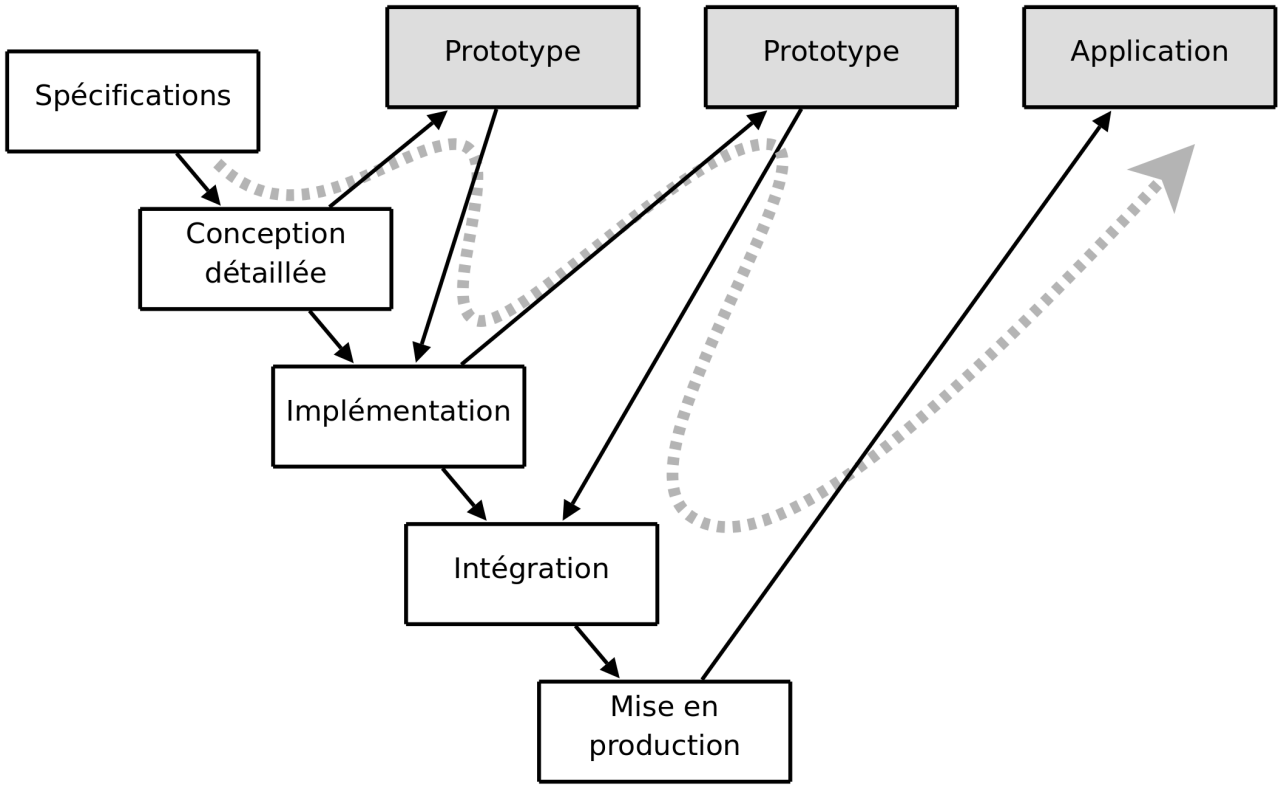
- Enchaîne les phases dans un déroulement séquentiel et linéaire depuis les spécifications jusqu'à la mise en production

Cycle de vie en V



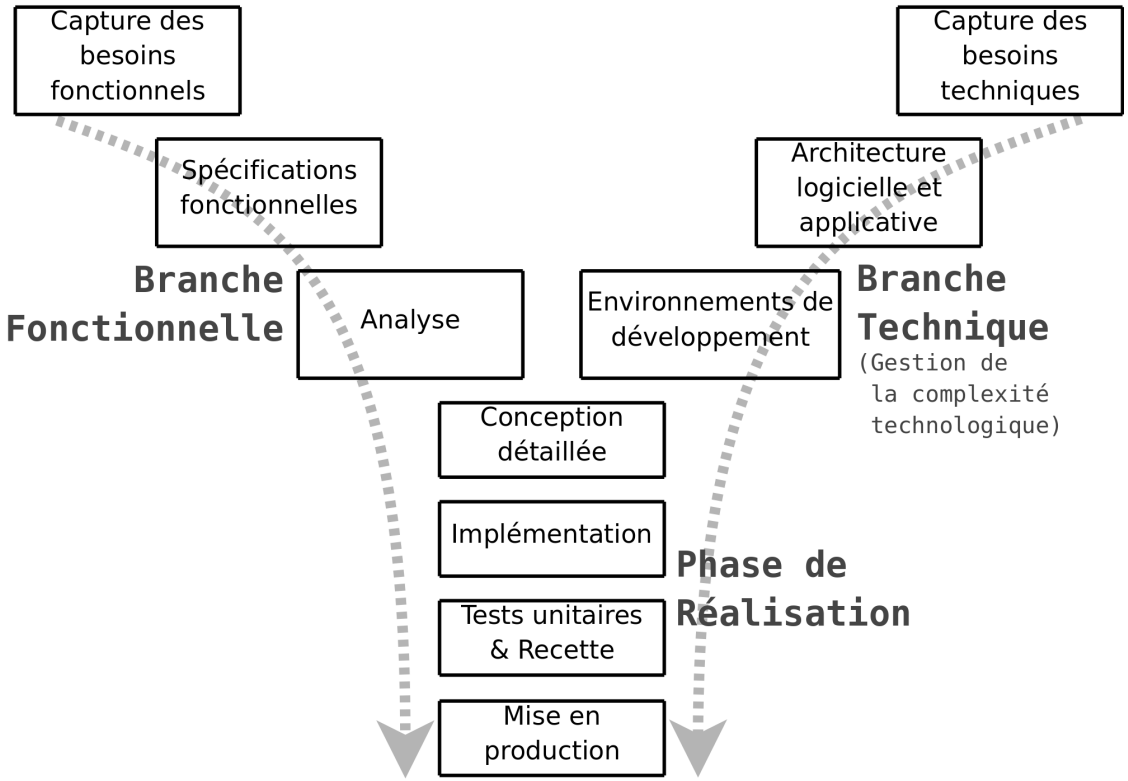
- Variante du modèle en cascade qui met l'accent sur les tests

Cycle de vie en dents de scie



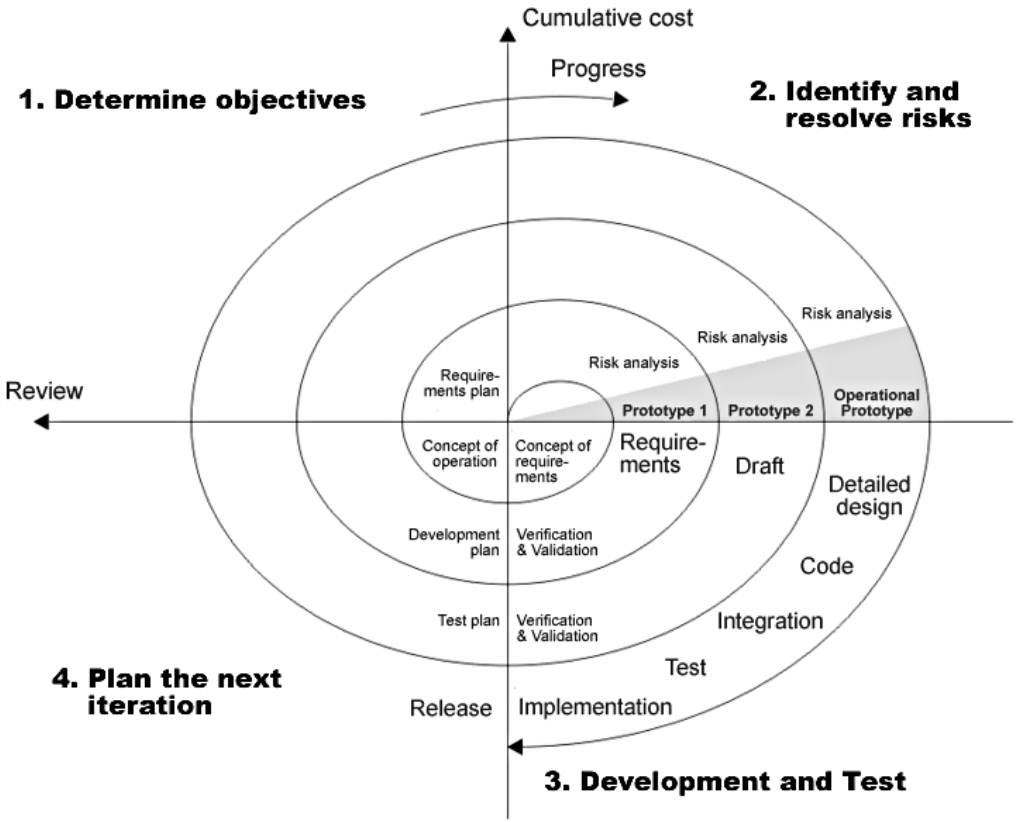
- Variante du modèle en cascade qui met l'accent sur les prototypes

Cycle de vie en Y



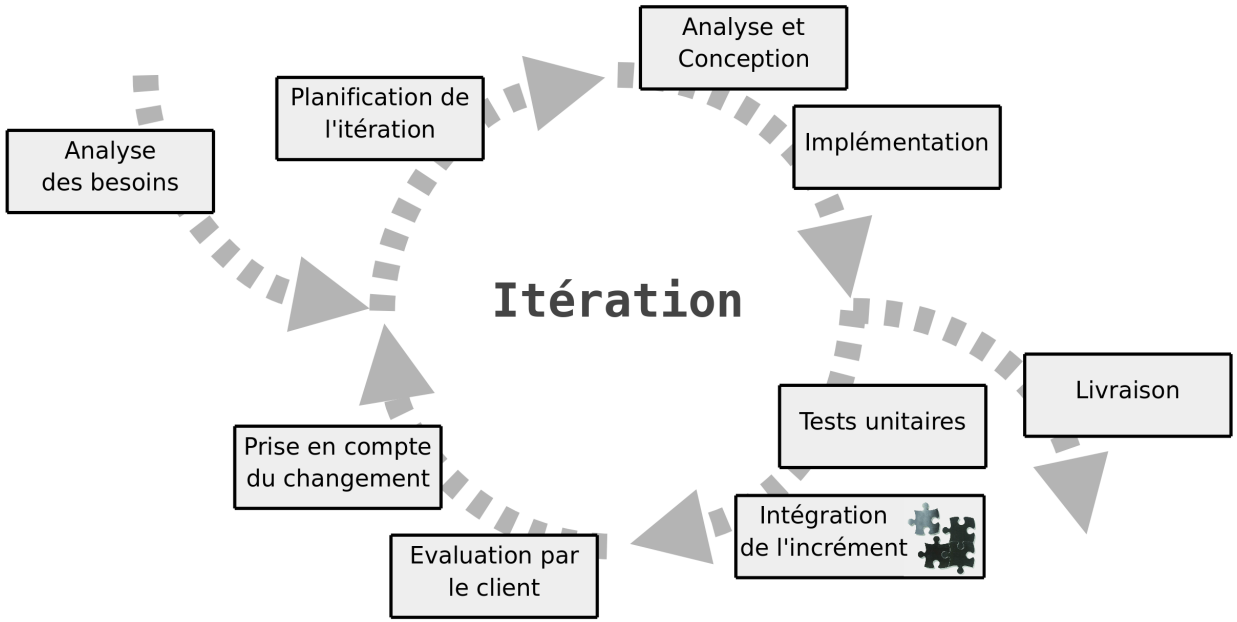
- Modèle qui insiste sur la non corrélation initiale des aspects fonctionnel et technique
- Introduction d'une forme itérative interne à certaines tâches

Cycle de vie en spirale



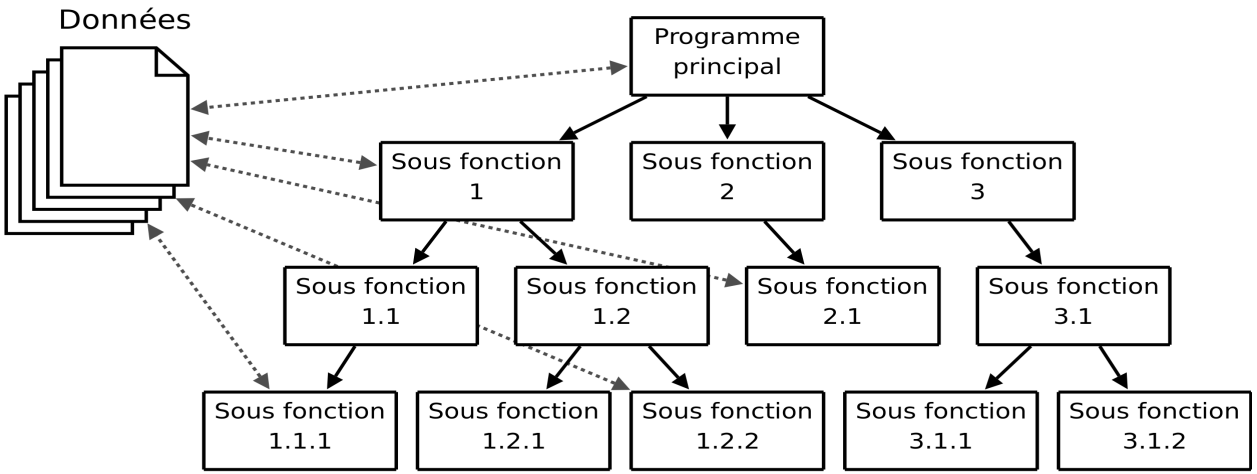
- Modèle qui met l'accent sur l'analyse et la résolution des risques

Cycle de vie itératif et incrémental



- Méta-modèle : chaque itération se déroule selon un cycle de vie classique
- Itération : version partielle mais fonctionnelle de l'application
- Évolution vers le produit final par incréments successifs
- $\longrightarrow$ Vérification continue des risques et de l'adéquation aux besoins
- $\longrightarrow$ Prise en compte du changement
- $\longrightarrow$ Bien adapté à une approche objet

Méthodes fonctionnelles ou structurées



- Approche traditionnelle et intuitive pocédant par décompositions successives jusqu'à obtenir des fonctions simples à implémenter $\longrightarrow$ approche descendante
- Dissociation du problème la représentation des données et du problème du traitement
- **L'architecture du système est dictée par la réponse au problème** (*i.e.* la fonction du système)

L'approche orientée objet

- Approche consistant à identifier les éléments du système et à en faire des objets
- Le logiciel devient une collection d'objets dissociés possédant des caractéristiques structurelles et comportementales
- La fonctionnalité émerge de l'interaction entre les différents objets
- L'état du système est décrit de façon décentralisée par l'état de chacun des objets
- **L'architecture du système est dictée par la structure du problème**

Les caractéristiques d'un objet

Objet

Identité : Elle permet de distinguer les objets, indépendamment de leur état

Caractéristiques structurelles : informations caractérisant l'objet, généralement représentées par des attributs (des variables)

Caractéristiques comportementales : méthodes (fonctions) qui spécifient les actions que l'objet est à même de réaliser

Approche fonctionnelle vs. approche objet

Expressivité identique

- Un langage orienté objet ne permet pas de produire des logiciels impossibles à implémenter selon une approche structurée

Organisation différente

- Les fonctions obtenues à l'issue de la mise en œuvre de l'une ou l'autre approche sont fondamentalement différentes

Évolutivité accrue

- Approche objet $\longrightarrow$ changement de l'interaction des objets
  $\longrightarrow$ limitation et bonne localisation des modifications à opérer
- Approche fonctionnelle $\longrightarrow$ modifications difficiles à localiser et se propageant souvent sur une grande partie de la structure
  $\longrightarrow$ dégénérescence, ou remise en question une profonde, de la structure hiérarchique

Conséquence naturelle et ultime de la modularisation

Cartes perforées (1800) $\longrightarrow$ assembleur (1947) $\longrightarrow$ langages "goto" (1956)
$\longrightarrow$ programmation structurée (1970) $\longrightarrow$ programmation orientée objet (1990)

Concepts importants de l'approche objet

Classe

Type de données abstrait, caractérisé par des propriétés (attributs et méthodes) communes à tout une famille d'objets.

Encapsulation

En définissant une interface

- Masquer des détails d'implémentation de l'objet
- Interdire, ou restreindre l'accès direct aux attributs de l'objet

$\longrightarrow$ stabiliser l'utilisation de l'objets

Concepts importants de l'approche objet

Héritage / Spécialisation / Généralisation

Héritage : Mécanisme de transmission des propriétés d'une classe vers une sous-classe

Spécialisation : Une classe peut être spécialisée, afin d'ajouter ou d'adapter ses caractéristiques

Généralisation : Plusieurs classes peuvent être généralisées en une classe qui les factorise

$\longrightarrow$ construction de hiérarchies de classes $\longrightarrow$ limiter la duplication (redondance) $\longrightarrow$ favoriser la réutilisation

Concepts importants de l'approche objet

Polymorphisme

Faculté d'une opération de s'appliquer à des objets de classes différentes et d'avoir un comportement adapté à ces objets

- Polymorphisme ad hoc (surcharge)
- Polymorphisme paramétrique (généricité)
- Polymorphisme d'héritage (spécialisation) : invoquer une opération définie pour un paramètre de type X avec un paramètre de type Y héritant du type X.

Élément essentiel de l'approche objet $\rightarrow$ augmente la généricité (donc la qualité) du code

Historique la programmation par objets

Simula – Simula I (1961-64) et Simula 67 (1967), premiers langages de programmation objets, sont conçus par Ole-Johan Dahl et Kristan Nygaard

Smalltalk – Alan Kay réalise en 1976 Smalltalk

C++ – Bjarne Stroustrup met au point C++ aux Bell Labs d'AT&T en 1982. Sa standardisation ANSI / ISO date de 1997

Java – Java est lancé par Sun en 1995

Introduction

Pour programmer une application, il ne faut pas de se lancer tête baissée dans l'écriture du code : il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation

C'est cette démarche antérieure à l'écriture que l'on appelle *modélisation* ; son produit est un *modèle*

Histoire des modélisations par objets

1990-95 : Plus de 50 méthodes de modélisation objets apparaissent

1994 : Consensus autour de 3 méthodes : OMT, OOD, OOSE

1995 : Méthode unifiée, *Unified Method 0.8* (OMT + OOD)

1996 : UML 0.9 (*Unified Method 0.8* + OOSE)

1996 : UML 1.0 (IBM, Microsoft, Oracle, DEC, HP, Rational, Unisys etc.)

1997 : L'OMG adopte UML 1.1

2005 : UML 2.0

UML est une norme qui s'impose et à laquelle tous les grands acteurs du domaine se sont rangés

UML n'est pas une méthode

UML n'est pas une méthode

C'est un langage graphique qui permet de représenter les divers aspects d'un système

UML permet de donner des *vues* partielles du système

UML comporte plus d'une dizaine de diagrammes standard qui se répartissent en deux catégories

Diagrammes UML