Cours Génie Logiciel Avancé
Chapitre 2 : Gestion de projet informatique

INSTITUT SUPERIEUR INFORMATIQUE ISI
المعهد العالي للإعلامية

Responsables du cours :
Héla Hachicha
Hatem Ben Sta

Année Universitaire : 2014 - 2015

Plan du cours

- Chapitre 1 : Introduction au génie logiciel
- Chapitre 2 : Gestion de projet informatique
- Chapitre 3 : Processus Unifié de Modélisation
- Chapitre 4 : Activités dans le Processus Unifié
- Chapitre 5 : Les Design Pattern
- Chapitre 6 : MDA : Model-Driven Architecture
- Chapitre 7 : Le langage de contraintes OCL
- Chapitre 8 : Méta-modélisation MOF
- Chapitre 9 : Les profils UML
- Chapitre 10 : MDE: Model Driven Engineering

Sommaire

- Chapitre 2 : Gestion de projet informatique
  - Cycle de vie des logiciels
  - Processus de développement de logiciel
  - Méthodes de conception
  - Gestion de projet

Cycle de vie des logiciels

Définitions de base

- Le développement des systèmes logiciels de grande taille nécessite :
  - beaucoup de temps et d'expérience
  - un ensemble d'étapes distinctes guidant le développement et l'utilisation.

- Le cycle de vie d'un logiciel est composé de l'ensemble des étapes : distinctes et guidant le développement et l'utilisation du logiciel.

Définitions de base

- Principalement, 5 étapes composent le cycle de vie d'un logiciel :

1. Analyse et définition des besoins.
2. Conception du système et du logiciel.
3. Réalisation et tests unitaires.
4. Tests du système.
5. Utilisation et maintenance.

Analyse et définition des besoins

- **Entrée :**

  Données fournies par des experts du domaine d'application et les futurs utilisateurs.

- **Méthodes :**

  - Il faut surtout établir un dialogue avec les experts du domaine, qui ne sont pas forcément des informaticiens.

  - Utiliser des méthodes plutôt cognitives : entretiens, questionnaires, observations de l'existant, études de situations similaires.

Analyse et définition des besoins

- **Démarche :**

  Pour établir les besoins (le cahier des charges), il faut étudier le domaine d'application :
  - l'état actuel de l'environnement du futur système ;
  - le rôle du système ;
  - les ressources disponibles et requises ;
  - les contraintes d'utilisation ;
  - les performances attendues ; …

- **Résultat :**

  Le cahier des charges du logiciel (spécification des besoins) décrivant :
  - l'environnement du futur système
  - son rôle
  - son utilisation (manuel d'utilisation préliminaire)
  - si nécessaire, partage entre matériel et logiciel

Conception du système

- **But :**
  - établir une première description du futur système
  - répondre à la question comment : comment réaliser le logiciel défini par le cahier des charges
- **Entrée:**
  - analyse des besoins + considérations techniques et faisabilité informatique
- **Méthodes:**
  - SADT, MERISE, UML, ...

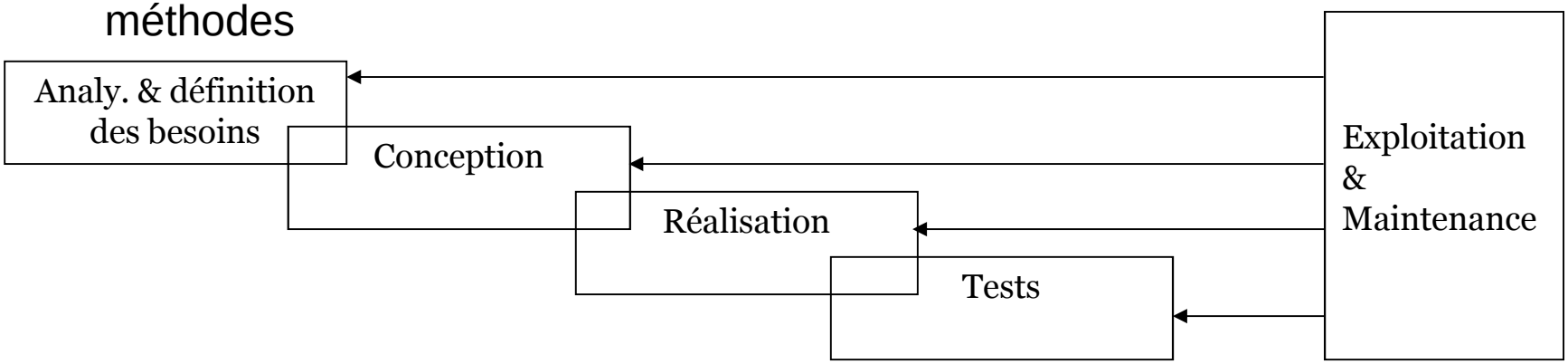
Réalisation et tests unitaires

- **But :**
  - Réalisation, à partir de la conception détaillée, d'un ensemble de programmes ou de composants de programmes
  - Projection des concepts du modèle sur un langage de programmation.

    -- Il s'agit d'un processus manuel ou semi-automatique
- **Entrée :**
  - conception détaillée + contraintes de l'environnement de programmation
- **Résultat :**
  - documents décrivant :
    - code source de chaque module
    - directives: compilation, liens, ...
    - documentation d'implémentation

Remarques

- ***les différentes étapes données précédemment :***
  1. ne sont pas disjointes : il est possible de trouver des recouvrements entre elles ;
  2. provoquent des retours d'information d'une étape donnée à sa précédente ;
  3. ne sont pas forcément appelées de la même manière par toutes les méthodes



- ***Avec les tests, deux étapes peuvent exister :***

  1. la vérification du système : « Est-ce que nous construisons bien le système ? »

  2. la validation du système : « Est-ce que nous construisons le bon système ? »

Remarques

- ***Vérifier un système revient à s'assurer que le système :***

  1. ne présente ni des contradictions ni des erreurs
  2. respecte certaines règles d'utilisation et des propriétés prédéfinies.

- ***Validation du système :***

  1. se fait en collaboration avec le client
  2. consiste à vérifier que les fonctionnalités demandées par l'utilisateur sont bien assurées par le système.

Les processus de développement de logiciel

Définitions de base

- **Processus de développement de logiciel :**

  1. Ensemble d'étapes ordonnées avec un déroulement séquentiel ou parallèle.
  2. Est aussi appelé : Modèles de développement de logiciel
  3. Il s'agit de l'ordonnancement des différentes étapes :
     - Analyse et définition des besoins ;
     - Conception du système et du logiciel ;
     - Réalisation et test unitaires ;
     - Tests du système ;
     - Utilisation et maintenance.
  4. Ces étapes peuvent être utilisées différemment

→ plusieurs types de processus existent

Processus de développement

- Historique :
  - Fin des années 60 : le modèle en cascade.
  - Durant les années 80 : le modèle en V (le plus utilisé aujourd'hui).
  - Plus récemment le modèle en spirale, plus complet, mais également plus complexe a introduit de nouveaux aspects comme l'analyse des risques et l'utilisation systématique de prototypes pour s'assurer de la bonne compréhension des besoins de l'utilisateur final.

Modèle en cascade

- Modèle très simple nécessite d'avoir un ensemble d'étapes (Boehm 97)
- Chaque étapes doit se terminer pour commencer la suivante.
- Des documents sont produits pour concrétiser la réalisation de chaque étapes.

Critique du modèle en cascade

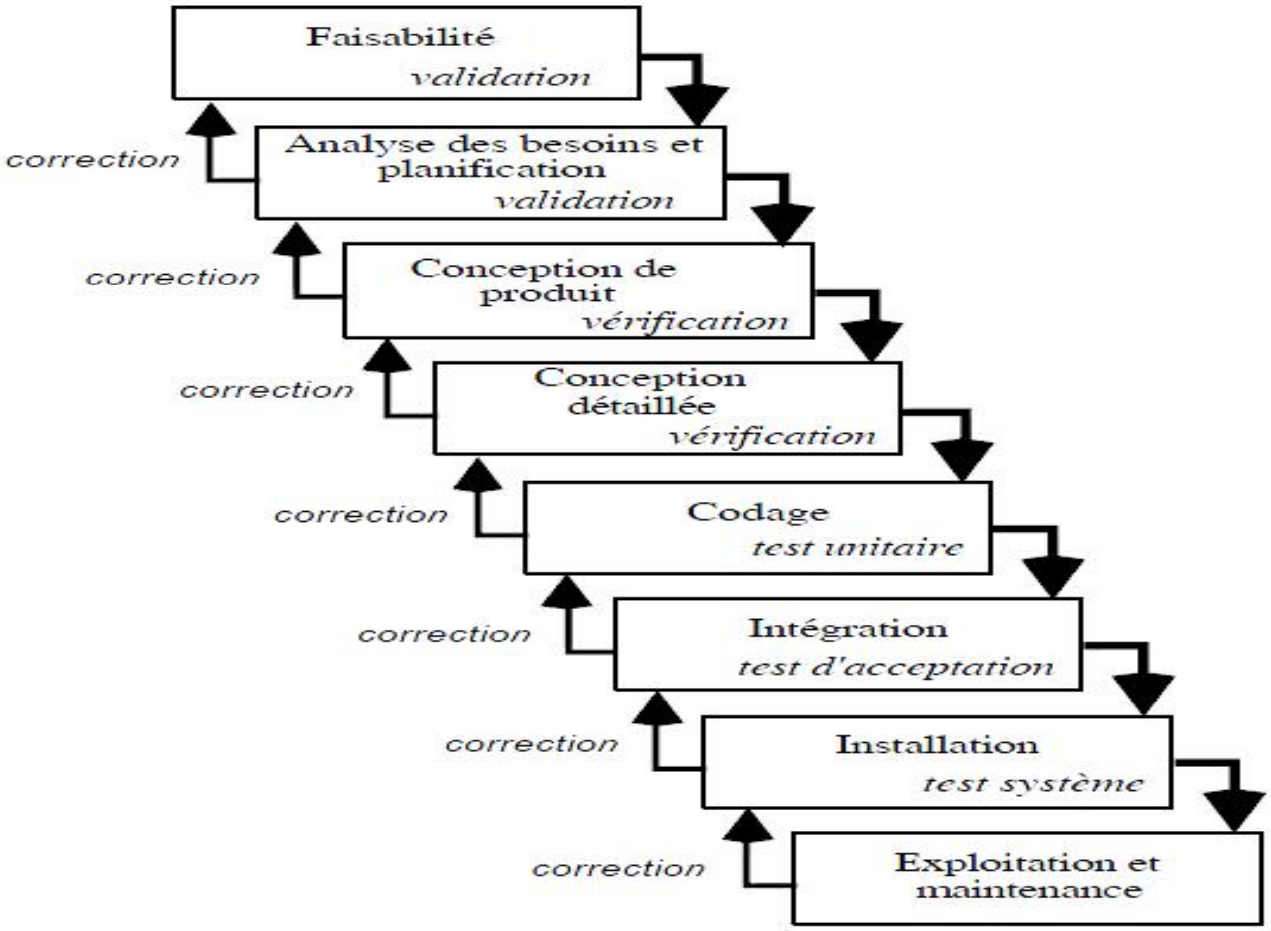
- Beaucoup de documentations
- Réponses difficile aux changements de spécifications
- Utilisation uniquement à partir de la phase de maintenance

➔ Le modèle en cascade rend coûteux le développement itératif puisque la rédaction des documents de validation de chaque phase demande beaucoup de travail.

➔ Ce modèle est inadapté au développement de systèmes dont la spécification est difficile à formuler *a priori*.

Modèle en cascade

- ***Schématiquement :***

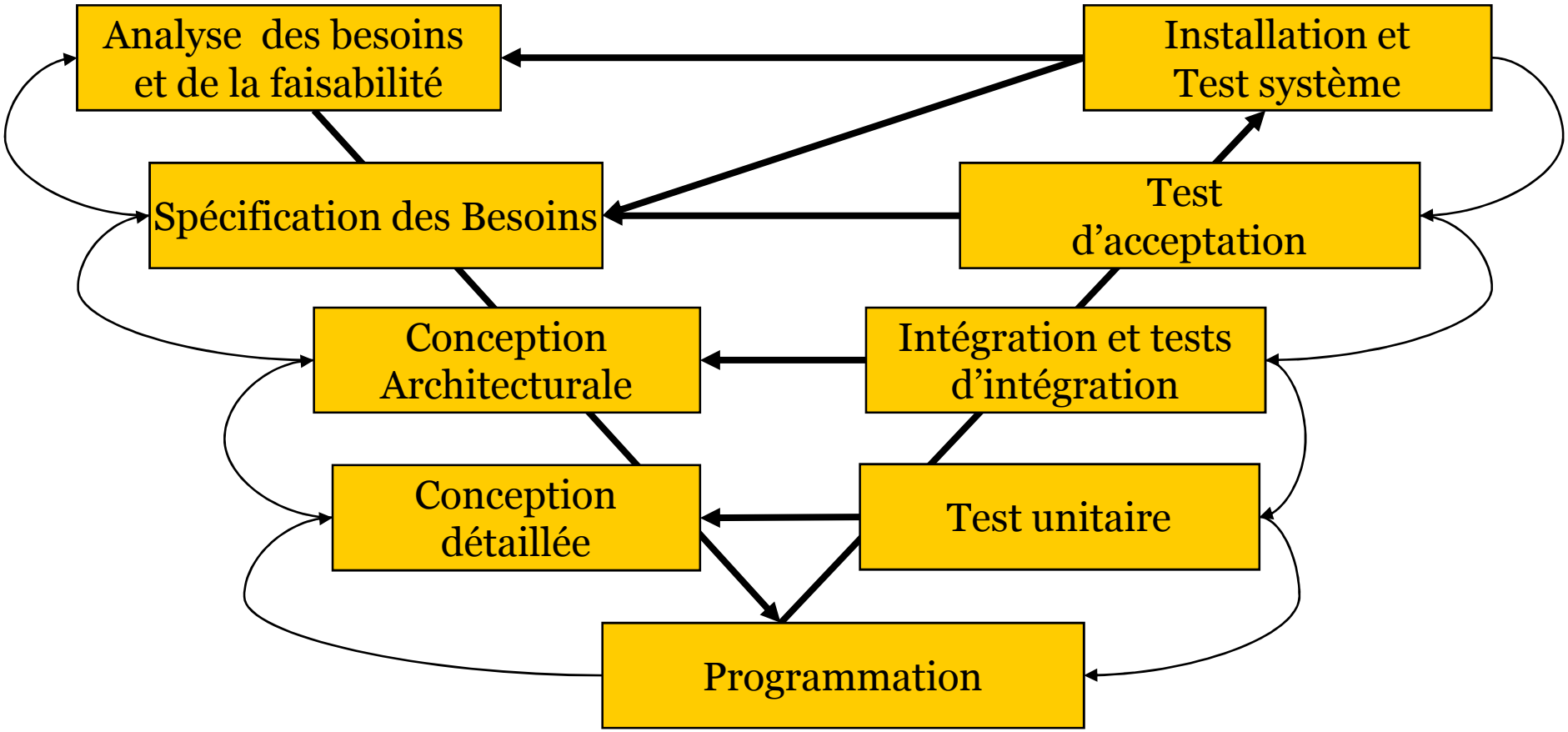


Modèle en V

- Conception architecturale = conception du produit : l'ensemble des modules constituant le logiciel et leurs interfaces
- Conception détaillée : conception de chaque module séparé
- Contrairement au modèle en cascade, ce modèle fait apparaître le fait que le début du processus de développement conditionne ses dernières étapes.
- Toute description d'un composant est accompagnée de tests qui permettront de s'assurer qu'il correspond à sa description
- Inconvénient majeur : il faut attendre la fin du projet pour avoir le retour des utilisateurs sur le système réellement implémenté.

Modèle en V

- ***Schématiquement :***



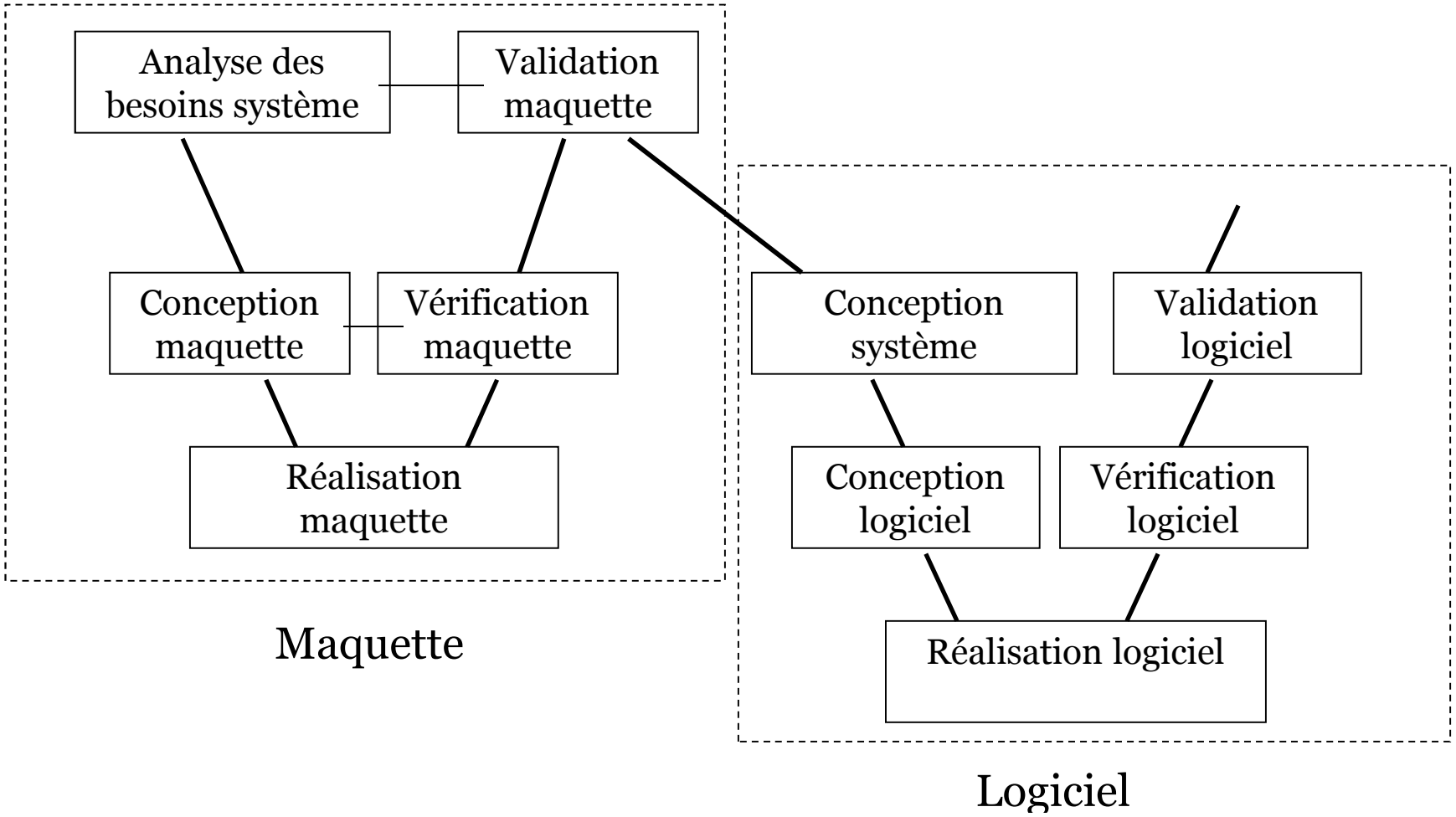
Modèle en W

- Dans le processus dit en W, des activités de maquettage permettent de s'assurer que les besoins système sont correctement satisfaits, ainsi que les besoins logiciel

- Un nombre limité d'itérations permet de gagner beaucoup de temps sur les étapes en aval et de limiter les mauvaises surprises au moment de la validation finale, alors que le budget du projet a été consommé

Modèle en W

- ***Schématiquement :***

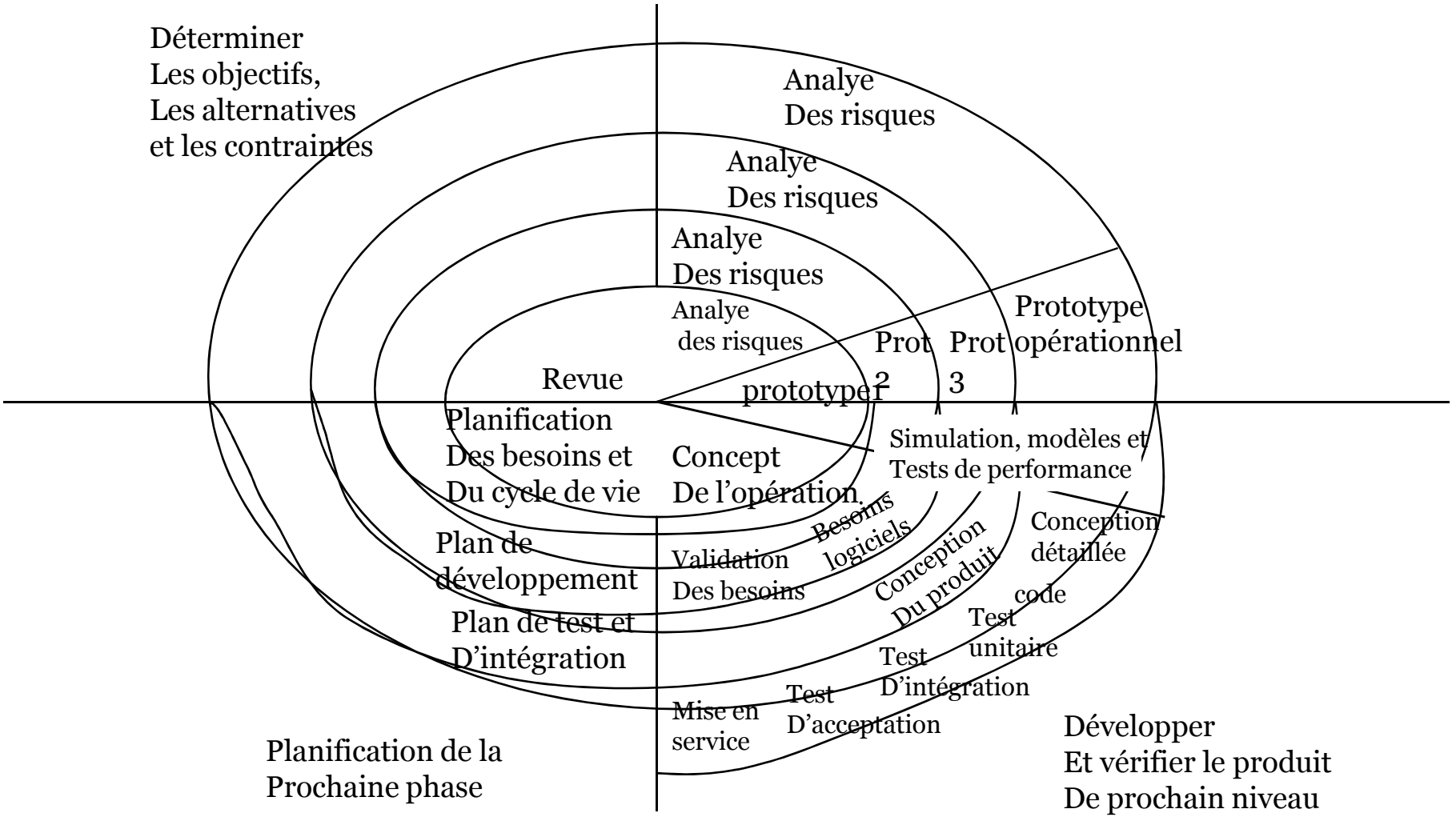


Modèle en spirale

- Proposé par B. Boehm en 1988, ce modèle est beaucoup plus général que le modèle en W.
- Il met l'accent sur l'activité **d'analyse des risques** : chaque cycle de la spirale se déroule en quatre phases :

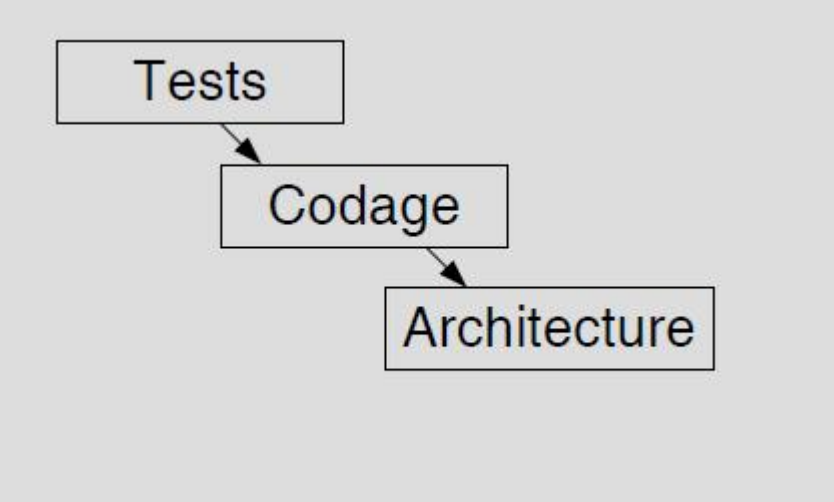
  - *détermination, à partir des résultats des cycles précédents ou de l'analyse préliminaire des besoins, des objectifs du cycle, des alternatives pour les atteindre et des contraintes ;*
  - *analyse des risques, évaluation des alternatives et, éventuellement maquettage ;*
  - *développement et vérification de la solution retenue, un modèle « classique » (cascade ou en V) peut être utilisé ici ;*
  - *revue des résultats et planification du cycle suivant.*

Modèle en spirale



Développement Agile

- Création des tests selon les spécifications
- Codage en conformité aux tests
- Emergence de l'architecture
  - À partir du code



Méthodes de conception

Méthodes de conception

- Méthodes de conception
  - Méthodes formelles
    - Validation mathématique de la conception
    - Réseaux de Pétri
    - Automates à états finis
  - RUP – Rational Unified Process (UML)
  - Model Driven Software Development

Méthodes de conception

- Les approches formelles utilisent des outils mathématiques et des méthodes de preuve pour construire un **logiciel correct par construction dont la** vérification est automatisée ou assistée.
- Méthodes : méthode B, *Model-checking, Logique de Hoare . . .*
- Outils et notations : Coq, Z, VHDL, Atelier B, CASL, Why, Frama-C, . . .
- Ces méthodes sont utilisées pour développer des logiciels critiques.
- Elles correspondent au niveau le plus élevé de certification.

Méthodes de conception

- Les approches semi-formelles visent à introduire un langage normalisé pour décrire le logiciel et sa spécification.
- Cependant, la **sémantique du langage de spécification n'est pas formalisée.**
- Bien que ces approches précisent le discours du concepteur si on le compare à celui décrit à l'aide du langage naturel, elles contiennent certaines ambiguïtés et n'offrent aucune garantie sur la qualité des résultats.
- Méthodes : Rationale Unified Process, Merise, . . .
- Outils et notations : UML, AnalyseSI, . . .
- Ces méthodes sont utilisées aujourd'hui par l'industrie du logiciel.

Méthodes de conception

- Les approches empiriques mettent en avant un ensemble de “bonnes pratiques” qui ont fait leur preuve par l’expérience.
- Méthodes : relecture de code, *extreme programming, programmation* défensive, . . .
- Outils : gestionnaire de versions, outil de documentation automatique, . . .

La gestion de projet

Quel est le rôle d'un chef de projet ?

- Les activités de gestion d'un projet informatique sont très similaires à celles des autres domaines :
  - Écriture de proposition de projet
  - Planification du projet
  - Évaluation des coûts
  - Surveillance du projet et écriture de rapport d'étapes
  - Sélection du personnel et évaluation
  - Écriture de rapport et de présentation

Écrire une proposition de projet

- À partir d'un appel d'offre, un chef de projet doit écrire une proposition de projet décrivant les objectifs du projet (en général, ses délivrables) et les grandes lignes de sa réalisation.
- Une proposition doit aussi contenir une évaluation des risques et des coûts.
- La plupart du temps, cette proposition doit servir d'argumentaire pour justifier la mise en route du projet.

→ *C'est une activité qui requiert une importante expérience et compréhension* du domaine d'activité. Le chef de projet engage sa responsabilité.

Planifier un projet

- Le chef de projet doit établir un jalonnement, c'est-à-dire une répartition des activités dans le temps en fonction de leurs dépendances et des ressources disponibles et d'une évaluation des risques liés à leur réalisation.

→ *Il s'agit d'un travail d'ordonnancement qui nécessite encore une connaissance* très précise du domaine, des équipes de développement, etc . . .

Veiller sur un projet

- De façon continue, le chef de projet s'assure du progrès des tâches et du respect des délais.

- En cas de retard, il doit réévaluer la planification et éventuellement renégocier les ressources et les contraintes du projet.

→ *La visibilité de la progression des activités est ici essentielle. Un chef de* projet doit donc savoir se doter d'indicateurs révélateurs sur l'état du développement.

Sélectionner le personnel

- En cohérence avec la politique de gestion du personnel (projet de carrière, formation continue, sous-traitement, . . .), le chef de projet doit affecter des activités et des rôles aux différentes personnes impliquées dans le projet.

→ *Des qualités relationnelles semblent donc utiles. . .*

Écrire un rapport

- Le chef de projet doit pouvoir communiquer une vue synthétique du projet à différents publics (autres chefs de projet, clients, responsables, etc . . .).

Bibliographie

- *Software Engineering – 8ème édition*

Sommerville – Addison Wesley

- *Génie logiciel*

J. Longchamp – Cours de CNAM