Analyse et Conception de Logiciels

Analyse et Conception de Logiciels

Horatiu CIRSTEA

Sujets abordés

- développement logiciel : les concepts, les méthodes
- les activités dans le développement logiciel
- modélisation avec UML
- techniques de test

http://arche.univ-lorraine.fr/course/view.php?id=312

Bibliographie

- Software Engineering, 9th ed. - Ian Sommerville, Pearson Education, 2011.
- Object-Oriented Software Engineering: Practical Software Development using UML and Java - Timothy C. Lethbridge, Robert Laganière, McGraw Hill, 2004.
- UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed.- Martin Fowler, Addison-Wesley, 2003.
- UML 2 - de l'apprentissage à la pratique - Laurent Audibert, Ellipses, 2009
- plus de références sur la page du cours

Exemples cours

Inspirés en partie de

- Software Engineering, 9th ed. - Ian Sommerville
- Object-Oriented Software Engineering: Practical Software Development using UML and Java - Timothy C. Lethbridge, Robert Laganière
- UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. - Martin Fowler
- UML 2 - de l'apprentissage à la pratique - Laurent Audibert, Ellipses, 2009
- UML 2 par la pratique - Pascal Roques, Eyrolles, 2009

- I -

Introduction au processus de développement logiciel

UNIVERSITÉ DE LORRAINE

Analyse et Conception de Logiciels

Horatiu CIRSTEA

Vue générale

Contexte

- Les économies de tous les pays développés dépendent du développement logiciel
- De plus en plus d'infrastructures et de systèmes sont contrôlés par des logiciels
  - administration
  - industrie
  - finances
  - loisirs
  - ...

Coûts

- Les coûts liés au logiciel (software) sont souvent plus importants que les couts des systèmes informatiques (hardware)
- Les logiciels d'un PC coûtent souvent plus cher que la machine
- Les coûts de maintenance d'un logiciel sont en général plus importants que les coûts de développement

Comment produire *efficacement* de *bons* logiciels?

Le génie logiciel (software engineering)

Software engineering is concerned with cost-effective software development.

I.Sommerville

Le génie logiciel (software engineering)

Le processus visant la résolution de problèmes posés par un client par le développement systématique et l'évolution de systèmes logiciels de grande taille et de haute qualité en respectant les contraintes de coûts, de temps, et autres.

T. Lethbridge, R. Laganière

Logiciels

Produits génériques

- PC software (bureautique, programmes graphiques, CAD, ...)
- *spécification donnée par le développeur*

Produits spécifiques

- systèmes embarqués, contrôle trafic aérien,...
- *spécification donnée par le client*

F.A.Q.

- Qu'est-ce qu'un logiciel (software) ?
  - Un programme informatique et la documentation associée.
- Quels sont les attributs d'un bon logiciel ?
  - Un bon logiciel doit fournir à l'utilisateur les fonctionnalités et performances demandées et doit être maintainable, fiable et utilisable.
- Qu'est-ce que l'ingénierie logicielle (software engineering) ?
  - Une discipline concernée par tous les aspects de la production de logiciels.

F.A.Q.

- Quelles sont les activités principales dans l'ingénierie logicielle ?
  - spécification, développement, validation et évolution (des logiciels)
- Quels sont les principaux défis du génie logiciel?
  - Faire face à la diversité croissante, les délais de livraison réduits et le développement de logiciels de confiance.

F.A.Q.

- Quels sont les coûts d'ingénierie logicielle ?
  - Environ 60% des coûts sont les coûts de développement, 40% des coûts liés aux test.
  - Pour les logiciels spécifiques, les coûts d'évolution dépassent souvent les coûts de développement.
- Quelles sont les meilleures techniques et méthodes de génie logiciel ?
  - différentes techniques sont appropriés pour différents types de système

Attributs des logiciels

- **Efficacité** : Aucun gaspillage de ressources (mémoire, temps de calcul, …).
- **Fiabilité et sécurité** : Les pannes ne doivent pas engendrer des dommages au système. Les intrus ne doivent pas pouvoir accéder et endommager le système.
- **Maintenabilité** : Le logiciel doit pouvoir évoluer pour correspondre aux demandes de changement.
- **Acceptabilité** : Apprentissage aisé, facilité d'utilisation, compatibilité avec les autres systèmes utilisés par les clients.

Les parties impliquées

- **Utilisateurs**

  Ceux qui se servent du logiciel

- **Clients**

  Ceux qui paient pour le logiciel

- **Développeurs**

  Ceux qui conçoivent le logiciel

- **Managers**

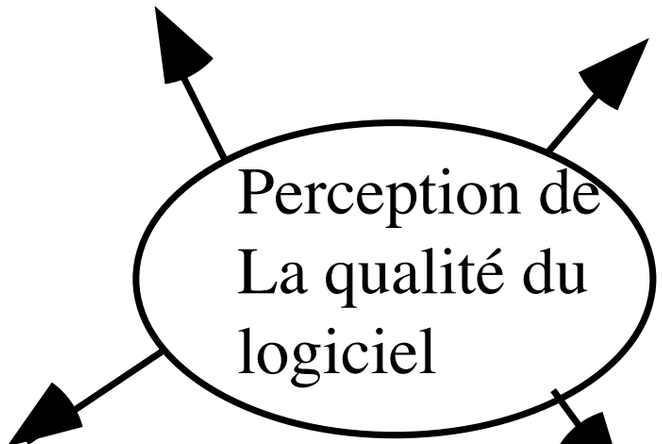
  Ceux qui supervisent la production du logiciel

Tous ces rôles peuvent être remplis par la même personne

La qualité du logiciel...

Client:
Résoud le problème
à un coût acceptable

Utilisateur:
Facile à apprendre,
utile et efficace



Développeur:
Facile à concevoir,
à maintenir, à réutiliser

Gestionnaire:
Se vend bien,
satisfait les clients,
peu coûteux à développer

Le génie logiciel

Concerne *tous* les aspects du développement logiciel de sa spécification jusqu'à la gestion de ses évolutions

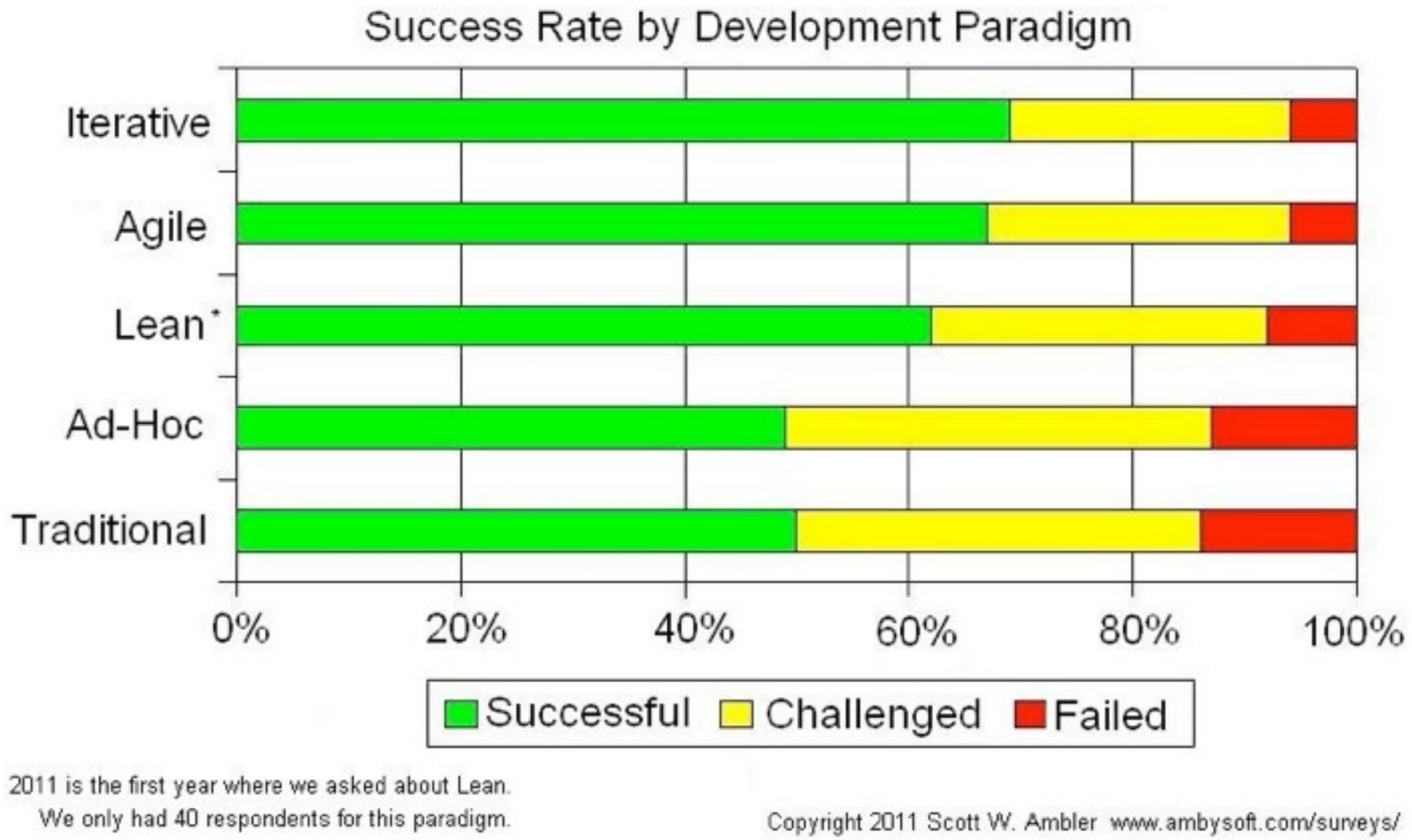
- utilise des méthodes et théories appropriées pour résoudre des problèmes sous des contraintes organisationnelles et financières
- non seulement un processus de développement

Pourquoi c'est important?

"This year's results show a marked decrease in project success rates, with 32% of all projects succeeding which are delivered on time, on budget, with required features and functions" says Jim Johnson, chairman of The Standish Group, "44% were challenged which are late, over budget, and/or with less than the required features and functions and 24% failed which are cancelled prior to completion or delivered and never used."

Boston, Massachusetts, April 23, 2009
New Standish Group report "CHAOS Summary 2009"

Pourquoi c'est important?



Ambysoft survey, 2011
http://www.ambysoft.com/surveys

Pourquoi c'est important?

- produire *efficacement* des logiciels *robustes*
- à long terme il est *moins couteux* d'utiliser des techniques et méthodes d'ingénierie logicielle

Activités

- **Spécification** : définir les fonctionnalités et les contraintes de fonctionnement.
- **Développement** : conception et implantation.
- **Validation** : vérification par rapport aux demandes du client.
- **Evolution** : modification par rapport aux demandes du client.

Facteurs importants

- **Hétérogénéité** : systèmes distribués composés de différents types d'ordinateurs et équipements
- **Changements économiques et sociaux** : besoin d'adapter rapidement des logiciels au changement
- **Sécurité** : confiance dans les logiciels développés

Types d'applications

- Stand-alone (non connectées à un réseau)
- Transactionnelles et interactives (e-commerce)
- Systèmes embarqués
- Batch processing
- Loisirs
- Modélisation et simulation
- Traitement collections de données
- Systèmes de systèmes

Principes fondamentaux

- Utiliser un processus de développement bien maîtrisé.
- Garantir la fiabilité et la performance.
- Comprendre et gérer le cahier de charges et la spécification du logiciel.
- Réutiliser des logiciels existants si possible (plutôt que d'en développer des nouveaux).

A retenir

- Le génie logiciel concerne tous les aspects liés à la production de logiciel
- Principaux attributs d'un logiciel: maintenabilité, fiabilité et sécurité, efficacité, acceptabilité
- Activités communes: spécification, développement, validation, évolution
- Les principes fondamentaux de l'ingénierie logicielle s'appliquent à tous les types de développement
- Les méthodes et outils d'ingénierie doivent être choisis en fonction des systèmes développés

Le développement logiciel

Modèles du processus de développement logiciel

- **Le cycle (processus) de développement** : un ensemble structuré d'activités nécessaires au développement d'un logiciel

  ➥ Différents processus de développement mais comportant des activités communes

- **Modèle de processus de développement** : une représentation abstraite d'un processus (par rapport à une certaine perspective).

Activités

- **Spécification**
  - définir le comportement du système;
- **Conception et implémentation**
  - définir l'organisation et implanter le système;
- **Validation**
  - vérifier que ça correspond à la demande;
- **Evolution**
  - modifier le système en fonction des demandes.

Cycle de développement

- Activités (avec sous-activités) et leur ordonnancement
- Produits (résultant des activités)
  - **ex:** modèle de l'architecture
- Rôles (et responsabilités)
  - **ex:** manager projet, programmeur, ...
- Pre- et post-conditions
  - **ex:** tous les besoins validés avant la conception, modèles UML examinés après la conception, ...

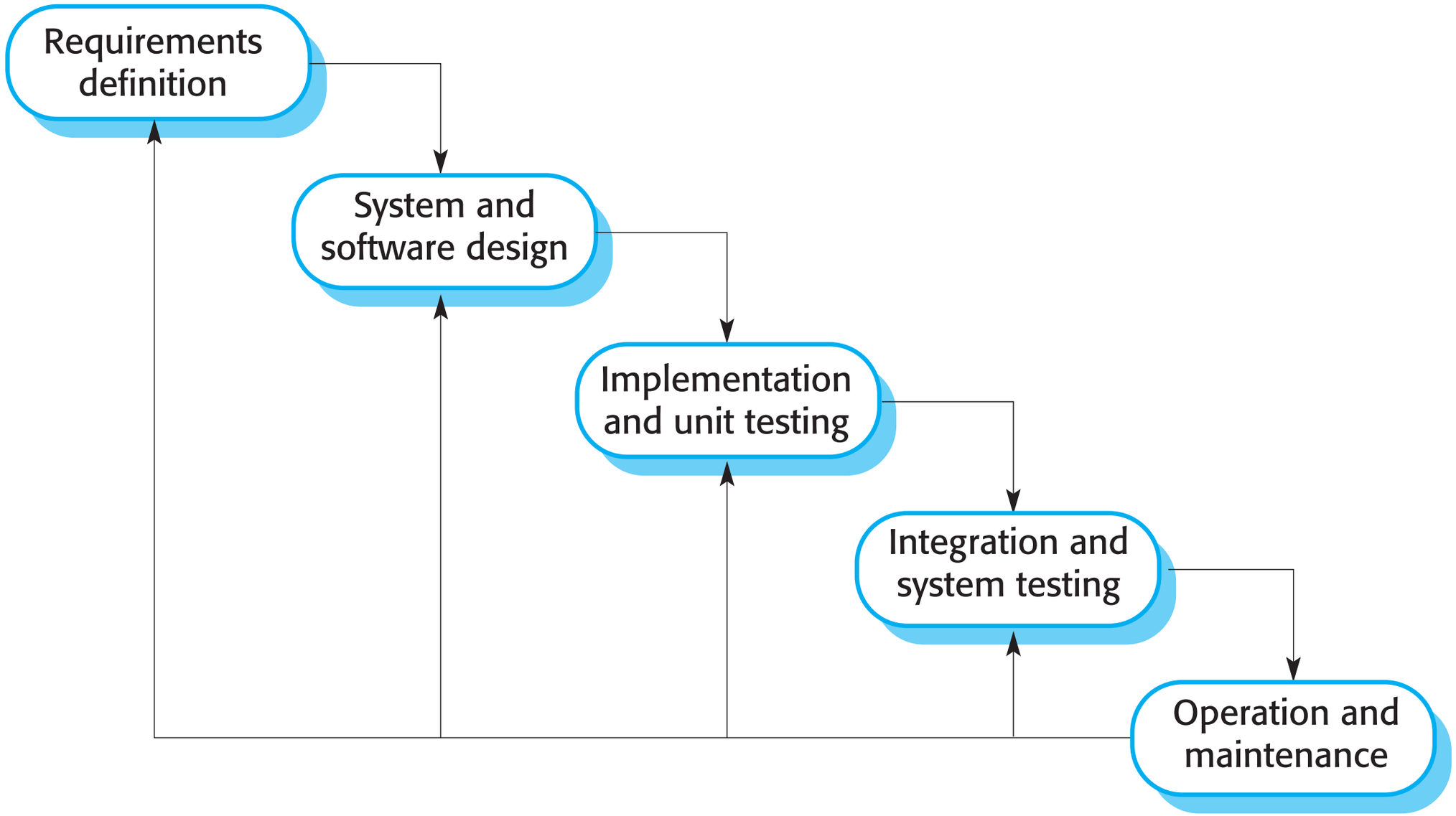
Plan-driven vs. Agile

- **Plan-driven** : les activités sont planifiées en avance et le progrès est évalué par rapport au plan.
- **Agile** : le planning est incrémental et plus facilement adaptable aux changements (demandés par les clients).

Modèles

- **Cascade** : phases distinctes de spécification et développement
- **Développement incrémental** : la spécification, le développement et la validation se superposent.
- **Modèle orienté réutilisation** : assemblage à partir de composants existants.

Cascade (Waterfall)



Waterfall

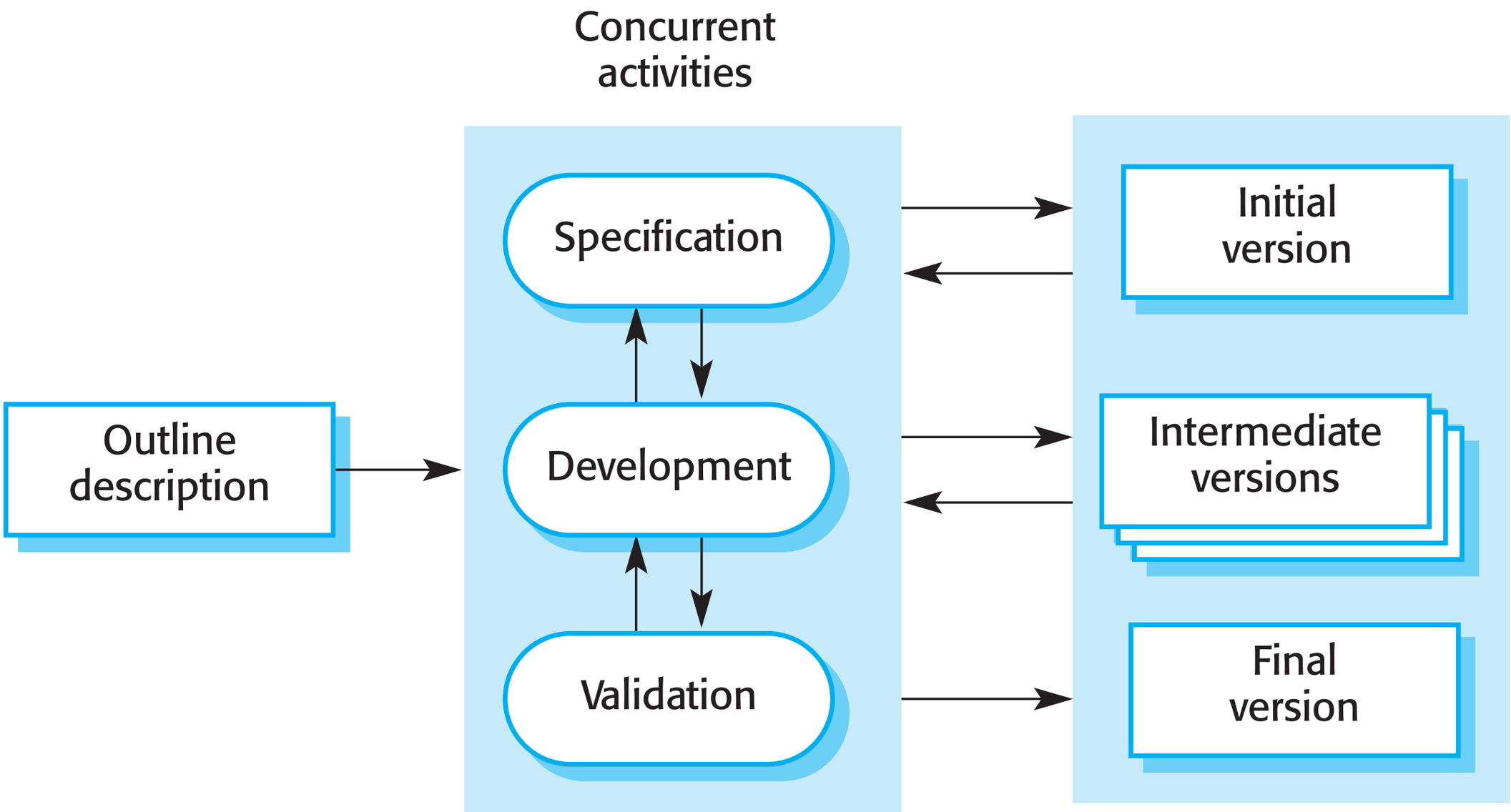
INCONVENIENTS ?

- modifications difficiles à intégrer après le début du processus (chaque phase doit être terminée avant de passer à la suivante)
- difficile à dire si le projet est en bonne voie (décalages dans le planning difficiles à anticiper)

Waterfall

- Découpage strict des phases
  - adapté quand les besoins sont claires et relativement stables
- Utilisé pour coordonner le développement
  - adapté quand le système est développé sur plusieurs sites

Développement incrémental



Développement incrémental

AVANTAGES ?

- Le coût pour l'intégration des modifications (demandées par le client) est réduit
  - moins d'analyse et documentation à refaire
- Plus facile à obtenir du feedback client par rapport au travail déjà réalisé.
  - les clients peuvent commenter les demos
- Livraison et déploiement chez le client plus rapides
  - les clients peuvent bénéficier du produit plus rapidement

Développement incrémental

INCONVENIENTS ?

- Le processus n'est pas visible (pour les managers)
  - la production de documents pour le nombreuses versions (delivrables) n'est pas rentable
- La structure du système a tendance à se dégrader avec l'ajout de nouveaux incréments
  - l'intégration de nouvelles fonctionnalités devient de plus en plus difficile

Développement incrémental

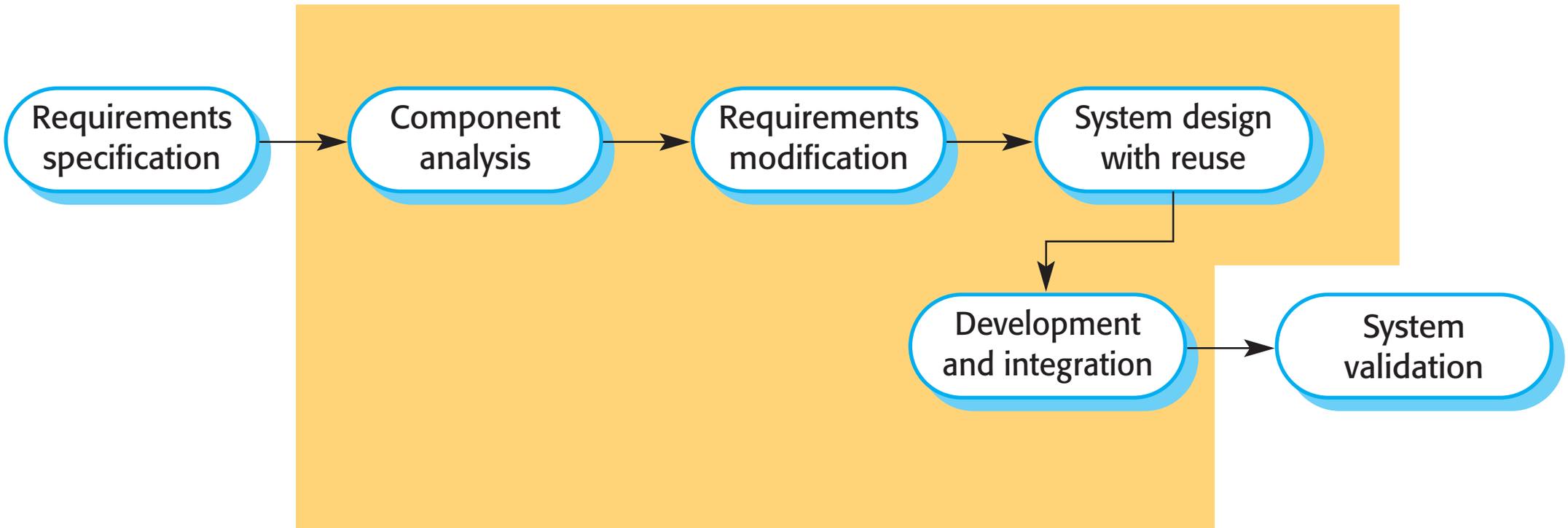
PROBLEMES ?

- Problèmes liés au management
- Problèmes liés aux contrats
- Problèmes liés à la validation
- Problèmes de maintenance
- Problèmes liés aux technologies

Modèle orienté réutilisation

- Basé sur le développement de systèmes à partir de composants existants - COTS (Commercial-off-the-shelf).
- Approche standard pour beaucoup de systèmes d'information

Modèle orienté réutilisation



Modèle orienté réutilisation

- Services Web (invocables à distance)
- Collections d'objets (package)
- Software stand-alone (COTS) configurable pour un environnent particulier

Les activités du processus de développement logiciel

Activités

- Spécification (analyse des besoins)
- Conception et implémentation
- Validation
- Evolution

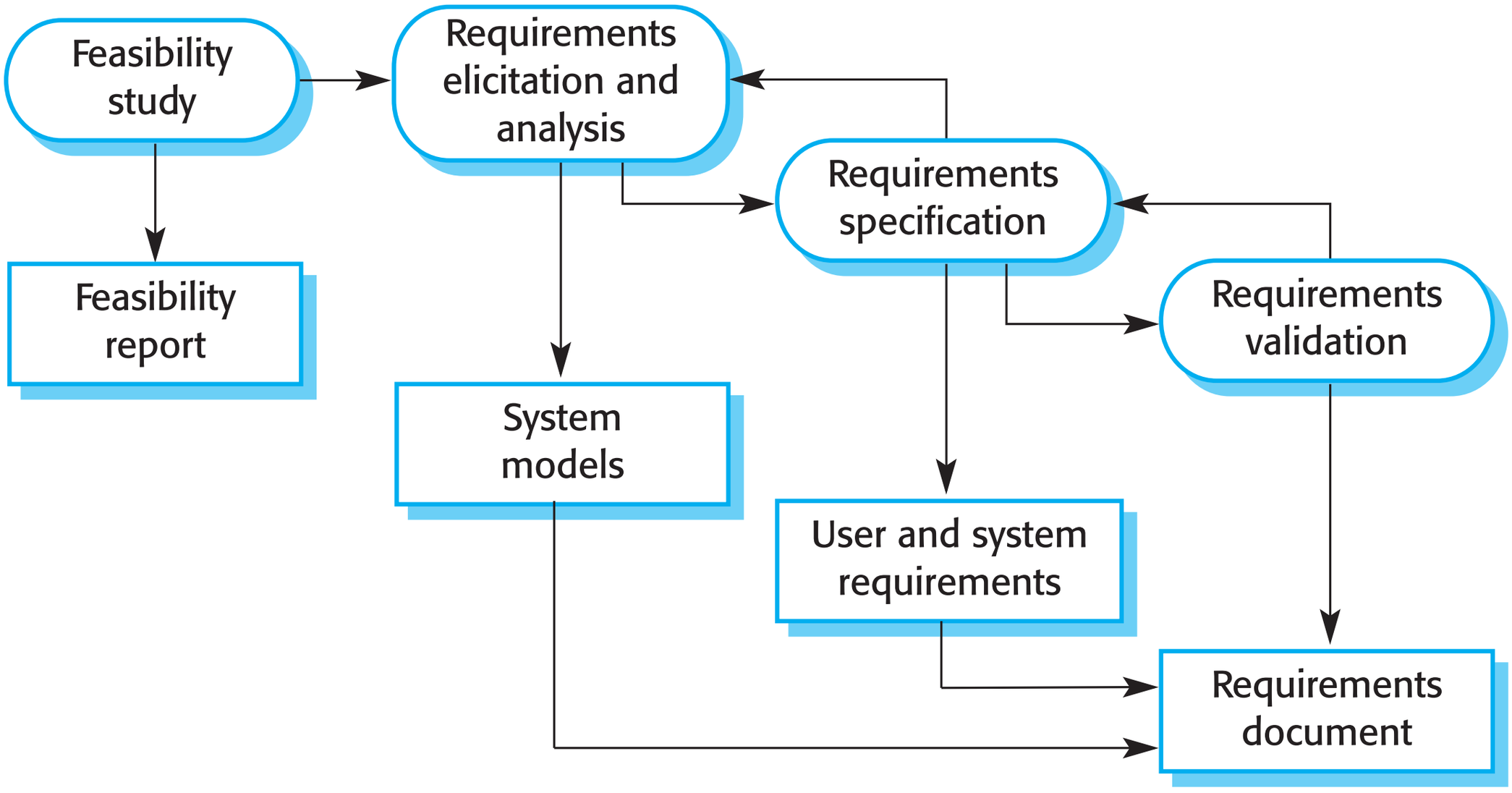
➥ Organisées en fonction du modèle

- Waterfall ➠ séquentielles
- Incrémental ➠ entrelacées

Analyse des besoins

- Etablir les services demandés et les contraintes de fonctionnement et de développement
- Activités de l'analyse des besoins
  - ✦ étude de faisabilité
  - ✦ découverte et analyse des besoins
  - ✦ spécification des besoins
  - ✦ validation des besoins

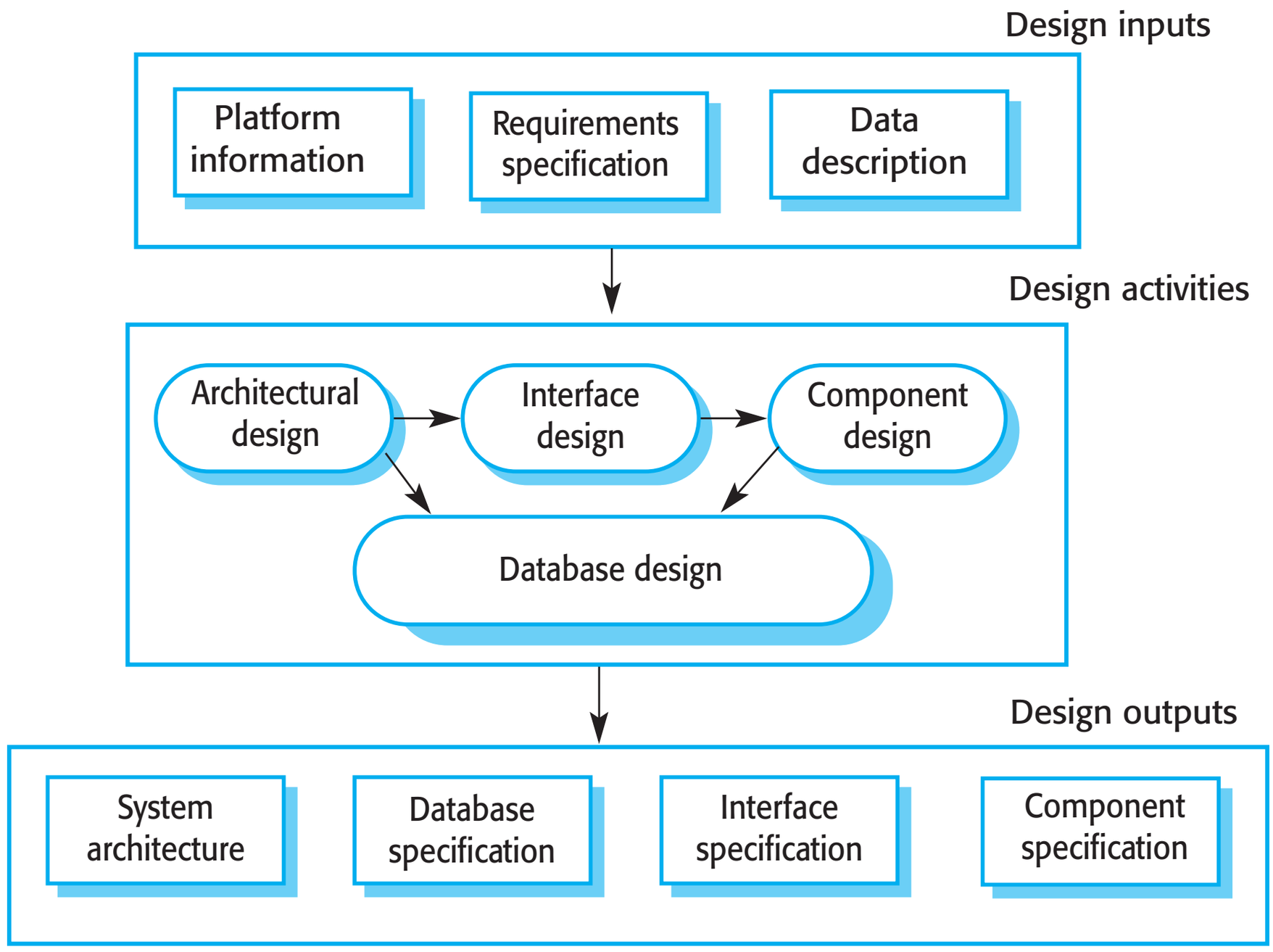
Analyse des besoins



Conception et implémentation

- Transformation de la spécification en un système exécutable
- Conception
  - définir une structure logicielle qui correspond à la spécification
- Implémentation
  - traduire la structure en un programme

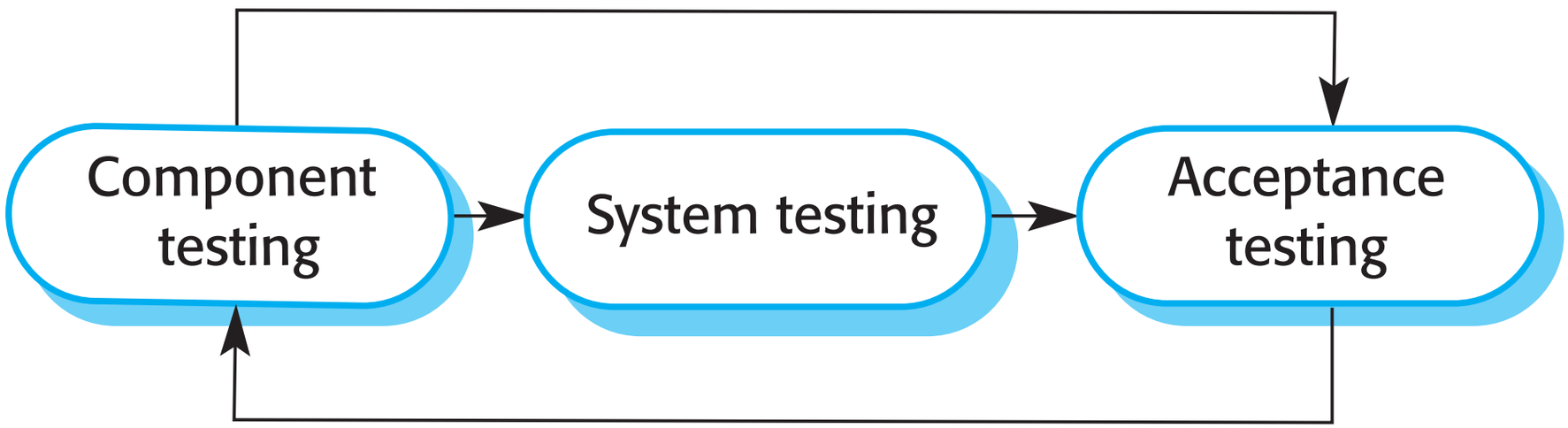
Conception et implémentation



Validation et vérification

- Montrer que le système est **conforme à la spécification** et **satisfait les besoins** du client
- **Test -** exécution avec des cas de test extraits de la spécification (des données réelles traitées)
- **Checking -** inspection et review

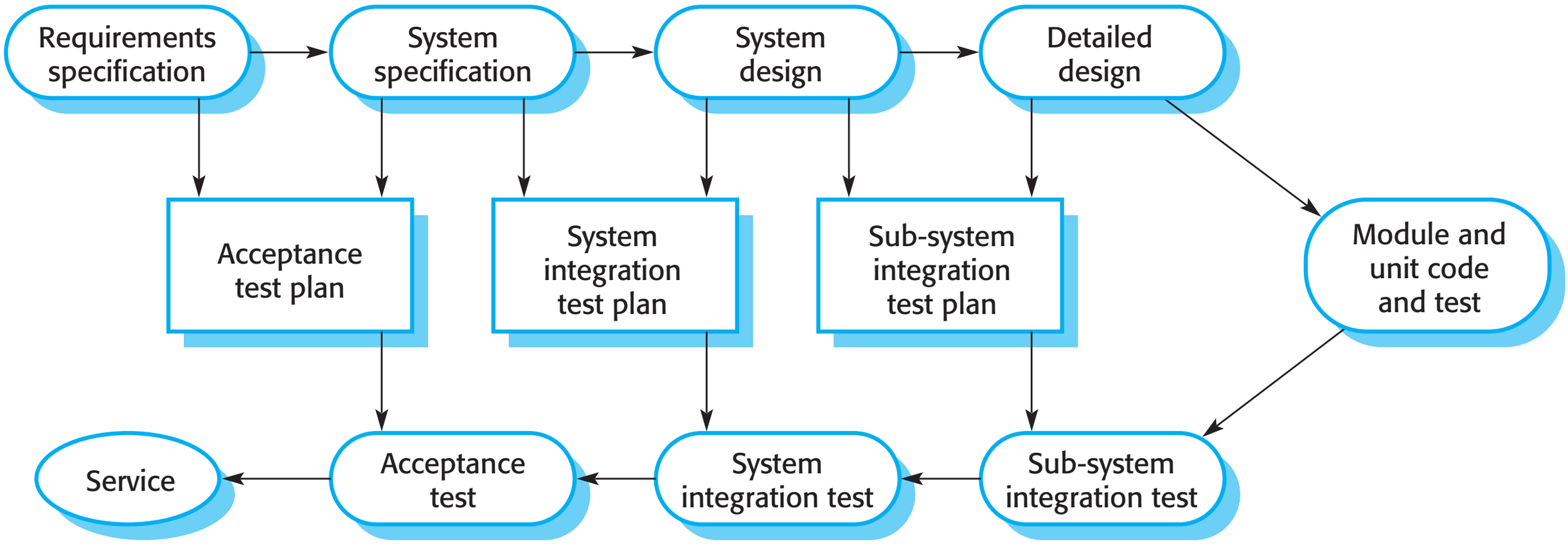
Testing



Testing

- **Component testing**
  - tests indépendants des composants
- **System testing**
  - tests du système (et des sous-systèmes)
- **Acceptance testing**
  - tests avec données client

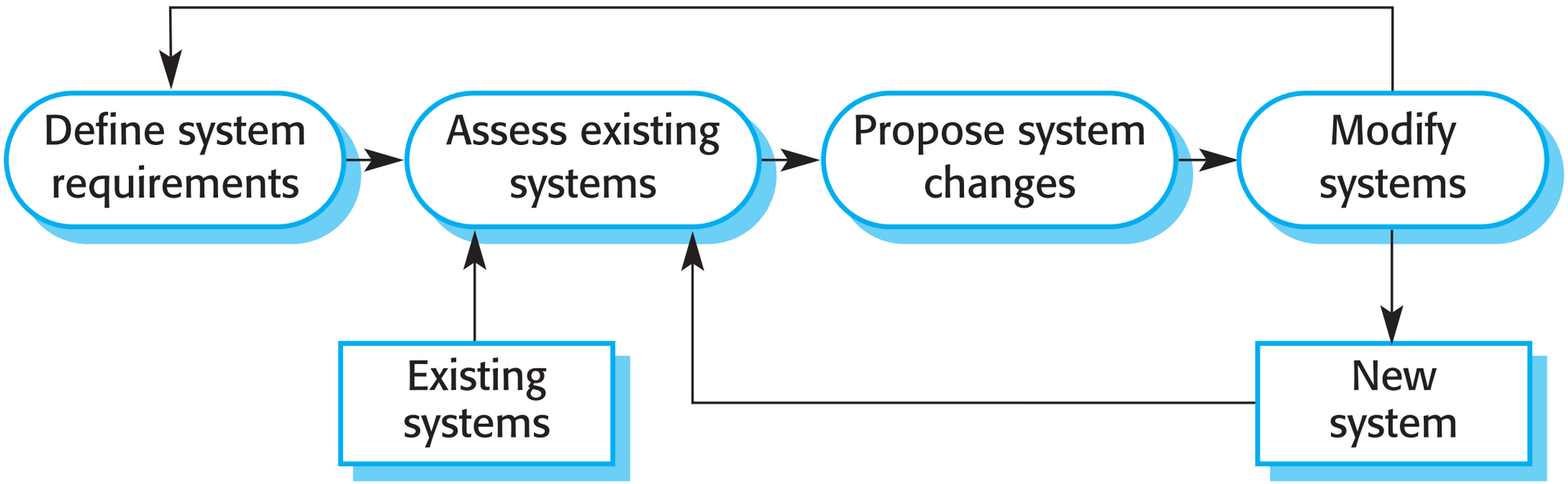
Testing dans un modèle en V



Evolution

- Le logiciel est naturellement flexible
- Les besoins du domaine peuvent changer et les logiciels correspondants doivent évoluer en conséquence
- La limite entre développement et maintenance est de plus en plus difficile à déterminer

Evolution


Define system requirements
Assess existing systems
Propose system changes
Modify systems
Existing systems
New system

Questions ?

Quiz

- Citer trois modèles de processus de développement
  - waterfall, incrémental, orienté réutilisation
- Quelles sont les activités communes à tous les processus de développement?
  - spécification , design et implémentation, validation, évolution
- Pourquoi les itérations sont limitées dans l'utilisation du modèle cascade?
  - les itérations (production et validation documents) peuvent être couteuses

Quiz

- Quelles sont les avantages du développement incrémental ?
  - coût d'intégration des modifications réduit
  - feedback client facilité
  - livraison et déploiement chez le client plus rapides

- Quelles sont les avantages de la livraison incrémentale ?
  - fonctionnalités disponibles plus tôt
  - incréments utiliser comme prototype pour la découverte des besoins
  - fonctionnalités prioritaires sont testées davantage
  - risque réduit d'échec global du projet

Quiz

- Principales activités dans l'analyse des besoins ?
  - étude de faisabilité
  - découverte et analyse des besoins
  - spécification des besoins
  - validation des besoins

A retenir

- Le cycle (processus) de développement est un ensemble d'activités nécessaires au développement d'un logiciel
- Les modèles de développement sont des représentations abstraites d'un processus : waterfall, incrémental, orienté réutilisation, ...

A retenir

- *Analyse des besoins* ➠ développement d'une spécification des besoins (cahier de charges)
- *Conception et implémentation* ➠ transformation du cahier de charges en un logiciel exécutable
- *Validation* ➠ vérification de la conformité avec la spécification et les besoins
- *Evolution* ➠ adapter le logiciel pour prendre en compte de nouveaux besoins

Les processus doivent prévoir des activités pour prendre en compte des changements.