Processus de logiciel

Objectifs

- Introduire des modèles de processus logiciel
- Proposer un modèle qui sera utilisé pour le projet de développement dans le cadre du cours

Le processus logiciel

- Un ensemble structuré d'activités pour le développement d'un système logiciel
  - Specification
  - Conception (Design)
  - Implémentation
  - Validation
  - Évolution
- Ces activités dépendent de l'organisation et du type de produit à construire
- Doit être explicitement défini s'il doit être géré

Processus d'ingénierie

- *Spécification* - Définir les besoins et contraintes du système
- *Conception* - Produire un modèle du système
- *Réalisation* - Construire du système
- *Test* - Vérifier l'adéquation entre la spécification et la réalisation
- *Installation* - Délivrer le système au client et s'assurer qu'il est opérationnel
- *Maintenance* - Réparer les fautes du systèmes au fur et à mesure qu'elles apparaissent

Caractéristiques d'un processus

- Sûre
  - Les erreur sont détectées avant la mise en service du produit
- Robuste
  - Les problèmes non prévus n'arrête pas la réalisation du produit
- Maintenable
  - Le processus évolue en fonction des besoins de changements d'organisation
- Efficace
  - Le temps de réalisation du produit est optimisé

Défis des processus logiciels

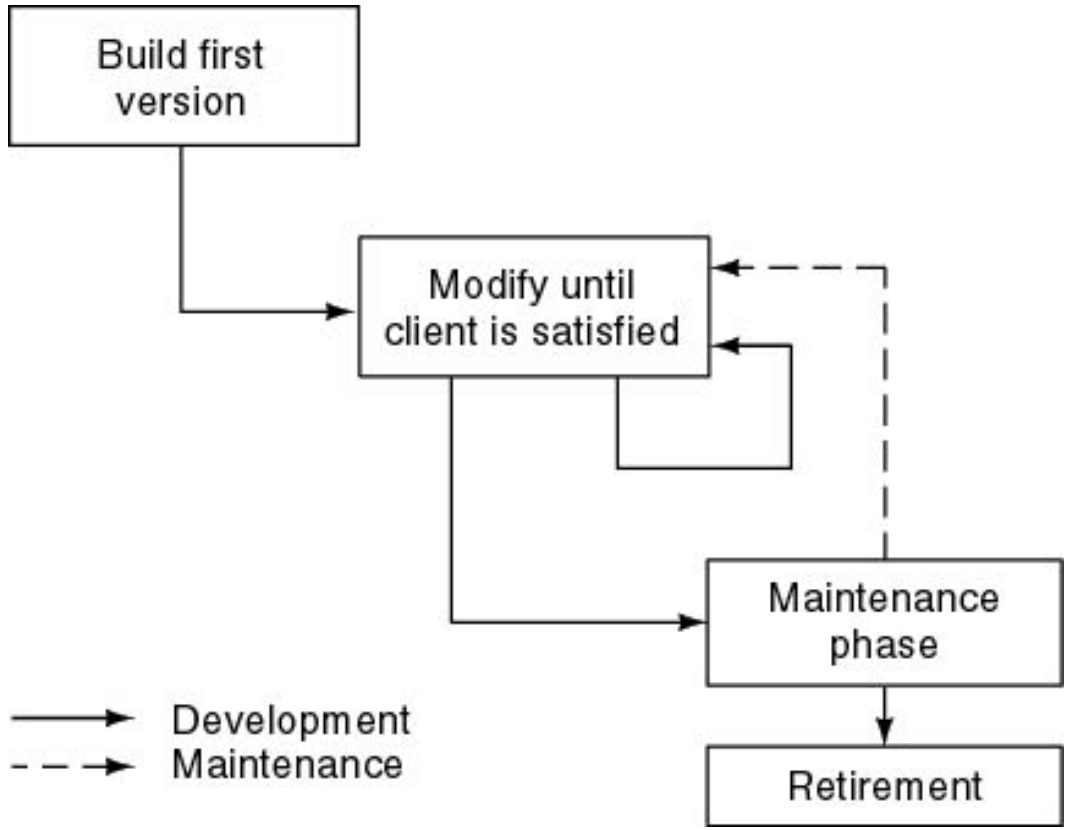
- Généralement, les spécifications sont incomplètes
- La frontière entre la spécification, la conception et la réalisation est floue
- Les tests ne sont pas réalisés dans l'environnement opérationnel du système
- Un logiciel ne connaît pas l'usure - maintenir ne veut pas dire remplacer un composant

Modèles génériques de processus

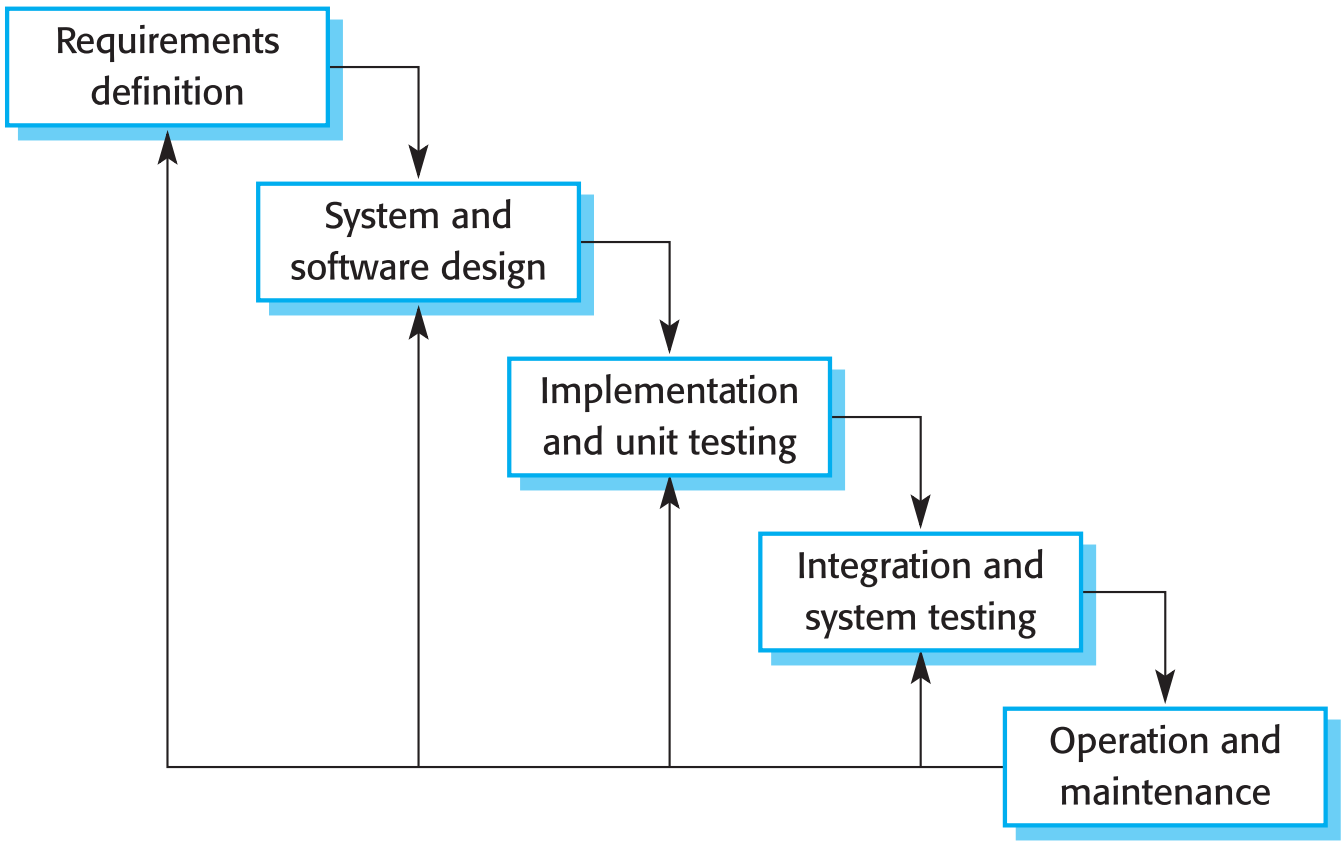
- Le Modèle en cascade
  - Série d'étapes clairement séparées allant de la spécification à la réalisation
- La programmation évolutive
  - Spécification et développement sont réalisés conjointement
- La transformation formelle
  - Un modèle de système est transformé formellement jusqu'à son implémentation
- Développement basé sur la réutilisation
  - Le système est assemblé à partir des composantes existants

Modèle « Build-and-Fixe »

- Problèmes
  - Pas de spécifications
  - Pas de design
- Totalement insatisfaisant
- Besoin de modèle de cycle de vie
  - Plan de dév.
  - Phases
  - Milestones



Modèle de la cascade



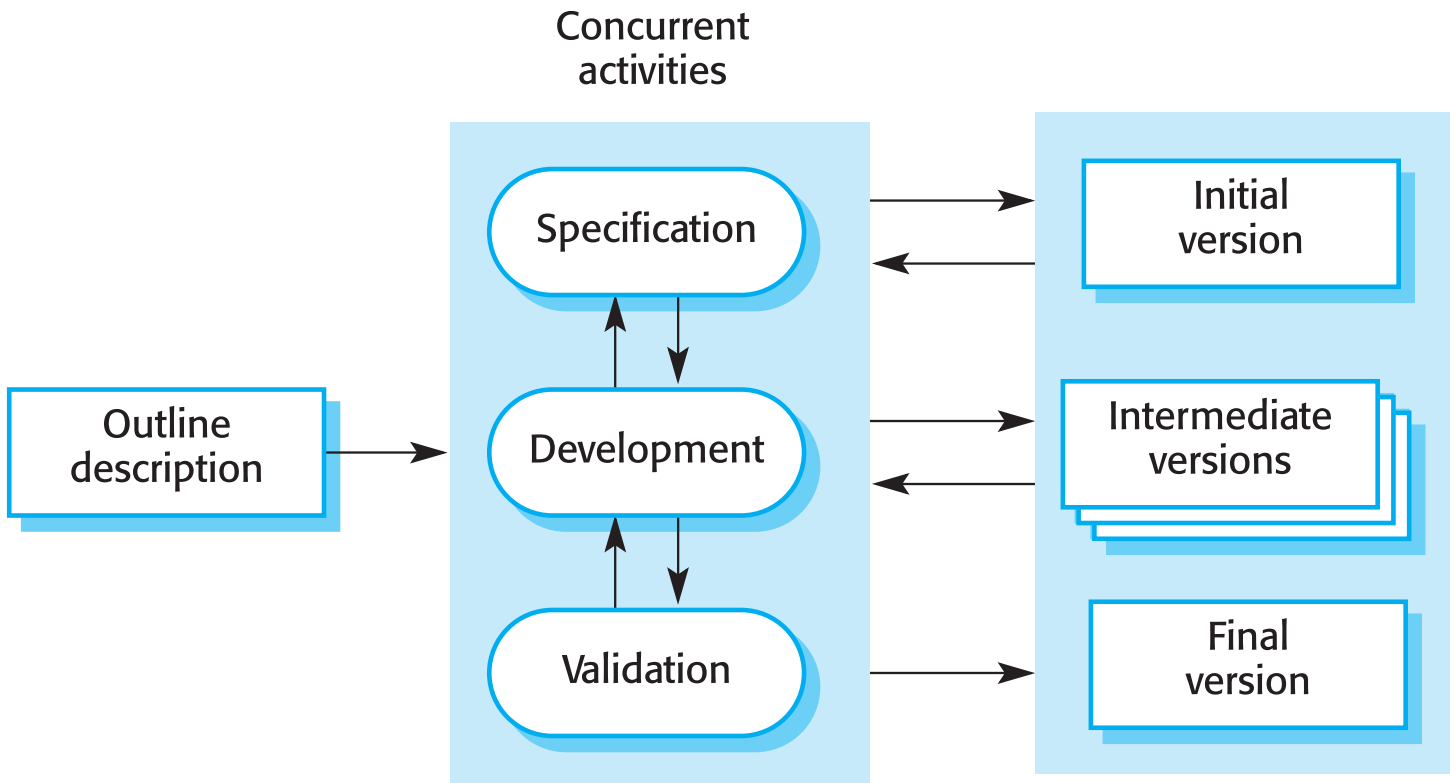
Phases du modèle en cascade

- Définition et analyse des besoins
- Conception système et logiciel
- Implémentation et tests unitaires
- Intégration et tests du système
- Mise en service et maintenance
- Le défaut du modèle en cascade est qu'il est difficile d'effectuer des modifications en cours de route

Problème du modèle en cascade

- Répartition inflexible du projet en des étapes distingues
- Le défaut du modèle en cascade est qu'il est difficile d'effectuer des modifications en cours de route
- Ce modèle est uniquement approprié lorsque les exigences de logiciel sont bien définies

Développement évolutif



Développement évolutif

- Développement exploratoire
  - L'objectif est de travailler avec les clients et d'élaborer un système final à partir des grandes lignes d'une spécification initiale. Doit commencer avec une bonne compréhension des besoins

- Prototype jetable
  - L'objectif est de comprendre les besoins du produit. Doit commencer avec une faible connaissance des besoins

Développement évolutif

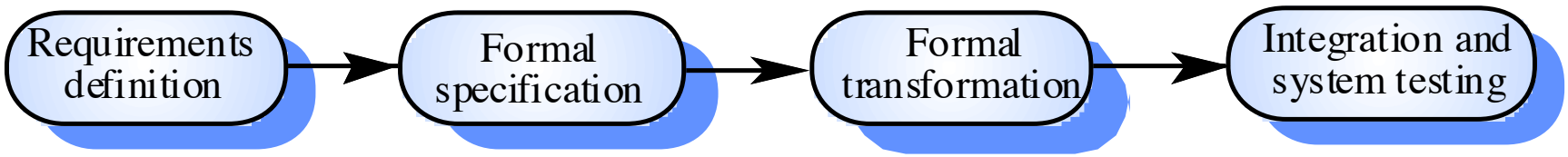
- Problèmes
  - Faible visibilité du processus
  - Le système est souvent mal structuré
  - Certains connaissances (e.g. dans les langages de prototypage) peuvent être nécessaires

- Domaine d'application
  - Pour des petits ou moyens systèmes interactifs
  - Pour une partie d'un grand système (e.g. interface utilisateur)
  - Pour système à vie courte

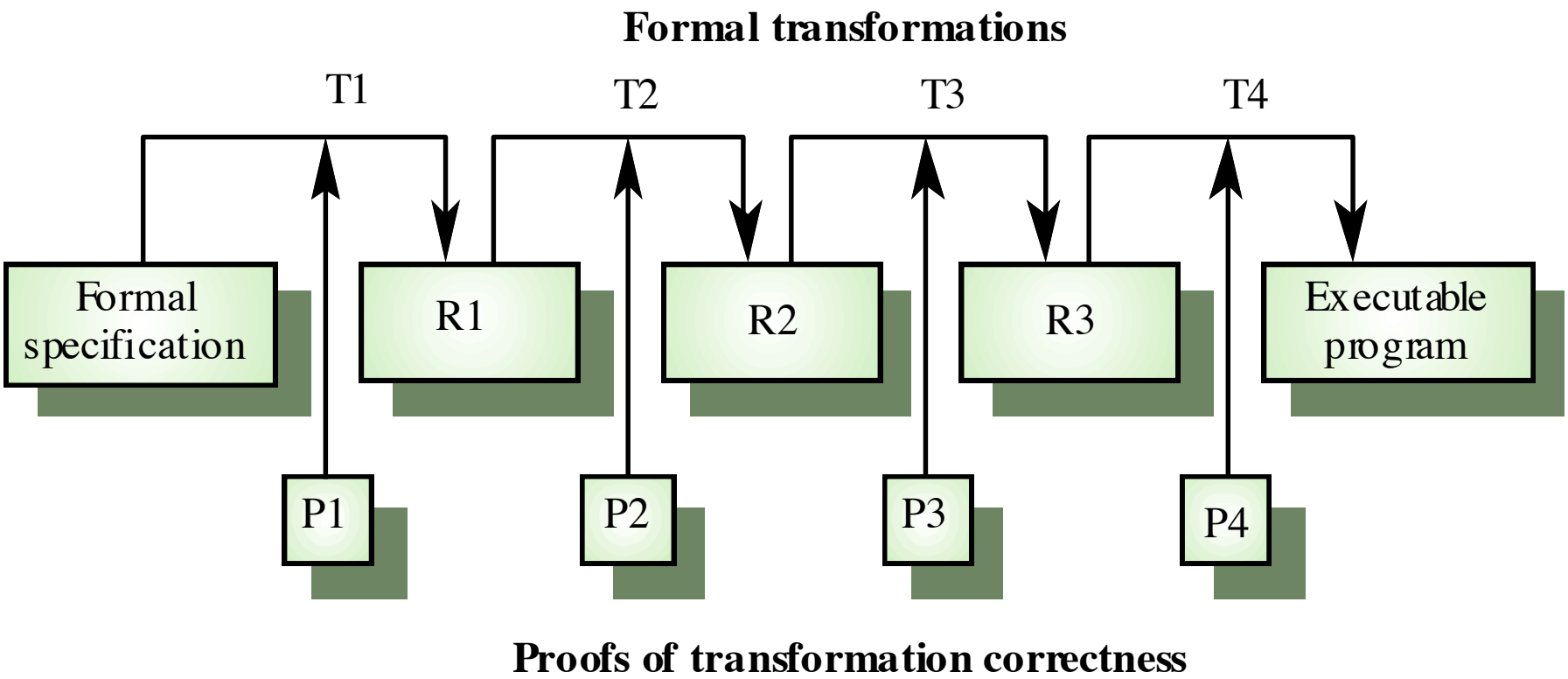
Développement formel

- Basé sur une transformation d'une spécification formelle (mathématique) à travers des différentes représentations jusqu'à un programme
- Les transformations sont 'correctness-preserving' donc il est évident de démontrer que le programme conforme à sa spécification
- Exemple: l 'approche 'Cleanroom'

Développement formel



Transformation formelle



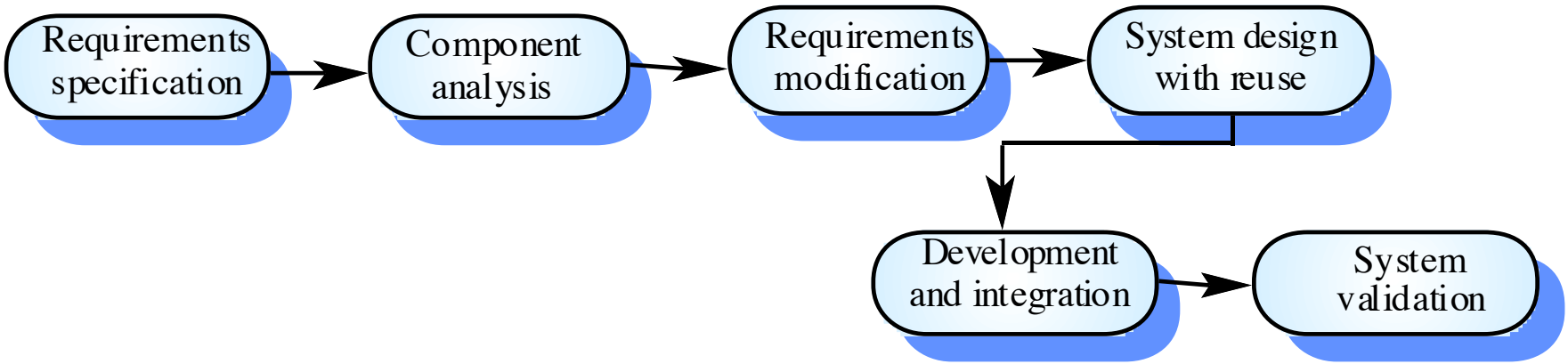
Développement formel

- Problèmes
  - Besoin des compétences et formations spécialisées pour appliquer la technique
  - Difficile à spécifier formellement certains aspects du système , par exemple l'interface d'usager
- Domaine d'application
  - Systèmes critiques, particulièrement des systèmes dont la fiabilité et sécurité doivent être prouvées avant que le système ne sera mis en opération

Développement basé sur la réutilisation

- Basé sur la réutilisation systématique, des systèmes sont intégrés à partir des composants existants ou COTS (Commercial-off-the-shelf)
- Étape du processus
  - Analyse des composants
  - Modification des exigences logicielles
  - Conception de système avec réutilisation
  - Développement et intégration
- Cette approche devient plus important mais encore des expériences limitées

Développement avec réutilisation



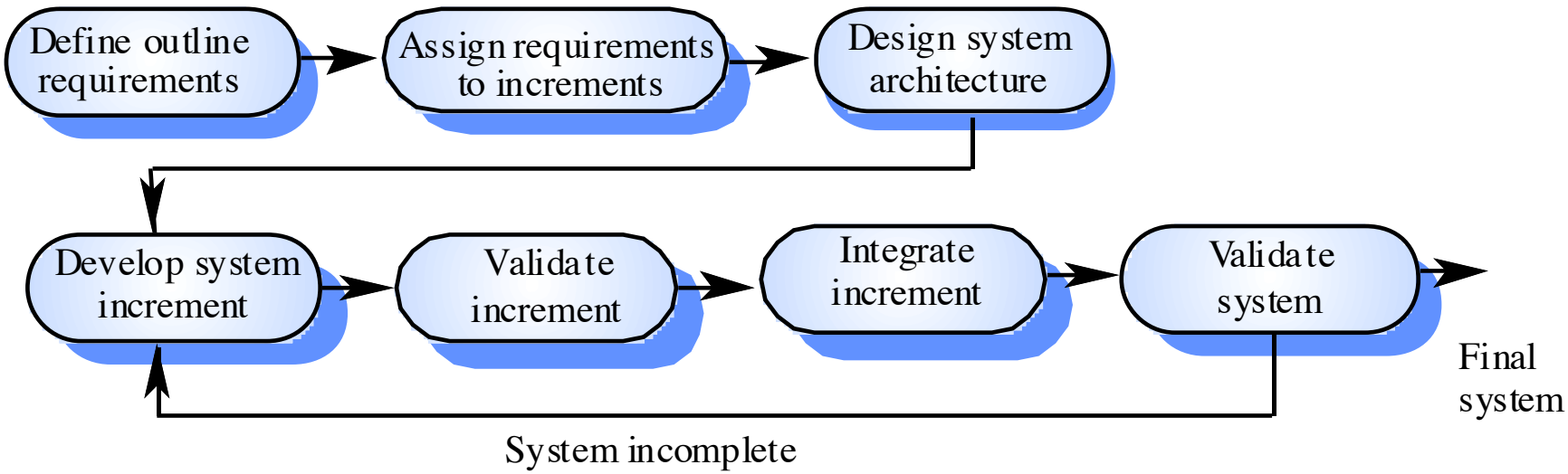
Processus itératifs

- Exigences de système évoluent toujours tout au long de la réalisation d'un projet, alors l'itération de processus où des étapes précédentes sont refaites est une partie du processus pour des grands systèmes
- Itération peut être appliquée dans tous les modèles génériques de processus
- Deux approches concernantes
  - Développement incrémental
  - Développement spiral

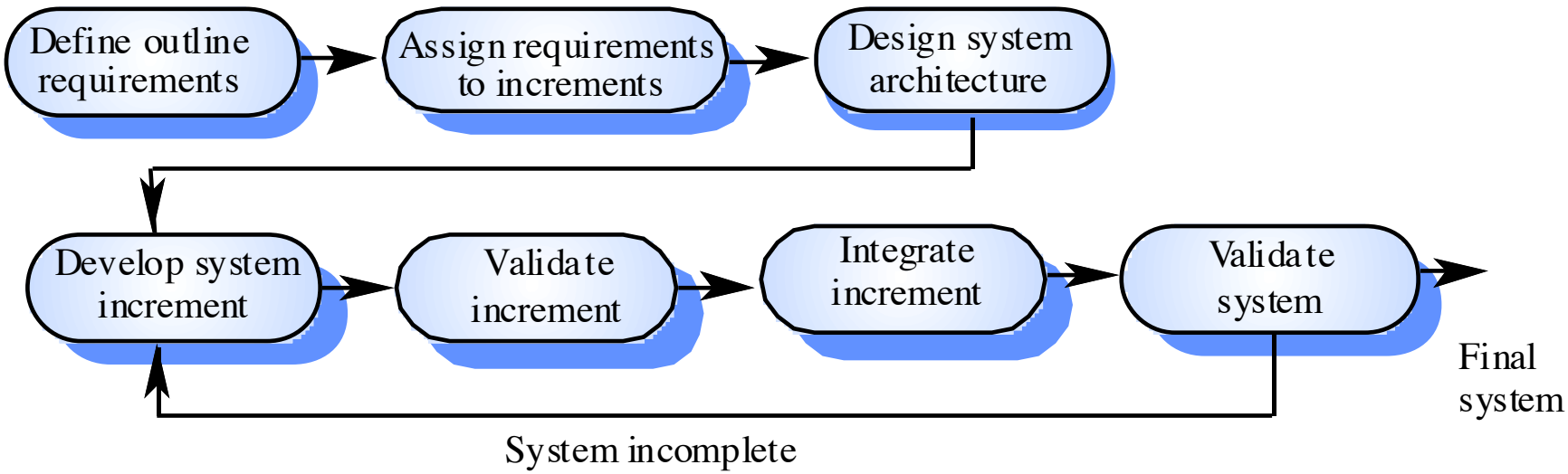
Développement incrémental

- Plutôt de livrer le système en différentes livraisons simples, le développement et la livraison sont divisés en ‘ itération ’ et chaque itération livre une partie des fonctionnalités requises
- Exigences d’usager sont mises en priorité en fonction de leur valeur d’affaire. L’exigence avec la priorité plus haute est réalisée en premier

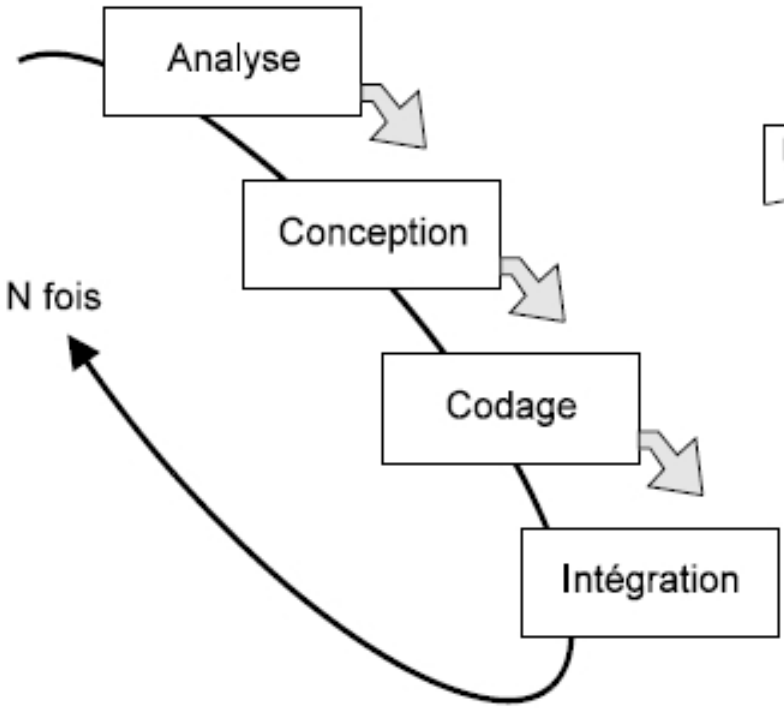
Développement incrémental

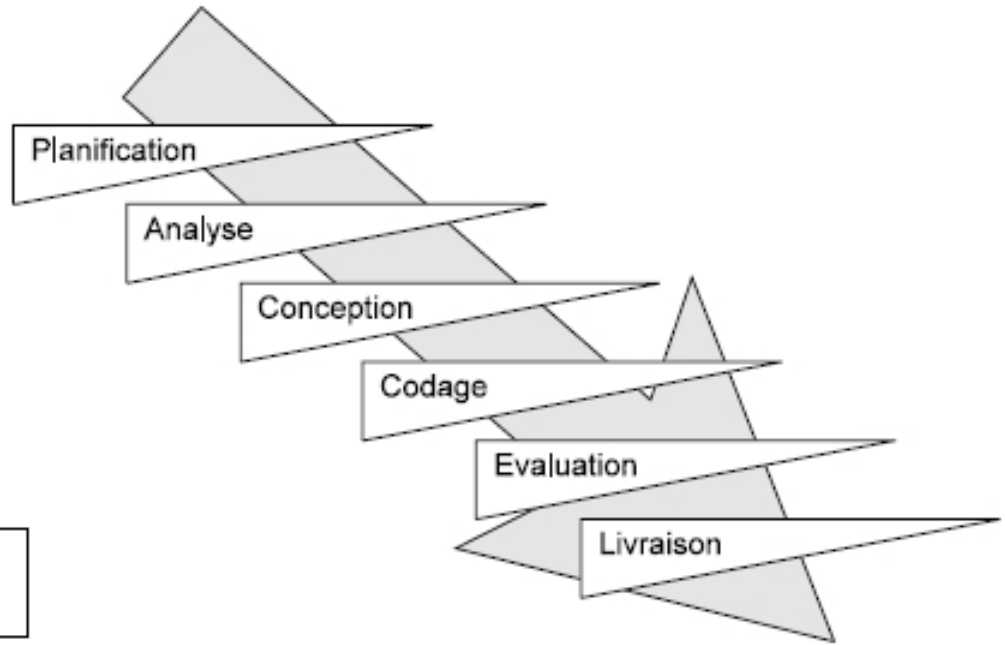


Développement incrémental



Développement incrémental





Ref.: Pierre Muller, www.irisa.fr/triskell/members/pierre-alain.muller/teaching/demarche

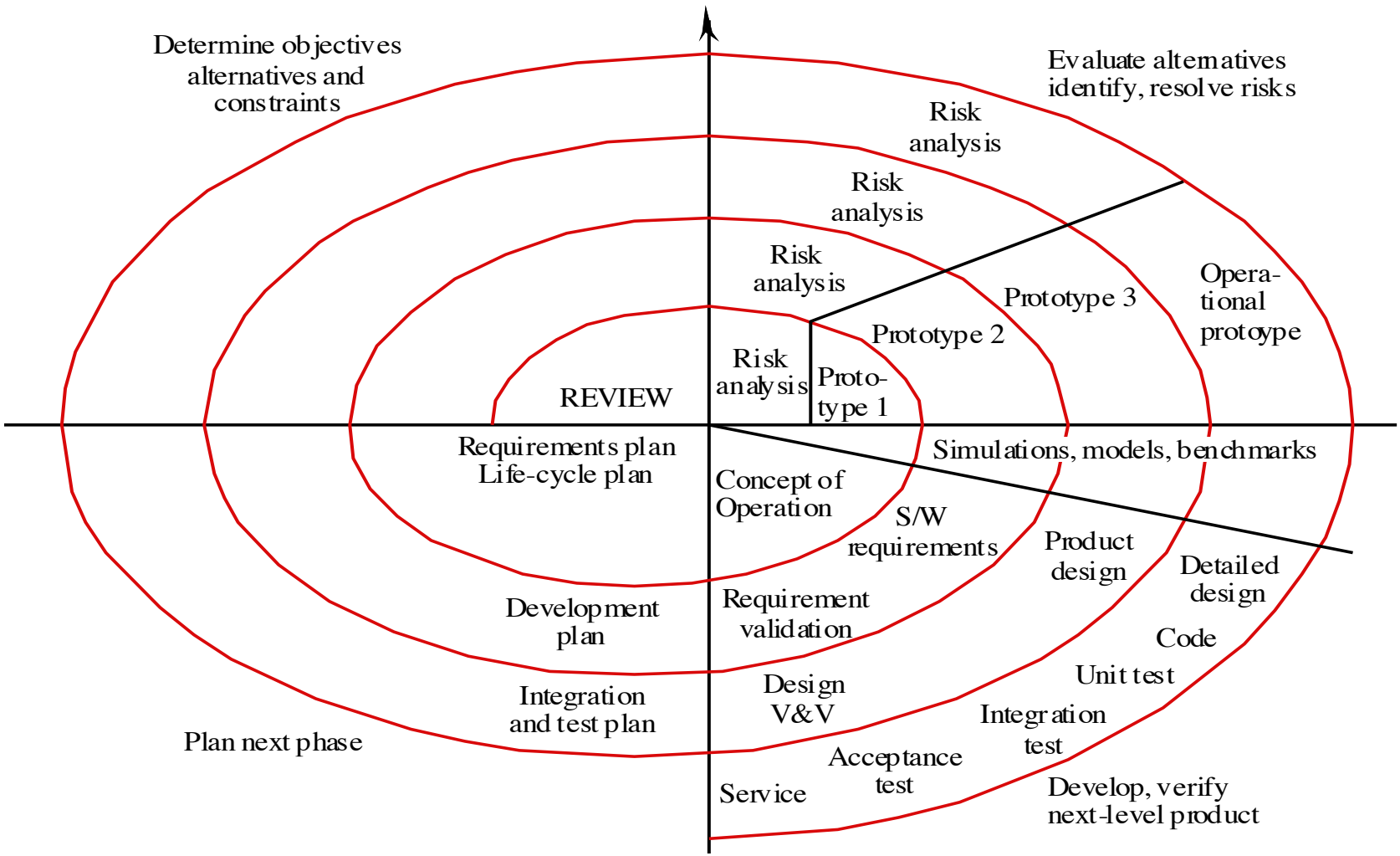
Avantages de développement incrémental

- Valeur d'usager peut être livrée après chaque itération, donc des fonctionnalités du système sont disponibles tôt
- Itération tôt (avec prototype) aide à découvrir les exigences pour l'itération prochaine
- Risque est plus base pour l'échec de projet
- Plus de tests sont réalisés

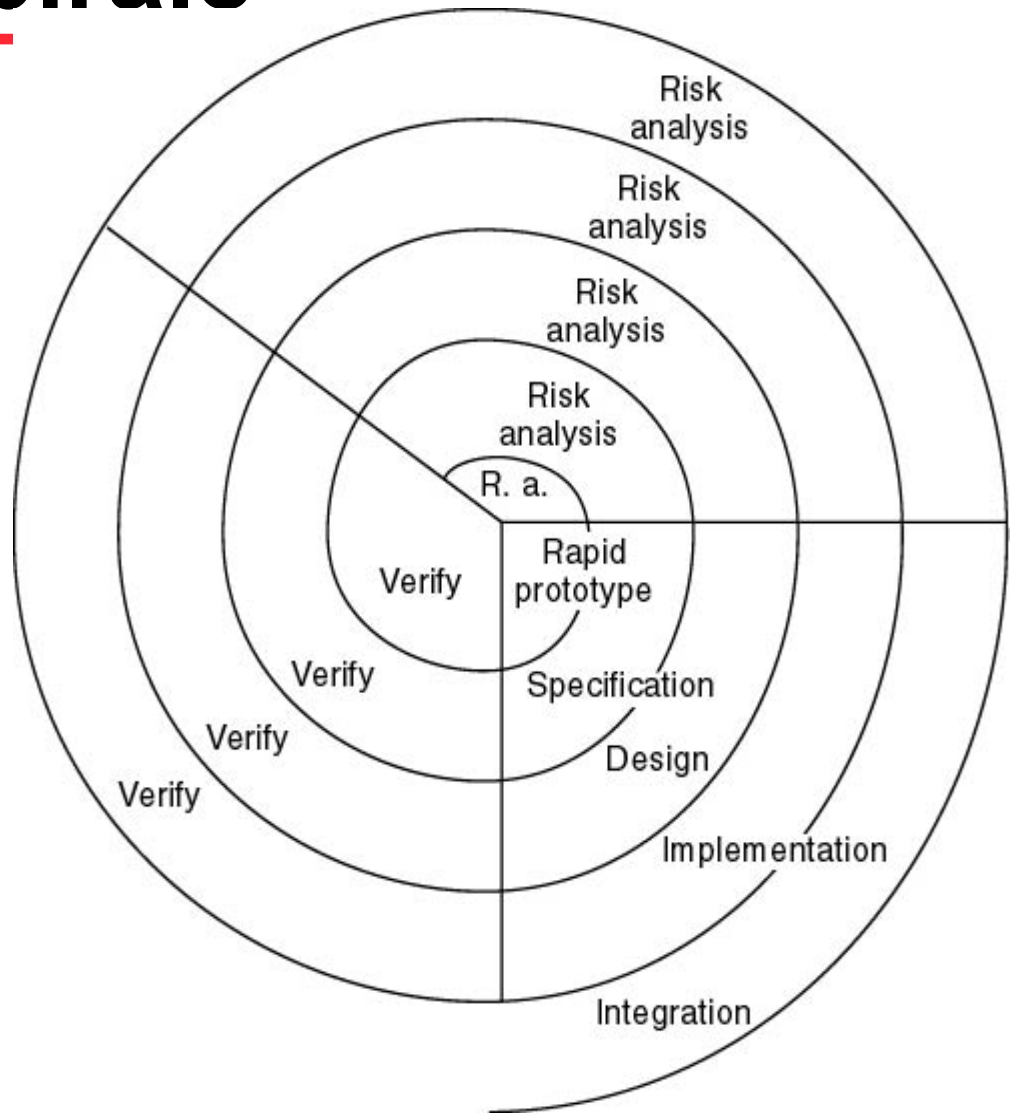
Développement en spiral

- Processus est représenté comme un spiral plutôt qu'une séquence des activités avec 'retour arrière'
- Chaque boucle dans le spiral représente une phase dans le processus
- Pas des phases fixées telles que spécification ou conception - boucles sont choisis en fonction de besoin
- Risques sont explicitement évalués et résolus à travers le processus

Modèle en spirale



Modèle en spirale



Phases du modèle en spirale

- Identification des objectifs
  - Les objectifs pour chaque phase du projet sont identifiés
- Évaluation et réduction des risques
  - Les risques sont identifiés, analysés et des informations sont recherchées pour les réduire
- Développement et validation
  - Un modèle approprié est choisi pour la phase suivante de développement.
- Planification
  - Le projet est révisé et des plans sont établis pour le tour suivant de spirale

Rational Unified Process (RUP)

Open Unified Process

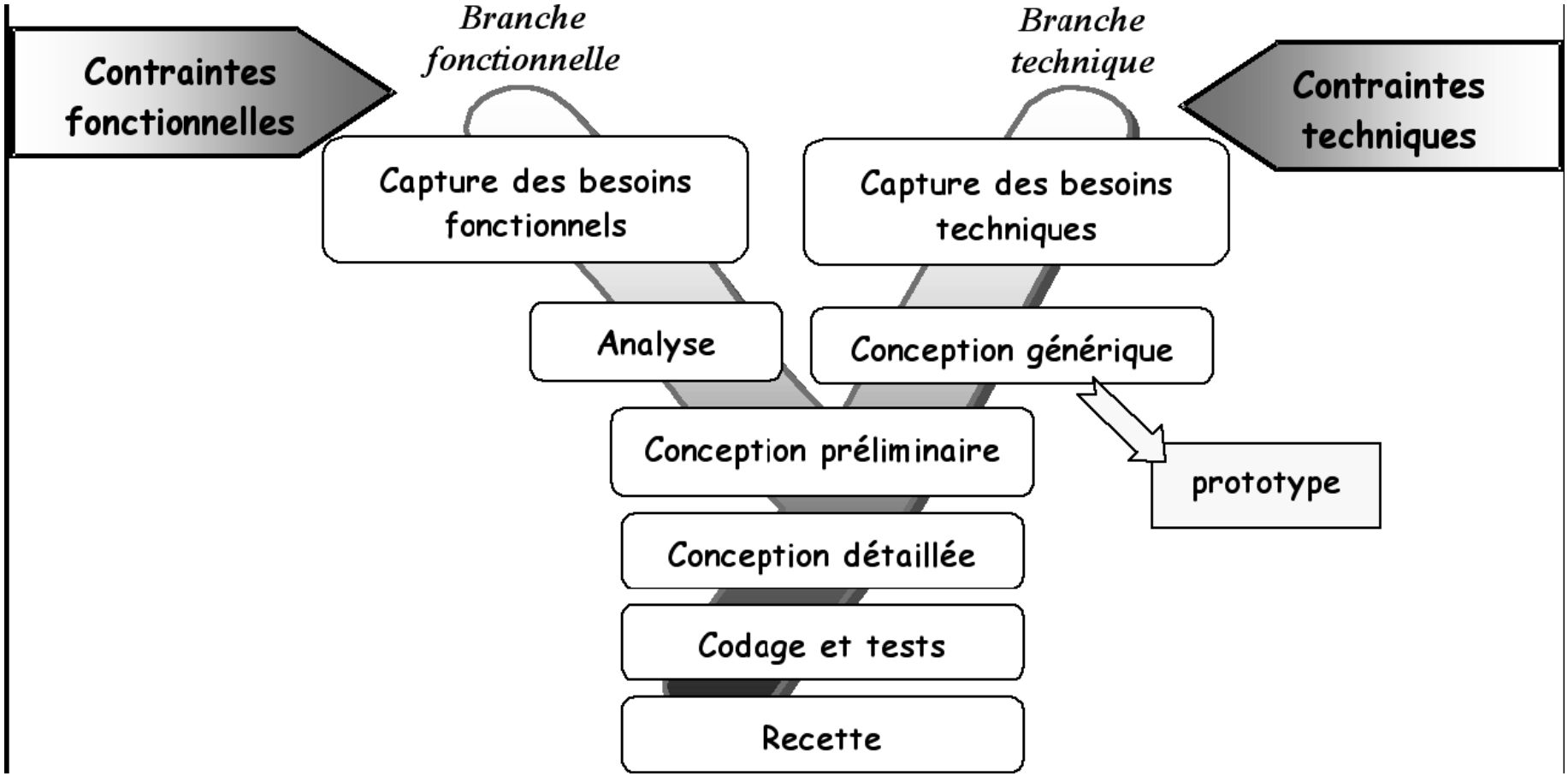
http://epf.eclipse.org/wikis/openup/

Extreme programming

- Une nouvelle approche pour le développement basée sur le principe pour lequel le développement et la livraison en petites itérations des fonctionnalités sont fondamentaux
- Compter sur l'amélioration continue du code, la participation d'usager et la programmation en binôme

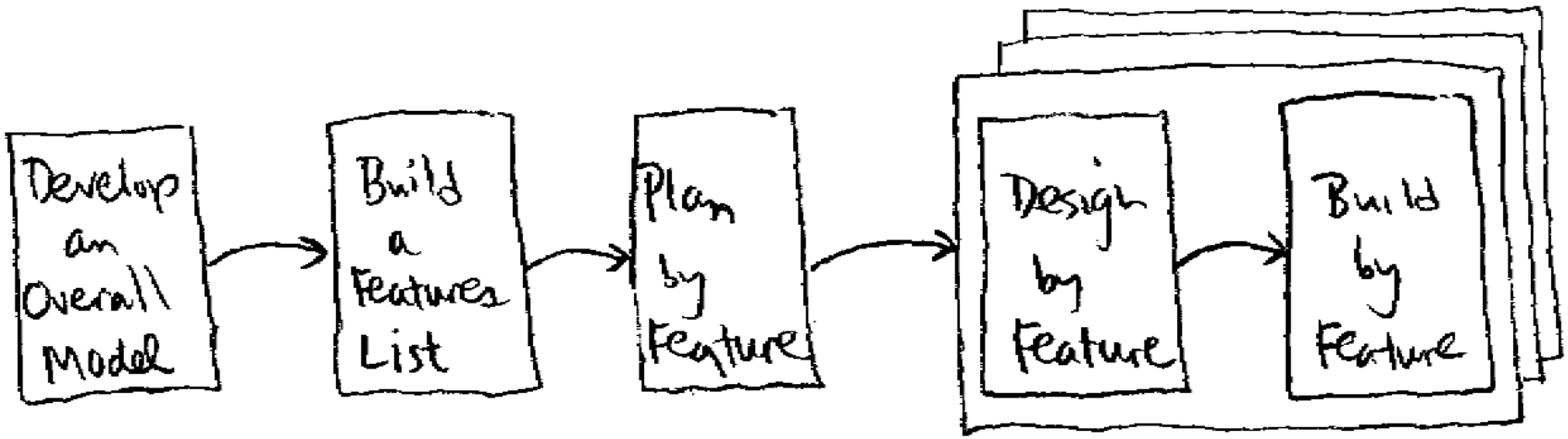
Une introduction de XP 

Processus Y (Pascal Roques)



Ref.: Pascal Roques, UML en action, Groupe Eyrolles, 2003

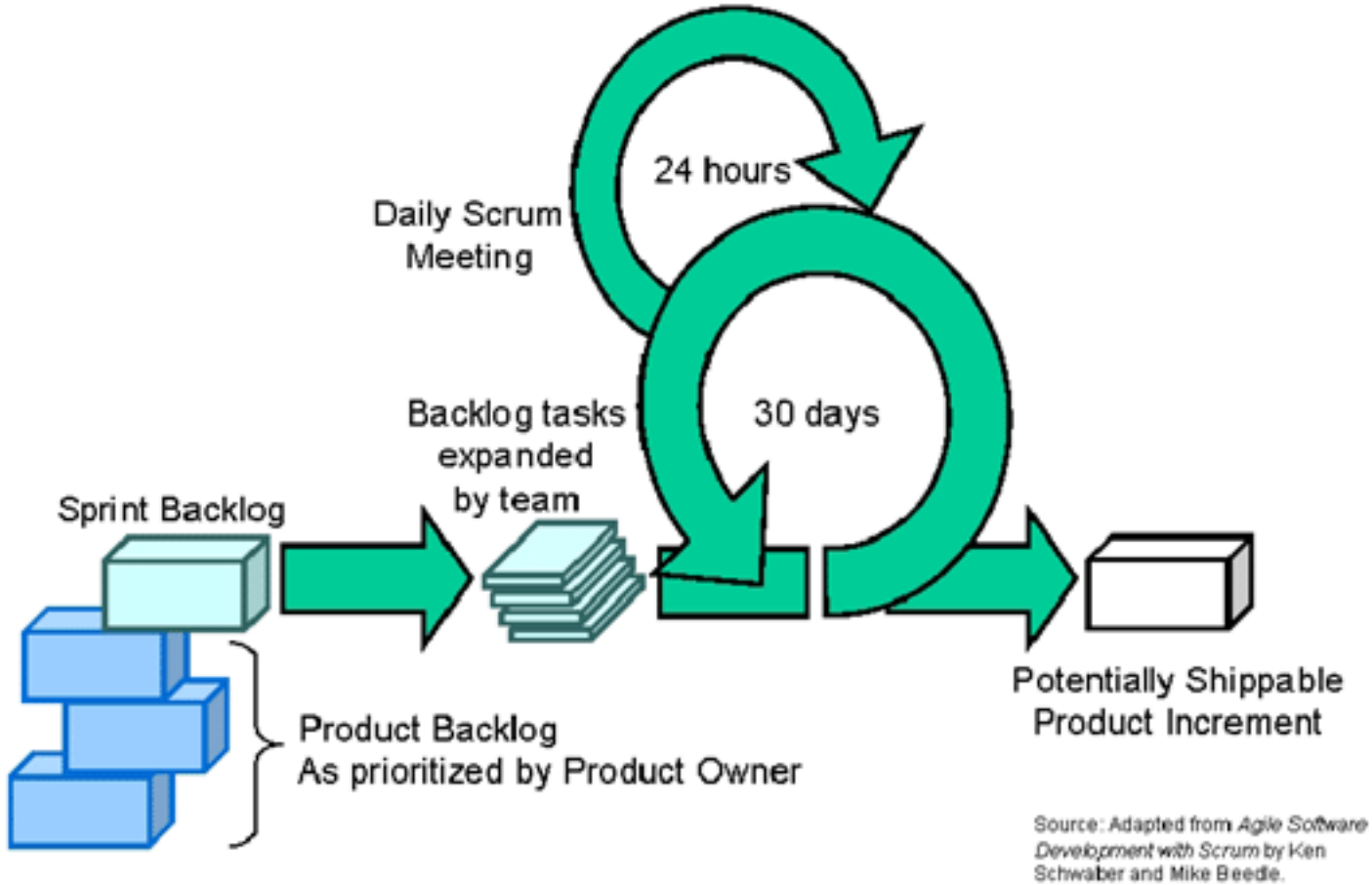
Feature-Driven Development



Presentation de FFD de Peter Coad

http://www.featuredrivendevelopment.com

SCRUM méthode



SCRUM Alliance
http://www.scrumalliance.org/

Points clés

- Processus logiciels sont des activités impliquées dans la production et évolution d'un système logiciel. Elles sont représentées dans un modèle de processus logiciel
- Modèles de processus décrivent l'organisation des activités
- Modèles de processus itératifs décrivent le processus logiciel comme un cycle d'activités

Source de références

- Software Engineering, Ian Sommerville, http://www.comp.lancs.ac.uk/computing/resources/IanS/SE8/index.html
- Processus Software Engineering: A Practitioner's Approach (6th Edition), R.S. Pressman, McGraw-Hill, 2005.
- Timothy C. Lethbridge and Robert Laganière, McGraw Hill, 2001, Object-Oriented Software Engineering: Practical Software Development using UML and Java
- http://www.ambysoft.com/
- Etc.