Génie Logiciel II(INF345)

Année Académique : 2025/2026

Etablissement : Faculté des sciences

Niveau : Licence Fonda, Troisieme année Année

Enseignants :

M. BAYANG SOULOUKNA P.

Programme

Introduction générale

Les modèles du développement des logiciels

Analyse et définition des besoins

Gestions des versions et des configurations

Modélisation (UML; Z; DFD…)

les interfaces graphiques

Informations générales

- Types de cours :
  - Cours magistral
  - Travaux dirigés
  - Travail Personnel Encadré
- Organisation des évaluations :
  - Contrôle continu
  - TPE
  - Examen
  - Session de rattrapage
- Durée : 60h

Bibliographie recommandée

- *Software Engineering*. Ian Sommerville. Addison-Wesley Pub Co; ISBN: 020139815X, 7th edition, 2004
- *Software Engineering: A Practioner's Approach*. Roger S. Pressman. McGraw Hill Text; ISBN: 0072496681; 5th edition, 2001
- *Using UML: Software Engineering with Objects and Components*. Perdita Stevens and Rob J. Pooley. Addison-Wesley Pub Co; ISBN: 0201648601; 1st edition, 1999
- *Designing Object-Oriented Software*. Rebecca Wirfs-Brock and Brian Wilkerson and Lauren Wiener. Prentice Hall PTR; ISBN: 013629825 7; 1990

Littérature

- *eXtreme Programming Explained: Embrace Change*. Kent Beck. Addison-Wesley Pub Co; ISBN: 0201616416; 1st edition (October 5, 1999)
- *The CRC Card Book*. David Bellin and Susan Suchman Simone. Addison-Wesley Pub Co; ISBN: 0201895358; 1st edition (June 4, 1997)
- *The Mythical Man-Month: Essays on Software Engineering*. Frederick P. Brooks. Addison-Wesley Pub Co; ISBN: 0201835959; 2nd edition (August 2, 1995)
- *Agile Software Development*. Alistair Cockburn. Addison-Wesley Pub Co; ISBN: 0201699699; 1st edition (December 15, 2001)
- *Peopleware: Productive Projects and Teams*. Tom Demarco and Timothy R. Lister. Dorset House; ISBN: 0932633439; 2nd edition (February 1, 1999)
- *Succeeding with Objects: Decision Frameworks for Project Management*. Adele Goldberg and Kenneth S. Rubin. Addison-Wesley Pub Co; ISBN: 0201628783; 1st edition (May 1995)
- *A Discipline for Software Engineering*. Watts S. Humphrey. Addison-Wesley Pub Co; ISBN: 0201546108; 1st edition (December 31, 1994)

Planning

Date	***Leçons***
	Introduction(RAPPELS) — Le cycle de vie du logiciel
	Collecte des besoins
	Modelisations / Conception des Interfaces Utilisateur
	Planification Gestion de projet
	Metriques logicielle / Qualité logicielle
	Examen Final

Chapitre 1 :
Introduction générale

Programme

- **C'est quoi le génie logiciel?**
- Le cycle de vie du logiciel
- Le développement itératif
- Les activités de développement du logiciel
- Documentation
- Méthodes et Méthodologies

Introduction générale

Matériel et logiciel

Systèmes informatiques

- 80 % de logiciel
- 20 % de matériel

Constat :

- Le matériel est relativement fiable
- Le marché est standardisé

Les problèmes liés à l'informatique sont essentiellement des problèmes de Logiciel

Pourquoi le génie logiciel?

Vue naive :

Spécifications du problème —*codage*→ Programme Final

Mais...

- D'où est-ce que les *spécifications* sont venues?
- Comment savez-vous que les spécifications correspondent aux *besoins des utilisateurs*?
- Comment avez-vous décidé de *structurer* votre programme?
- Comment savez-vous que votre programme *respecte réellement les spécifications*?
- Comment savez-vous que votre programme *fonctionne toujours correctement*?
- Que faites-vous si les besoins des *utilisateurs changent* ?
- Comment *partagez-vous les tâches* si vous travaillez en équipe?

C'est quoi le génie logiciel? (I)

Quelques Définitions et Conséquences

"L'art de developper des logiciels de qualité en respectant le temps et le budget impartis"

- Compromis entre la perfection et les contraintes physiques
  - le génie logiciel doit traiter des contraintes réels
- Etat de l'art!
  - La communauté décide des "bonnes practiques" + enseignements permanentes

C'est quoi le génie logiciel? (II)

"multi-person construction of multi-version software"

— Parnas

- Travail en équipe
  - Problème de passage à l'échelle ("programmer bien" n'est pas suffisant) + Problèmes de Communication
- Le logiciel doit évoluer ou périr
  - Les Changements sont la norme, pas l'exception

C'est quoi le génie logiciel? (III)

"Le génie logiciel est <u>different</u> des autres domaines de l'ingénieurie"

— Sommerville

- Pas de contrainte Physique
  - limite = cerveau humain
- Les contraintes sont d'ordre politique
  - Equilibre entre les clients

Définition génie logiciel ?

- « On appelle Génie Logiciel l'application des méthodes scientifiques au développement de théories, méthodes, techniques, langages et outils favorisant la production de logiciel de qualité ». Bertrand Meyer
- En resume le genie logiciel est un :
  - Ensemble
    - de méthodologies
    - de méthodes
    - de techniques
    - D'outils
  - pour produire, utiliser et maintenir du logiciel de qualité industrielle

Définition logiciel

Le terme computer software ou logiciel est souvent synonyme de programme ou de code source. Souvent, il est aussi synonyme de produit. Ainsi le logiciel inclut

- le code source,
- tous les documents associés (Les documents de définition (requirement), les spécifications conceptuelles, le code source, les plans de tests, les principes d'opération, les procédures d'assurance de la qualité, les rapports des problèmes du logiciel, les procédures de maintenance,) ;
- enfin la documentation(les manuels d'utilisateur, les instructions d'installation, les aides de formation constituent.).

Les produits logiciels incluent aussi bien les logiciels système que les applications développées dans un but spécifique.

Programme

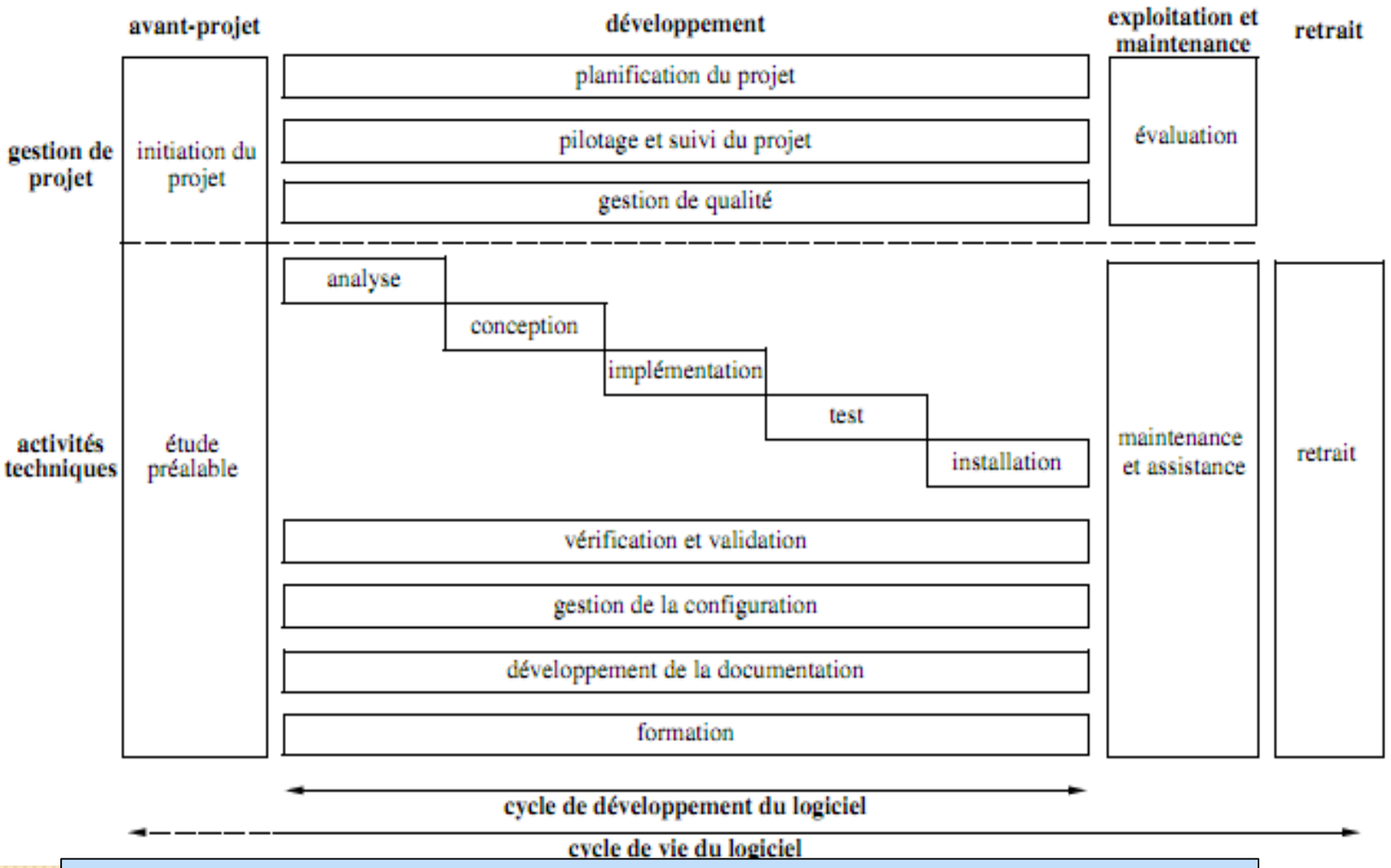
- C'est quoi le génie logiciel?
- **Le cycle de vie du logiciel**
- Le développement itératif
- Les activités de développement du logiciel
- Documentation
- Méthodes et Méthodologies

cycle de vie logiciel

Cycle de vie du logiciel ?

- Enchaînement des activités de développement logiciel
- Definition des Pré et Post conditions pour chaque phase
- Procédures de gestion et d'encadrement
- Procédures de mesures
- Cycle de vie logiciel : synonyme de méthodologie logiciel

Cycle de vie du logiciel



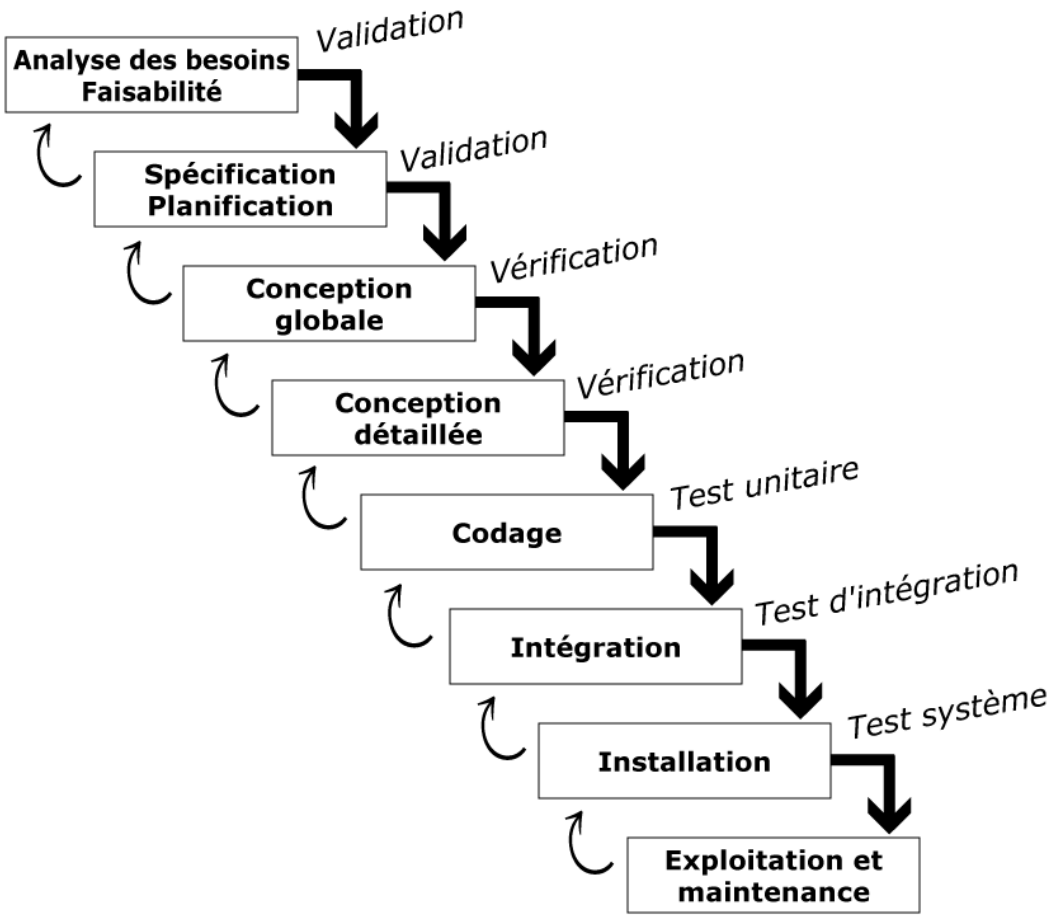
NB: ces activités ne sont pas necessairement sequentielles

Programme

- C'est quoi le génie logiciel?
- Le cycle de vie du logiciel
- **Le développement itératif**
- Les activités de développement du logiciel
- Documentation
- Méthodes et Méthodologies

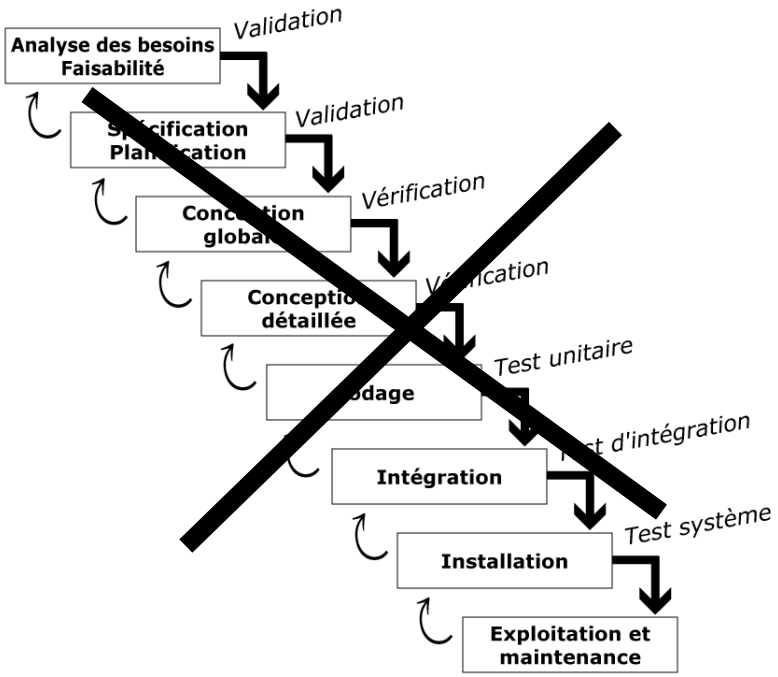
Cycle de vie classique

Les modèles classiques de cycle de vie des logiciels définissent les activités de développement comme une "**cascade**" d'étapes qui se déroulent les unes après les autres.



Cycle de vie classique

Les modèles classiques de cycle de vie des logiciels définissent les activités de développement comme une "cascade" d'étapes qui se déroulent les unes après les autres.



Le modèle en cascade n'est pas réaliste pour plusieurs raisons :

- les exigences doivent être *congelés* trop tôt dans le cycle de vie
- Les exigences sont *validées trop tard*

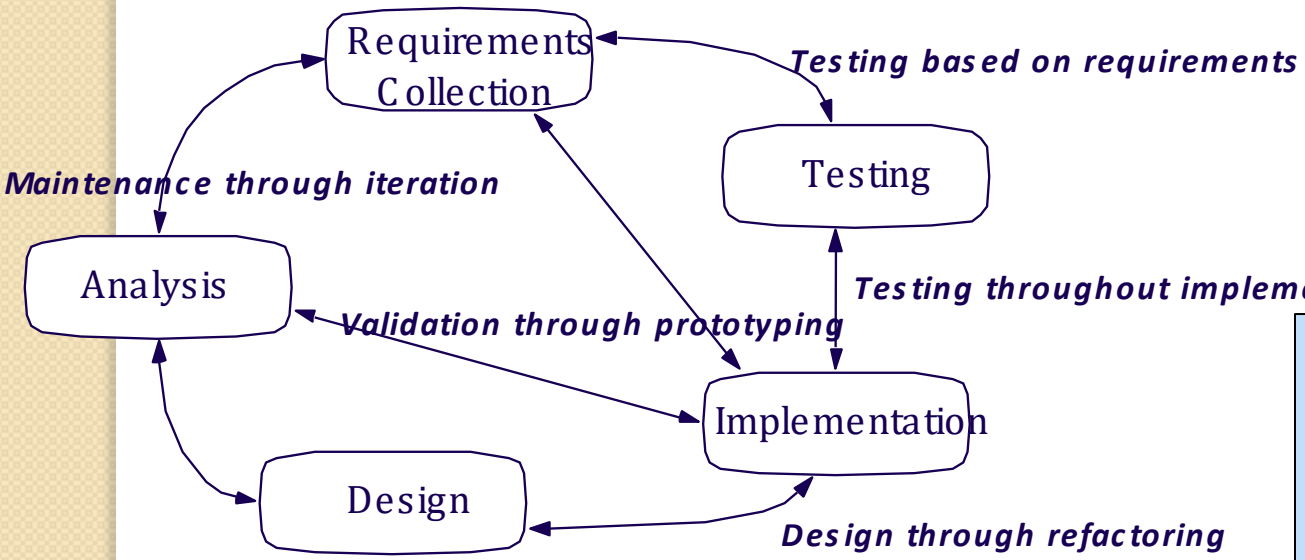
Problèmes avec le cycle de vie en cascade

1. "les projets réels suivent rarement le flux séquentiel que le modèle propose. *Des itération* se produisent toujours et créent des problèmes dans l'application du paradigme"
2. "Il est souvent *difficile* pour le client de déclarer *tous les besoins* explicitement. Le cycle de vie classique exige cela et a de la difficulté à gérer les incertitudes naturelles qui existent au début de nombreux projets."
3. "Le client doit avoir de la patience. Une *version du programme(s)* ne sera pas disponible avant la fin du projet. Une erreur majeure est resté inaperçue jusqu'à ce que le programme final soit construit, alors cela peut être désastreux."

— *Pressman, SE, p. 26*

Cycle de vie Itératif

Dans la pratique, le développement est toujours itérative, et *toutes* les activités progressent en parallèle.



Si le modèle de cascade est une pure fiction, pourquoi est-il toujours le modèle logiciel dominant?

Cycle de vie Itératif

Planifiez pour *itérer* votre analyse, conception et implémentation.

- Vous n'aurez pas le bon produit dès la première fois, alors *intégrer*, *valider* et *tester* aussi souvent que possible.

"Vous devez utiliser le développement itératif uniquement sur les projets que vous souhaitez *réussir*"

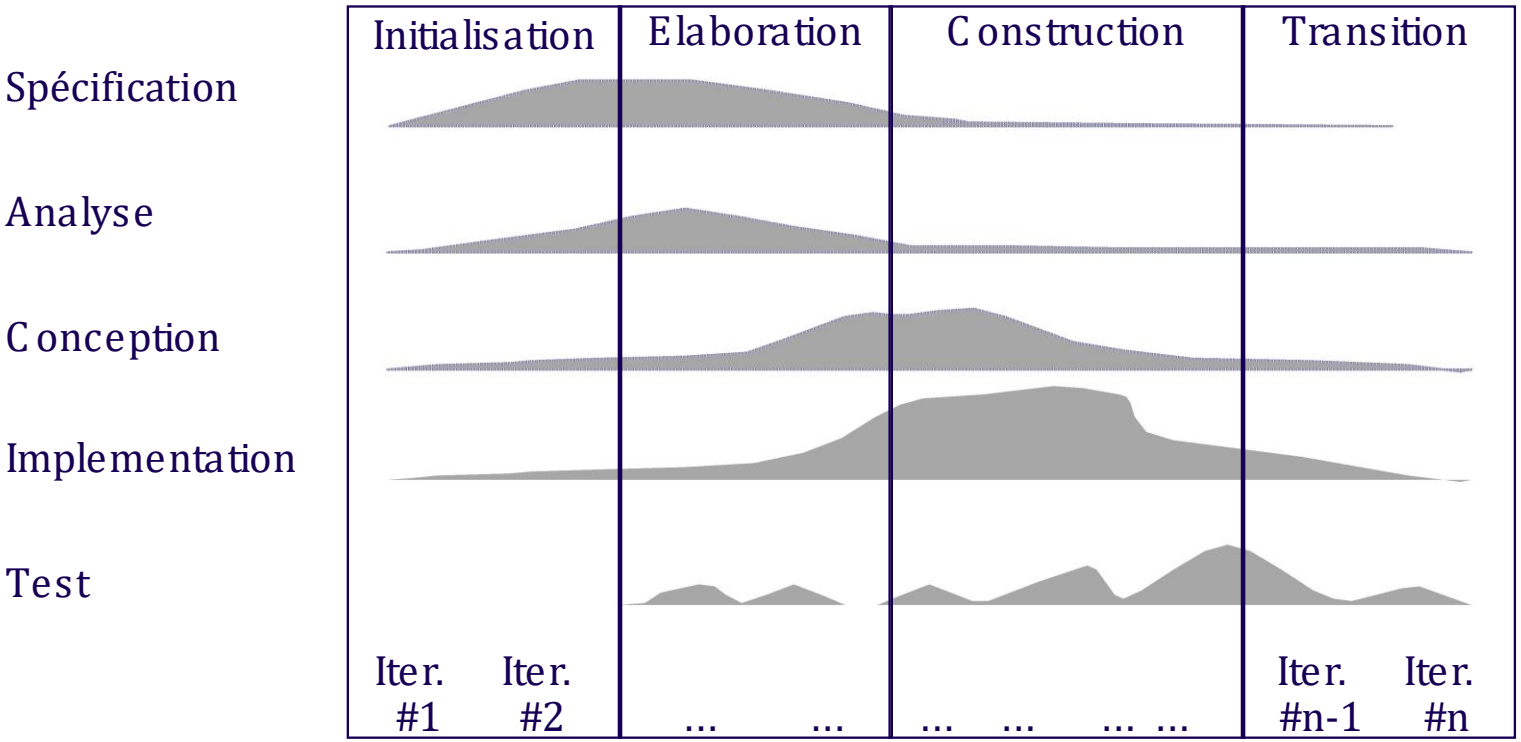
- *Martin Fowler, UML Distilled*

Développement Incrémental

Plannifier le développement *incrémental* (à savoir, prototype) du système.

- Si possible, *avoir toujours une version en cours d'exécution du système*, même si la plupart des fonctionnalités sont encore à implementer.
- *Intégrer* de nouvelles fonctionnalités dès que possible.
- *Valider* les versions incrémentales par rapport aux exigences de l'utilisateur.

Le Processus Unifié



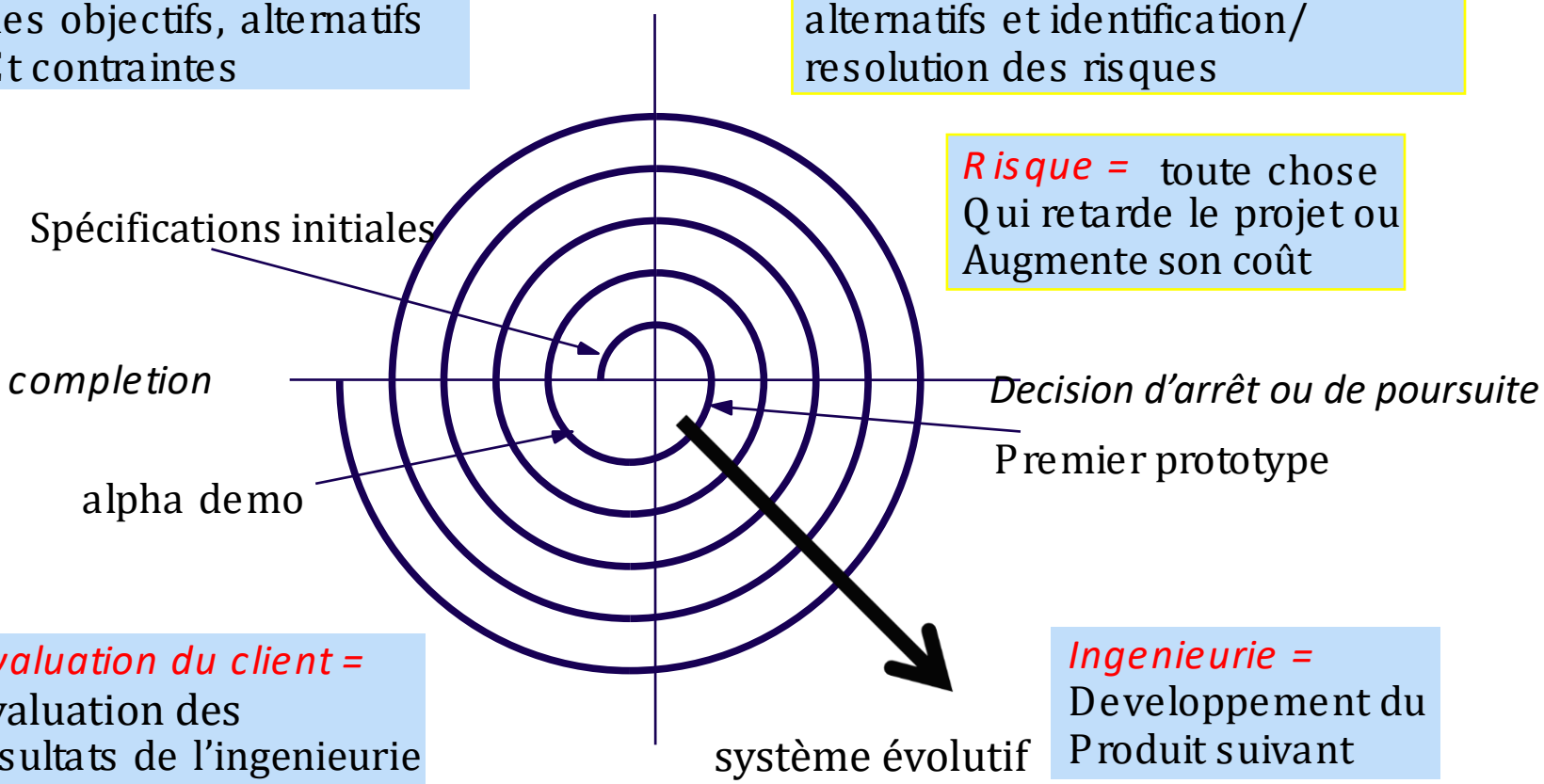
Comment envisagez-vous le nombre d'itérations?
Comment décidez-vous que vous avez achever ?

Cycle de vie en spiral de Boehm

Planning = determination des objectifs, alternatifs Et contraintes

Analyse de risque = Analyse des alternatifs et identification/ resolution des risques

Risque = toute chose Qui retarde le projet ou Augmente son coût



Evaluation du client = Évaluation des resultats de l'ingenieurie

Ingenieurie = Developpement du Produit suivant

Programme

- C'est quoi le génie logiciel?
- Le développement itératif
- Le cycle de vie du logiciel
- **Les activités de développement du logiciel**
- Méthodes et Méthodologies

Collecte des besoins

Les besoins des clients sont souvent exprimés de façon *informelle* :

- Fonctionnalités
- scénarios d'utilisation

Bien que les besoins peuvent être documentés sous forme écrite, ils peuvent être *incomplètes*, *ambiguës*, voire *incorrectes*

Changement des besoins

Les besoins *changeront*!

- *insuffisamment capturé ou exprimé* en premier lieu
- les besoins des utilisateurs et des entreprises *peuvent changer* au cours du projet

La validation est nécessaire *tout au long* du cycle de vie du logiciel, non seulement lorsque le "système final" est livré!

- construire un *feedback* constant dans votre plan de projet
- plan pour les *changement*
- le *prototypage* rapide [par exemple, l'interface utilisateur] peut aider à clarifier les exigences

Analyse des besoins et Spécification

L'Analyses est le processus de spécification de *quoi* un système doit faire.

- L'intention est de fournir une compréhension claire de ce que le système est et ce que ses concepts sous-jacents sont

Le résultat de la phase d'analyse est le *document de spécification*.

Est-ce que le cahier des charges correspondant aux besoins réels du client?

Analyse Orientée Objet

Une *analyse orientée objet* se traduit par des modèles du système qui décrivent :

- Les *classes* d'objets qui existent dans le système
  - *Les responsibilités* de ces classes
- *Les relations* entre ces classes
- *Les cas d'utilisation* et les *scenarios* qui décrivent
  - *Les operations* qu'on peut effectuer sur le système
  - Les *séquences* admissibles de ces opérations

Prototypage (I)

Un *prototype* est un logiciel développé pour tester, explorer ou de valider une hypothèse, à savoir pour *réduire les risques*.

Un *prototype d'exploration*, aussi connu comme un prototype jetable, est destiné à *valider les exigences* ou *d'explorer des choix de conception*.

- Prototype UI - valider les besoins des utilisateurs
- prototypage rapide - valider les exigences fonctionnelles
- prototype expérimental - valider la faisabilité technique

Prototypage (II)

Un *prototype évolutif* est destiné à évoluer dans les étapes en un produit fini.

- Faire "grandir" l'application de façon itérative , la conception et la refactorisation tout au long du chemin,

Tout d'abord le faire,
puis le faire bien,
puis le faire rapidement

Conception

La conception est le processus de préciser *comment* le comportement du système spécifié sera réalisé à partir des composants logiciels. Les résultats sont *l'architecture* et les *documents de conception détaillée*.

la conception orientée objet propose des modèles qui décrivent:

- comment les opérations du système sont implémenté par *l'interaction des objets*
- comment les classes se rapportent les uns aux autres et la manière dont elles sont liées par *l'héritage*
- les *attributs* et les *opérations* associées à des classes

La conception est un processus itératif, procédant en parallèle avec l'implementation!

Loi de Conway

- “les organisations qui définissent des systèmes ... sont contraintes de les produire sous des designs qui sont des copies de la structure de communication de leur organisation.”

Implementation et Tests

L'Implementation est l'activité de la *construction* d'une solution logicielle aux exigences du client.

Tests est le processus de *validation* que la solution est conforme aux exigences.

- Le résultat de l'implémentaion et des tests est une solution *entièrement documentée* et *validée*.

Conception, Implementation et Tests

La conception, l'implementation et les tests sont des activités itérative

- L' implementation n' «implémente pas la conception", mais le document de conception *documente l'implémentation*!

- Les tests du système reflètent la spécification des exigences
- Le test et l'implémentation vont main dans la main
  - Idéalement, la spécification de cas de test *précède* la conception et l'implémentation

Maintenance

Maintenance est le processus de changement d'un système après qu'il a été déployé.

- *Maintenance corrective* : identifier et reparer le *defauts*
- *Maintenance adaptative* : *adapte* la solution existantes à de nouvelles plateformes
- *Maintenance perfective* : réponde à une demande d'amélioration du logicie
- *maintenance préventive* réponde à une demande d'amélioration du logiciel

Dans un cycle de vie en spirale, toute activité après la livraison et le déploiement du premier prototype peut être considéré comme "maintenance"!

Les activités de maintenance

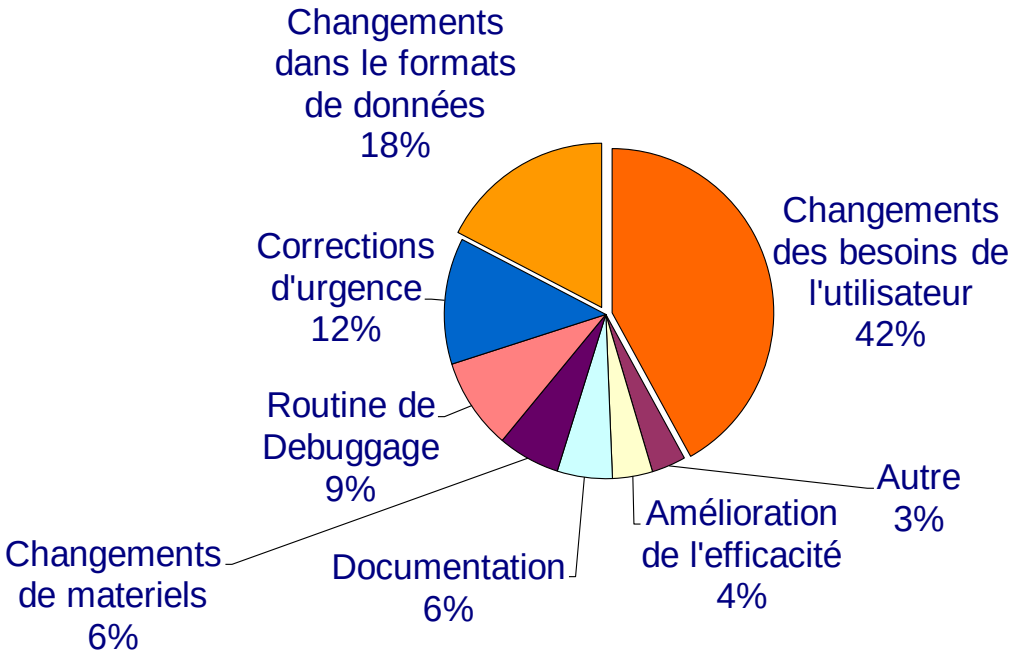
La “Maintenance” comprend :

- La gestion de la configuration et de la version
- Le reengineering
- mise à jour de tous les documents d'analyse, de la conception et manuel d'utilisateur

Des tests automatisés répétables permettent l'évolution et le refactoring

Coût de la Maintenance

La “Maintenance” c’est *80% du coût du logiciel!*



Consequences : la plupart des coûts du projet concernent la poursuite du développement *après* le déploiement

– Lientz 1979

Programme

- C'est quoi le génie logiciel?
- Le développement itératif
- Le cycle de vie du logiciel
- Les activités de développement du logiciel
- **Documentation**
- Méthodes et Méthodologies

Documentation des Logiciels

La documentation, malgré quelques efforts de normalisation et d'automatisation, est le plus souvent inadaptée parce qu'il n'y a plus de distinction claire entre ces deux types : celui de « production » qui permet de décrire, de concevoir, de produire et de faire évoluer le produit du point de vue de l'équipe projet ; celui « d'utilisation » qui permet de décrire de comprendre, d'enseigner et d'utiliser le produit du point de vue de l'utilisateur final.

Documentation des Logiciels

Les documents d'un logiciel

- doc interne ou de production : informaticiens
  - développement du logiciel
  - maintenance du logiciel
- doc externe ou d'utilisation : utilisateurs
  - présentations des fonctions
  - pas de détail de réalisation
- L'ensemble des documents
  - un cahier des charges: besoins
  - un dossier de spécifications
  - un dossier de conception détaillée
  - un dossier de programmation
  - un dossier des procédures de tests
  - un manuel d'installation et de mise en œuvre
  - un manuel d'utilisation
  - …

Documents Gestion des projets

Chaque document doit respecter une certaine norme. Elle doit permettre de connaitre à tout moment l'état du projet et des composants logiciels, conserver l'historique du système . Elle doit contenir:

- Date : création, dernière mise à jour, fin de validité s ipossible.
- Auteur : maitrise d'ouvrage ou maitre d'ouvre.
- Objet : décrire ce qui va être ou ce qui est attaché.
- Type : standard/référentiel de developpement ?
- Version : dans un ensemble de documents

Document / phase

documents \ phases	avant-projet	analyse	conception générale	conception détaillée	implémen-tation	tests unitaires	intégration et test d'intégration	installation et test de réception
cahier des charges du projet	■	→						
spécification		■	→					→
conception générale			■	→			→	
conception détaillée				■	→	→		
listages					■	→		
tests unitaires				▒	?	■		
test d'intégration			▒	?			■	
test de réception		▒	?					■
manuels d'utilisation et d'exploitation		▒	→ ?				■	→ ?

■ document élaboré entièrement pendant la phase

▒ document partiellement élaboré pendant la phase

→ document en entrée de la phase

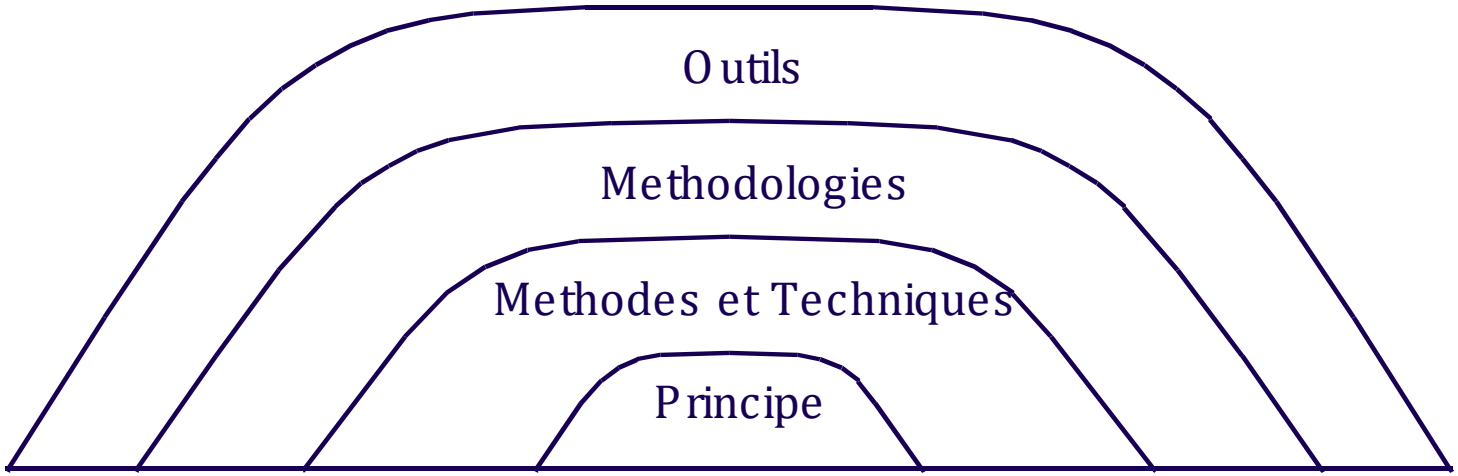
? document éventuellement complété pendant la phase

Programme

- Vue d'ensemble du cours
- C'est quoi le génie logiciel?
- Le développement itératif
- Les activités de développement du logiciel
- Documentation
- **Méthodes et Méthodologies**

Méthodes et Méthodologies

Principe = déclaration générale décrivant les propriétés souhaitables
Méthode = orientations générales régissant une activité
Technique = plus technique et mécanique que la méthode
Méthodologie = ensemble de méthodes et techniques mis ensemble



— *Ghezzi et al. 1991*

Méthodes Orienté Objet : un bref historique

Première génération:

- Adaptation des notations existantes (diagrammes ER, diagrammes d'états...): Booch, OMT, Shlaer and Mellor, ...
- techniques spécialisé :
  - cartes CRC; la conception axée sur la responsabilité; conception par contrat

Séconde génération:

- Fusion: Booch + OMT + CRC + méthodes formelles

Troisième génération:

- Unified Modeling Language:
  - Notation uniforme : Booch + OMT + Cas d'utilisations+ ...
  - Plusieurs méthodes basées sur UML (Catalysis, UP…)

Ce que vous devriez savoir!

- Comment le génie logiciel diffère de la programmation?
- Pourquoi le modèle "cascade" est irréaliste?
- Quelle est la différence entre l'analyse et la conception?
- Pourquoi planifier pour itérer? Pourquoi le développement incrémental?
- Pourquoi la programmation représente seulement une petite partie du coût d'un projet de logiciel «réel»?
- Quels sont les avantages et les inconvénients principaux des méthodes orientées objet?

Peux-tu répondre a ces questions?

- Quel est l'attrait du modèle «cascade»?
- Pourquoi les exigences changent?
- Comment pouvez-vous confirmer qu'un modèle d'analyse capture les besoins réels des utilisateurs?
- Quand arrêter l'analyse et commencer la conception?
- Quand l'implémentation peut-il commencer?
- Quels sont les bons exemples des lois de Conway?