

# Analyse des besoins et cahier des charges

- Terminologie
- La faisabilité
- L 'analyse des besoins
- Le cahier des charges

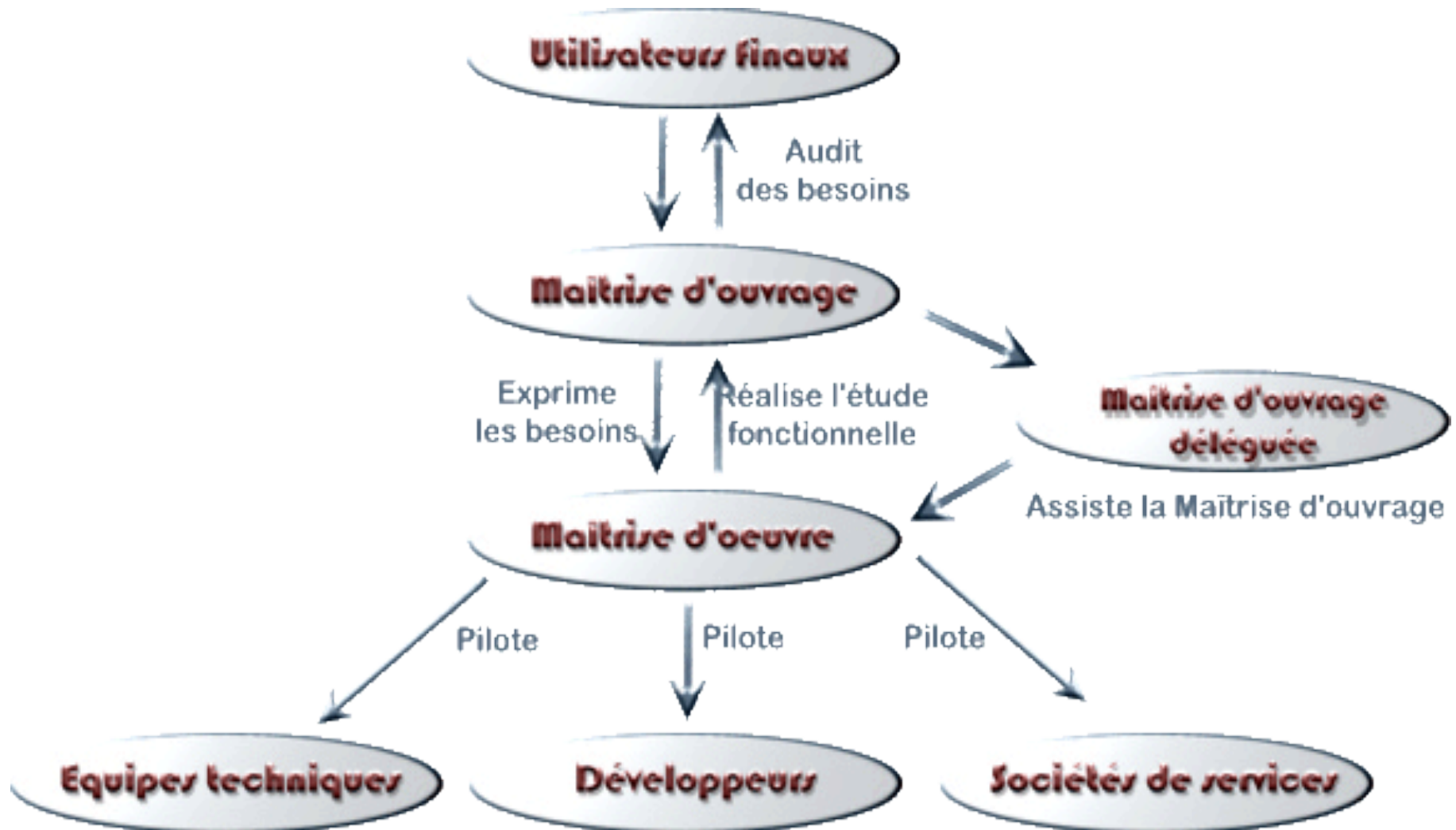
# *Systeme informatique*

- “Un ensemble d’éléments qui sont organisés pour accomplir un but prédéfini par un traitement de l’information”
- utilise des :
  - Logiciels
  - Matériels (informatiques)
  - Personnes
  - Bases de données (ensemble *organisée* de données)
  - Documentation
  - Procédures (étapes qui définissent comment utiliser les éléments du système)

# Développement d'un système

- La maîtrise d'ouvrage
  - Entité responsable de l'expression du besoin
  - Souvent non informaticien
  - *Besoin réel / budget*
  - ☞ Possibilité de **maîtrise d'ouvrage déléguée**
- La maîtrise d'œuvre
  - Entité responsable de la concrétisation de l'idée en outil informatique
  - Pas de connaissance fonctionnelle
  - *Bons choix techniques, adéquation avec les besoins, performances...*


# Différence dans les *maîtrises*



# Étude de faisabilité

- Tous les projets sont faisables !
  - étant donné des ressources et un temps infinis
- Mais les ressources sont limitées...

# Étude de faisabilité (suite)

- Faisabilité économique
- Faisabilité technique  au plus tôt
  - Risques de développement
  - Disponibilité des ressources
  - Technologie nécessaire
- Faisabilité légale
- Alternatives

# Étude de faisabilité : aspects économiques

- Analyse du rapport Coût/Bénéfice :
    - Coût du système
    - Bénéfices mesurables (en € )
    - Bénéfices **non** mesurables
      - meilleure conception
      - meilleures décisions marketing
      - satisfaction accrue du client
- ☞ L'analyse Coût/Bénéfice est souvent le moyen d'obtenir le feu vert de la direction

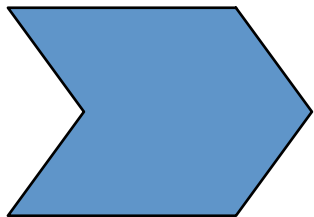
# Analyse des besoins

- Définition des besoins à différents niveaux d'abstraction :
  - Besoins de l'utilisateur
  - Besoins des composants
- Définition du système à réaliser avec le point de vue de l'utilisateur et/ou du client
  - ☞ Les utilisateurs doivent être capables de comprendre ce document
- ☞ Analyse des besoins : *LE QUOI*
- ☞ Conception : *LE COMMENT*



# Le processus d'analyse

- Processus de découverte, de raffinement, de modélisation et de spécification
- Les utilisateurs/clients et les développeurs ont des rôles **actifs**
- Les utilisateurs ne sont pas satisfaits par un système bien conçu et bien implémenté



*Les utilisateurs veulent des systèmes  
qui satisfont leurs besoins*

# Bases de la communication

- Écouter le client
  - Écoute  $\neq$  Compréhension
- Préparer les réunions
  - Connaissance du client et des contacts
  - Lecture des documents disponibles
  - Penser aux objectifs de la réunion
  - Penser aux problèmes
  - Être à l'heure...

# Initier la communication

- La première réunion peut être *bizarre*
  - Pas de connaissance des intervenants
  - Attentes différentes
  - Mais : chacun veut que cela réussisse
- Compréhension minimale du problème :
  - Qui est derrière la demande de cette réalisation ?
  - Qui va utiliser la solution proposée ? Avec quels bénéfices ?
  - Quelle serait une "bonne" solution ?
  - Quel sera l'environnement de la solution ?
  - Y-a-t-il des contraintes ? Des problèmes de performance ?
  - Qui sont les bons interlocuteurs ? => réponses "officielles"
  - Ai-je oublié des questions ?
  - A qui d'autre dois-je m'adresser ?

# Une bonne analyse

- Objectif premier : Maximiser la satisfaction des utilisateurs et des clients
- En tenant compte de 3 types de besoin
  - Normaux : besoins explicitement établis
  - Attendus : implicites, pas exprimés mais nécessaires
  - *Excitants* : allant au delà des espérances des clients

# Indications à suivre...

- Comprendre le problème avant de commencer à créer la spécification des besoins
  - Ne pas résoudre le *mauvais* problème
- Développer des prototypes des interfaces utilisateurs (IHM)
  - Les interfaces utilisateurs déterminent souvent la qualité...
- Noter et tracer l'origine et les raisons d'un besoin
- Utiliser des vues multiples sur les besoins
  - Réduit les risques de rater quelque chose
- Classer les besoins par priorité
- Travailler pour éliminer les ambiguïtés

# Le cahier des charges

- Première étape de l'expression du besoin
- Description globale des fonctions d'un nouveau produit ou des extensions à un produit existant
  - Énoncé du problème à résoudre
  - Liste des fonctions de base
  - Caractéristiques techniques
  - Priorités de réalisation
  - Facteurs de qualité
- Il doit être validé par le client et/ou l'utilisateur
- Il est la base du contrat entre clients et développeurs

# Difficultés à établir le cahier

- Expression de la faisabilité
  - utiliser une maquette pour simuler
- Précision et non ambiguïté
  - utiliser un formalisme différent du langage naturel ?
- Le cahier des charges est un document technique, sans considération économique
  - sauf si on lui adjoint un plan de projet
- Recherche de *précision, cohérence, complétude, testabilité, traçabilité, maintenabilité, flexibilité...*

# Contre les problèmes du langage naturel

- Imprécisions et ambiguïtés qui devront être levées lors de la phase d'analyse
  - ☞ Scinder le texte en paragraphes pour une meilleure traçabilité
  - ☞ Ne pas inclure plusieurs concepts dans un même paragraphe
- ☞ Ne pas mélanger :
  - Besoins : ce qui doit être fourni
  - Buts : souhait, vœu pieu, mais impossible à tester
  - Contraintes : qui doivent être décrites séparément



# Les besoins non-fonctionnels

- Restrictions ou contraintes sur un service fourni par le système :
  - plate-forme matérielle
  - temps de réponse
  - MTBF : *Mean Time Between Failures*
- Raisons :
  - besoins des utilisateurs
  - contraintes de budget, ...

☞ Ces besoins doivent être quantifiables !

# Cahier des charges *épuré*

- Couverture
- Introduction
- Spécification des besoins fonctionnels
- Spécification des besoins non fonctionnels
  - Standards à atteindre, plate-forme, taille mémoire
- Glossaire

# Couverture :

- Nom du projet / du produit
- Date
- Numéro de version
- Auteur(s)
- Responsabilités de chaque auteur
- Changements **clés** depuis la précédente version

# Un plan type norme AFNOR X50-151

## 1. Présentation générale du problème

### 1.1 Projet

#### 1.1.1 Finalités

#### 1.1.2 Espérance de retour sur investissement

### 1.2 Contexte

#### 1.2.1 Situation du projet par rapport aux autres projets de l'entreprise

#### 1.2.2 Etudes déjà effectuées

#### 1.2.3 Etudes menées sur des sujets voisins

#### 1.2.4 Suites prévues

#### 1.2.5 Nature des prestations demandées

#### 1.2.6 Parties concernées par le déroulement du projet et ses résultats (demandeurs, utilisateurs)

#### 1.2.7 Caractère confidentiel si il y a lieu

### 1.3 Enoncé du besoin (finalités du produit pour le futur utilisateur tel que prévu par le demandeur)

### 1.4 Environnement du produit recherché

#### 1.4.1 Listes exhaustives des éléments (personnes, équipements, matières...) et contraintes (environnement)

#### 1.4.2 Caractéristiques pour chaque élément de l'environnement

# Norme AFNOR X50-151 (suite)

## 2. Expression fonctionnelle du besoin

### 2.1 Fonctions de service et de contrainte

2.1.1 Fonctions de service principales (qui sont la raison d'être du produit)

2.1.2 Fonctions de service complémentaires (qui améliorent, facilitent ou complètent le service rendu)

2.1.3 Contraintes (limitations à la liberté du concepteur-réalisateur)

2.2 Critères d'appréciation (en soulignant ceux qui sont déterminants pour l'évaluation des réponses)

2.3 Niveaux des critères d'appréciation et ce qui les caractérise

2.3.1 Niveaux dont l'obtention est imposée

2.3.2 Niveaux souhaités mais révisables

# Norme AFNOR X50-151 (suite)

## 3. Cadre de réponse

### 3.1 Pour chaque fonction

#### 3.1.1 Solution proposée

3.1.2 Niveau atteint pour chaque critère d 'appréciation de cette fonction et modalités de contrôle

#### 3.1.3 Part du prix attribué à chaque fonction

### 3.2 Pour l 'ensemble du produit

#### 3.2.1 Prix de la réalisation de la version de base

#### 3.2.2 Options et variantes proposées non retenues au cahier des charges

#### 3.2.3 Mesures prises pour respecter les contraintes et leurs conséquences économiques

#### 3.2.4 Outils d 'installation, de maintenance ... à prévoir

#### 3.2.5 Décomposition en modules, sous-ensembles

#### 3.2.6 Prévisions de fiabilité

#### 3.2.7 Perspectives d'évolution technologique

# Cahier des charges / Description of Work

- Résumé exécutif
  - une demi page pour aller à l'essentiel avec objectifs attendus
- 1. Description du projet
  - Contexte de travail
    - Environnement, positionnement
  - Motivations
    - Bien fondé du projet, exemples
  - Défis
    - Points difficiles (défi global, puis sous-points)
  - Objectifs
    - Objectifs qui seront évalués en fin de projet
  - Scénario(s)
    - 1 ou plusieurs scénarios expliquant comme les résultats du projet pourront être appliqués sur des cas concrets
  - Critères de succès
    - Comment évaluer le projet vis-à-vis des objectifs



# Cahier des charges / Description of Work

- 2. Etat de l'art
  - Description générale
    - Ecosystème du projet (technologies)
    - Outils utilisés usuellement pour traiter le prob
- 3. Méthodologie et planification
  - Stratégie générale
    - Modèle de cycle de vie (cascade, itération)
    - Phases de mise en œuvre et lien entre les phases
  - Découpage en lots
    - Lots du projet avec un numéro, titre, type de travail, nom du responsable...





# Cahier des charges / Description of Work

- Planification
  - Diagramme de Gantt du projet (cf. fin du cours)
- Livrables associés au projet
  - Liste des livrables, lot associé, nature (document, code, etc.)
- Jalons
  - Point de vérification du projet (typiquement livraison)
- Pilotage et suivi
  - Principes de pilotage (durée des itérations pour un pilotage agile, moyen et personnel pour une approche plus classique...)



**SPECIALISTES**

# Cahier des charges / Description of Work

- 4. Description de la mise en œuvre du projet
    - Interdépendances des lots et tâches
    - Description des lots (objectif, contenu, livrable)
    - Résumé de l'effort
- => dans JIRA



**SPECIALISTES**

- 5. Participants
  - Liste des personnels



**SPECIALISTES**

- 6. Bibliographie, références, acronymes

# Revue de spécification : questions

- Interfaces importantes décrites ?
- Diagrammes clairs ? Texte supplémentaire nécessaire ?
- Grandes fonctionnalités assurées ?
- Contraintes de conception réalistes ?
- Risques technologiques considérés ?
- Critères clairs de validation établis ?
- Y-a-t-il des incohérences, des omissions, des redondances ?
- Le contact avec l'utilisateur est-il terminé / complet ?

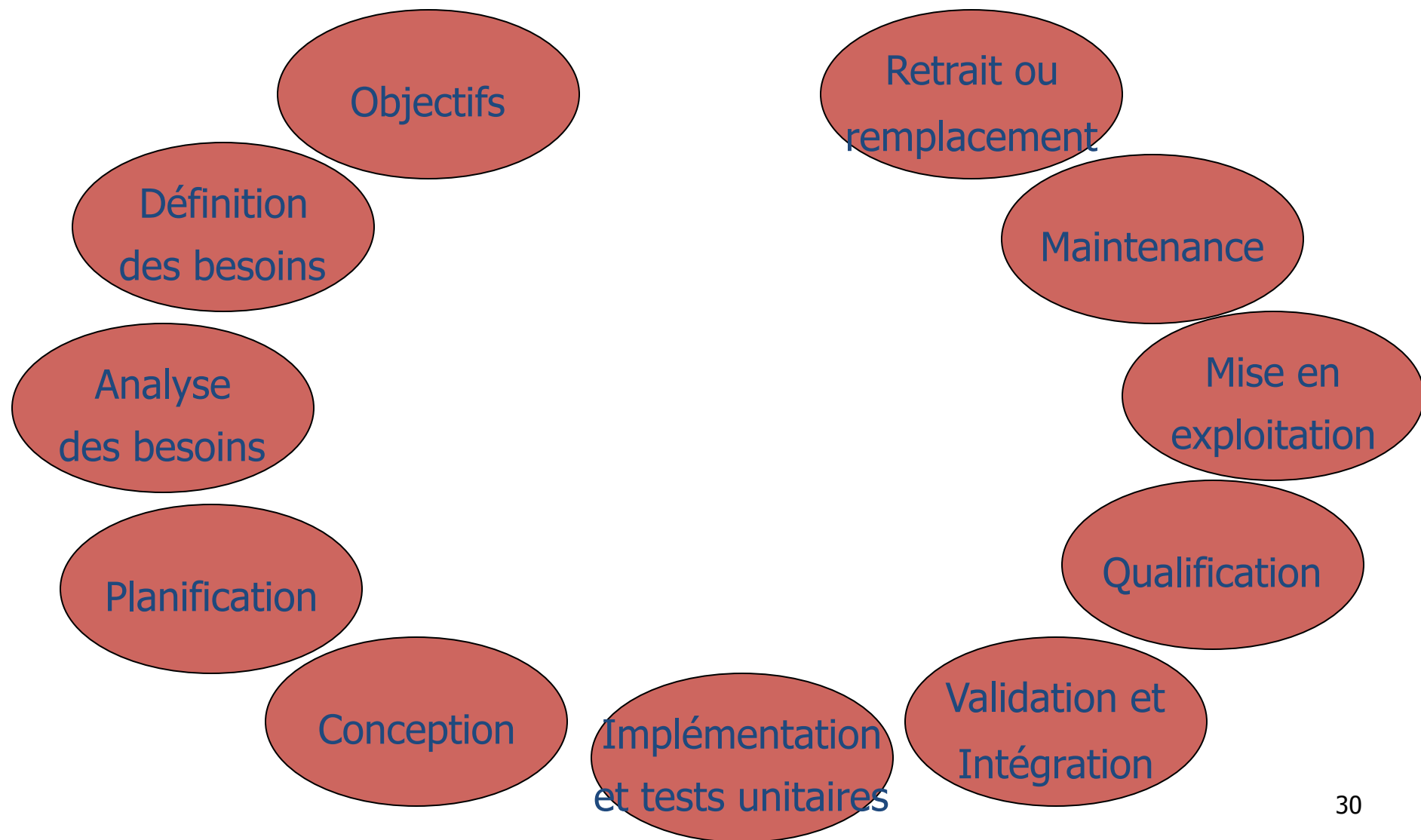
# Cycle de vie du logiciel

- Les phases du cycle de vie
- Les modèles de développement

# Notion de cycle de vie

- Description d'un processus pour :
  - la création d'un produit
  - sa distribution sur un marché
  - son retrait
- Cycle de vie et assurance qualité
  - Validation : le bon produit ?
  - Vérification : le produit correct ?

# Les phases du cycle de vie



# Objectifs

- Fixés par *les donneurs d'ordre*
  - le management
  - ou une (bonne) idée...
- Quelques définitions
  - Clients : ceux qui veulent le produit
  - Utilisateurs : ceux qui vont l'utiliser
  - Développeurs : ceux qui vont le fabriquer

# Définition des besoins

- Un cahier des charges est normalement établi par le **client** en interaction avec utilisateurs et encadrement :
  - description des fonctionnalités attendues
  - contraintes non fonctionnelles (temps de réponse, place mémoire,...)
  - possibilités d'utilisation de *Use Cases*

👉 ***A l'issue de cette phase : cahier des charges***



# Analyse des besoins

- C'est la définition du produit
  - Spécification précise du produit
  - Contraintes de réalisation
- A l'issue de cette phase :
  - Client et fournisseur sont d'accord sur le produit à réaliser (IHM comprise)
  - ➡ **Dossier d'analyse (spécifications fonctionnelles et non fonctionnelles)**
  - ➡ **Ébauche de manuel utilisateur**
  - ➡ **Première version du glossaire du projet**

# Planification

- Découpage du projet en tâches avec enchaînement
  - Affectation à chacune d'une durée et d'un effort
  - Définition des normes qualité à appliquer
  - Choix de la méthode de conception, de test...
  - Dépendances extérieures (matériels, experts...)
- ☞ ***Plan qualité + Plan projet (pour les développeurs)***
- ☞ ***Estimation des coûts réels***
- ☞ ***Devis destiné au client (prix, délais, fournitures)***

# Conception

- Définition de l'architecture du logiciel
- Interfaces entre les différents modules
- Rendre les composants du produits indépendants pour faciliter le développement

👉 ***Dossier de conception***

👉 ***Plan d'intégration***

👉 ***Plans de test***

👉 ***Mise à jour du planning***

# Implémentation et tests unitaires

- Codage et test indépendant de chaque module
- Produits intermédiaires :

☞ ***Modules codés et testés***

☞ ***Documentation de chaque module***

☞ ***Résultats des tests unitaires***

☞ ***Planning mis à jour***

# Validation et Intégration

- Chaque module est intégré avec les autres en suivant le plan d'intégration
- L'ensemble est testé conformément au plan de tests

➡ ***Logiciel testé***

➡ ***Tests de non-régression***

➡ ***Manuel d'installation***

➡ ***Version finale du manuel utilisateur***

# Qualification

- Tests en vraie grandeur, dans des conditions normales d'utilisation
- Tests non-fonctionnels :
  - Tests de charge
  - Tests de tolérance aux pannes
- Parfois Bêta-test

☞ ***Rapports d'anomalie***

- *Déterminant dans la relation client-fournisseur*

# Mise en exploitation

- Livraison finale du produit (packaging)
- Installation chez le client
- Est-ce la fin des problèmes ?

 **AU CONTRAIRE**

 **Ce n'est rien en comparaison de la...**

# Maintenance

- Rapport d'incident (ou anomalie)
- Demande de modification corrective
- Demande d'évolution (avenant au contrat)
- Code et documentation modifiés...
- Nouvelle série de tests :
  - unitaires
  - d'intégration
  - de non-régression



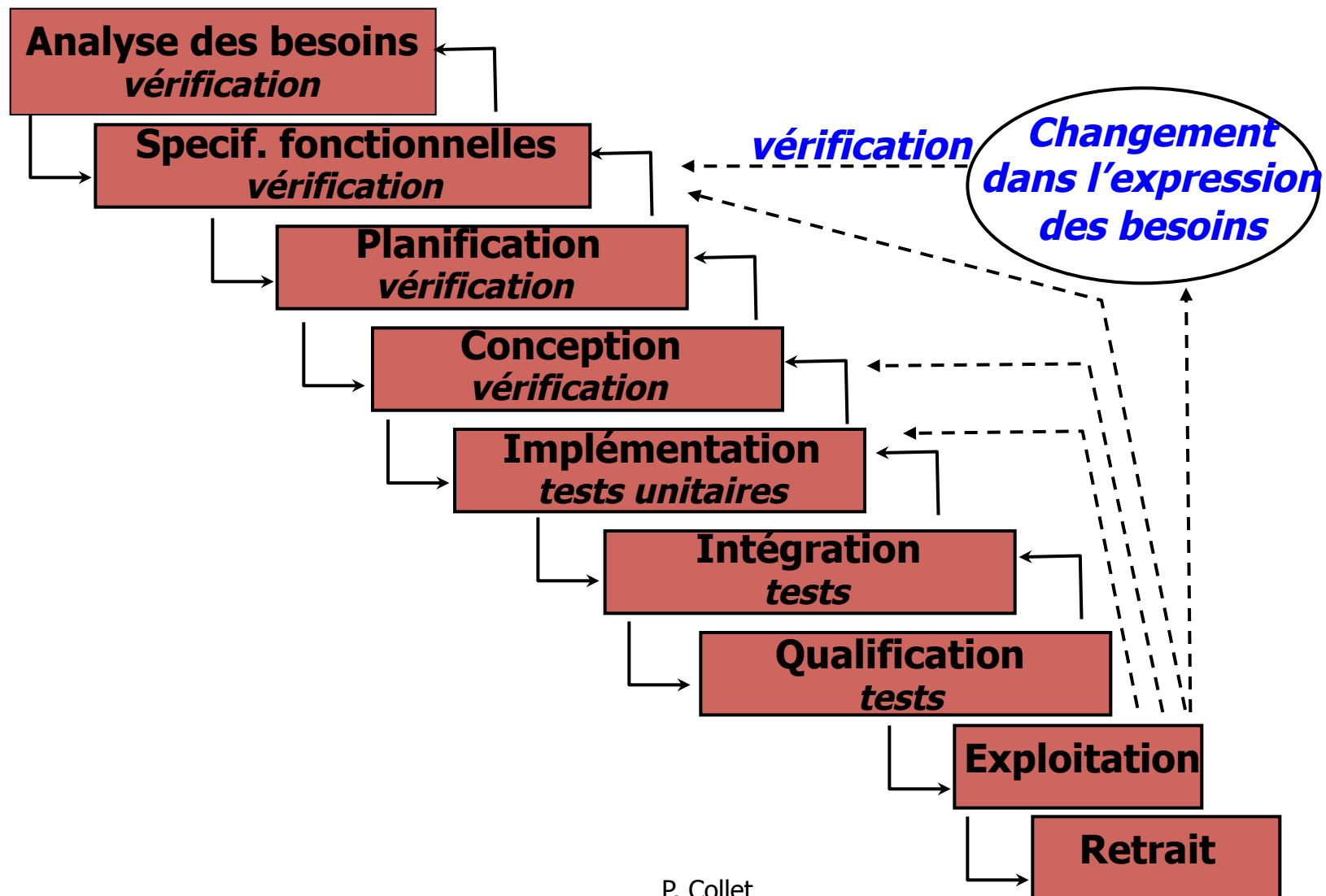
# Exemples de durée de cycle

- SGBD relationnel
  - 1er proto : 5 à 7 ans  
Investissement > 100H An
  - 1er système commercial : 3 à 4 ans  
Investissement > 150H An
  - Maintenance : > 10 ans  
10 à 15 H par an  
nouvelle livraison tous les 6 mois à 1 an
- Langage ADA (1983)
  - Définition et analyse des besoins : 3 ans
  - Compilateur industriel : 3ans  
Investissement > 50H An
  - Maintenance : > 15 ans  
5 à 10 H par an  
livraison tous les 1 ou 2 ans
  - ☞ Nouvelle version : Ada95

# Les approches de développement

- Approche cartésienne, déterministe
  - structurée descendante : *cascade ou V*
- Approche heuristique, par prototypage
  - ascendante : *incrémental ou prototypage*
- Approche objets :
  - aucune organisation **spécifique** n'est vraiment mise en avant

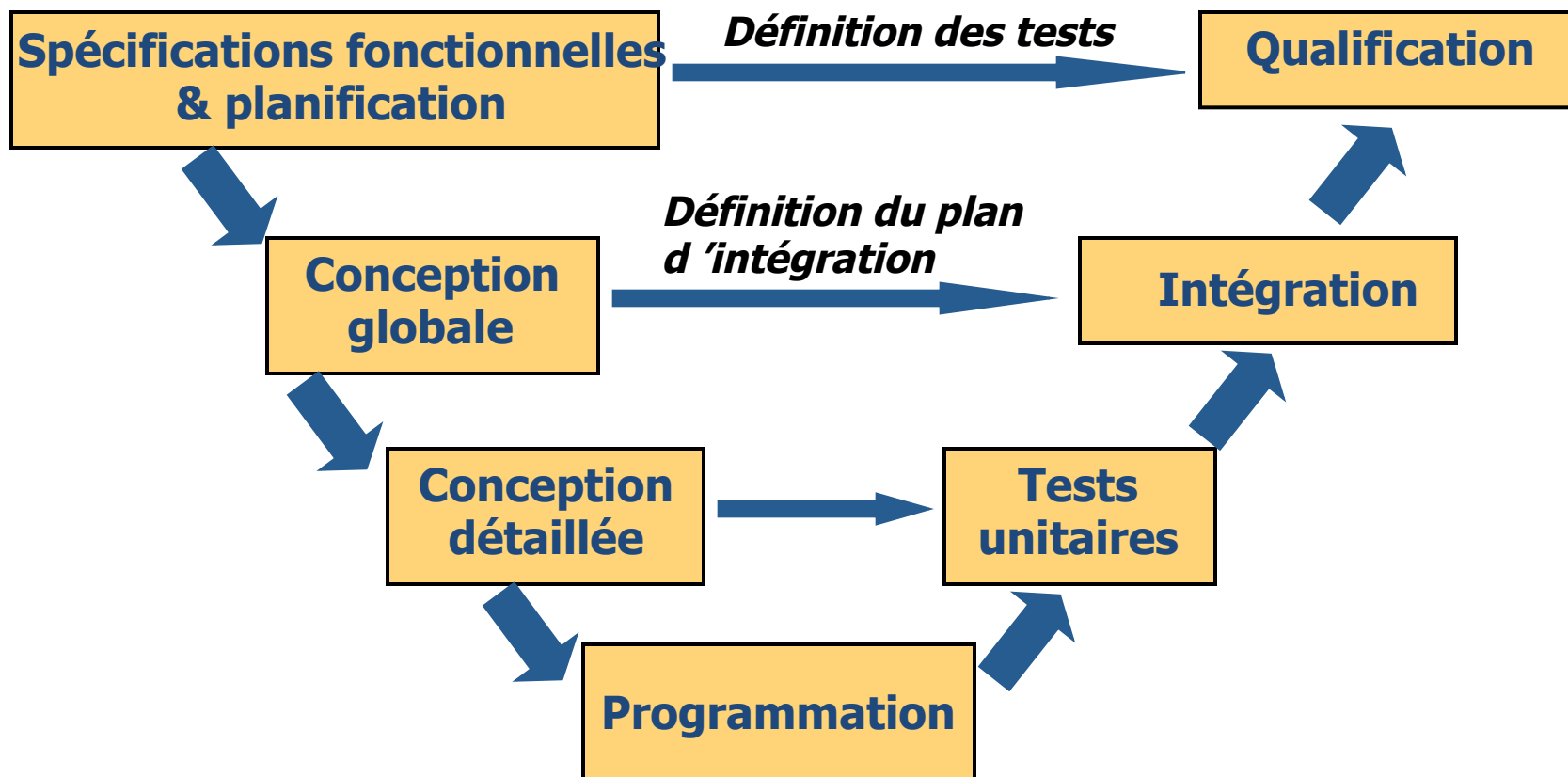
# Modèle en cascade (1970)



# Problèmes du modèle en cascade

- Les vrais projets suivent rarement un développement séquentiel
- Établir tous les besoins au début d'un projet est difficile
- Le produit apparaît tard
- Seulement applicable pour les projets qui sont bien compris et maîtrisés

# Modèle en V

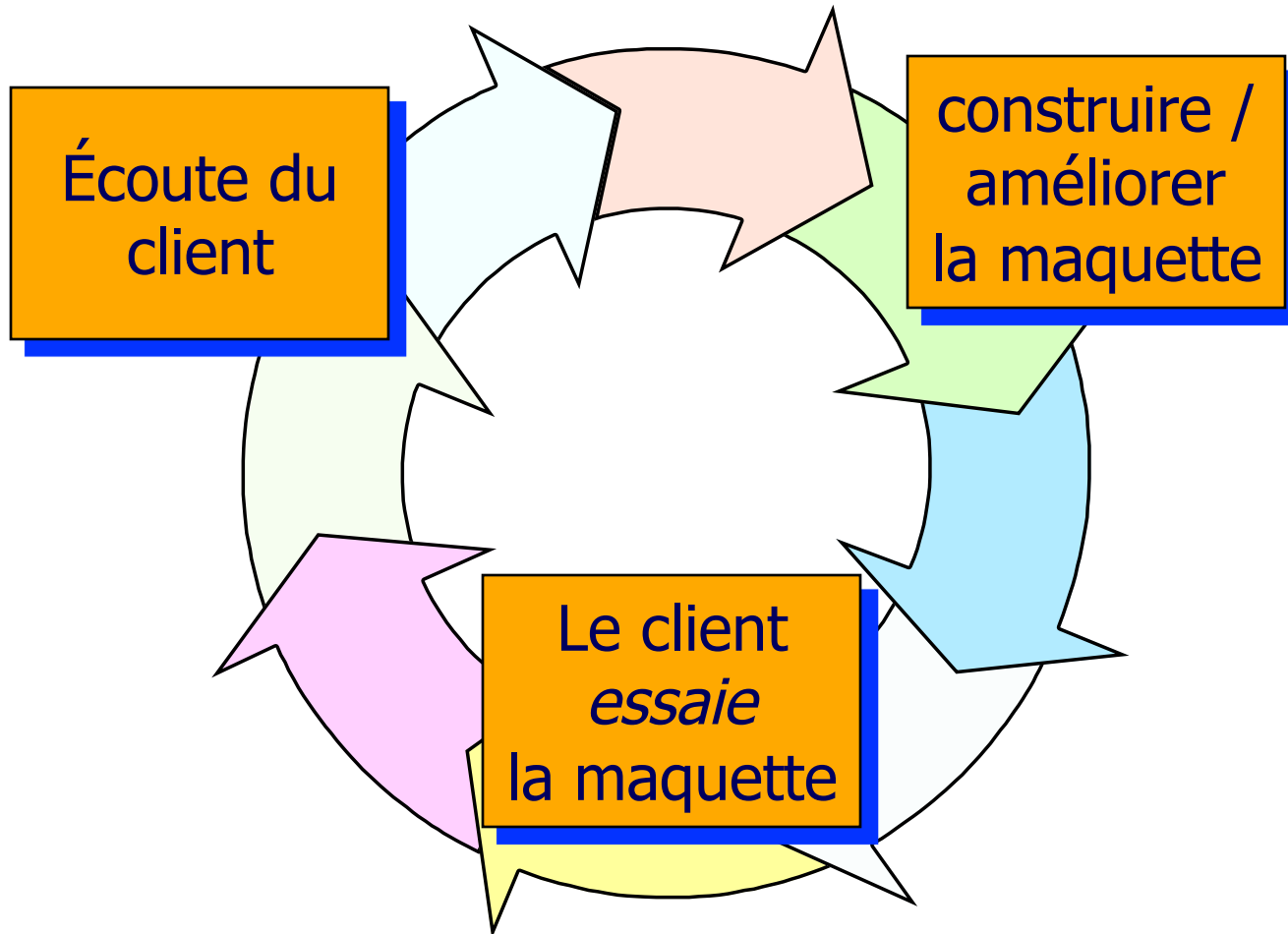


☞ *Gestion des configurations, de projet, plan assurance qualité*

# Comparaison

- Le cycle en V
  - permet une meilleure anticipation
  - évite les retours en arrière
- Mais
  - le cadre de développement est rigide
  - la durée est souvent trop longue
  - le produit apparaît très tard

# Prototypage



# Prototypage, RAD

*RAD : Rapid Application Development*

- Discuter et interagir avec l'utilisateur
- Vérifier l'efficacité réelle d'un algorithme
- Vérifier des choix spécifiques d'IHM
- Souvent utilisé pour identifier les besoins
  - Prototype jetable (moins de risque ?)
- Souvent implémenté par des générateurs de code
  - Prototype évolutif

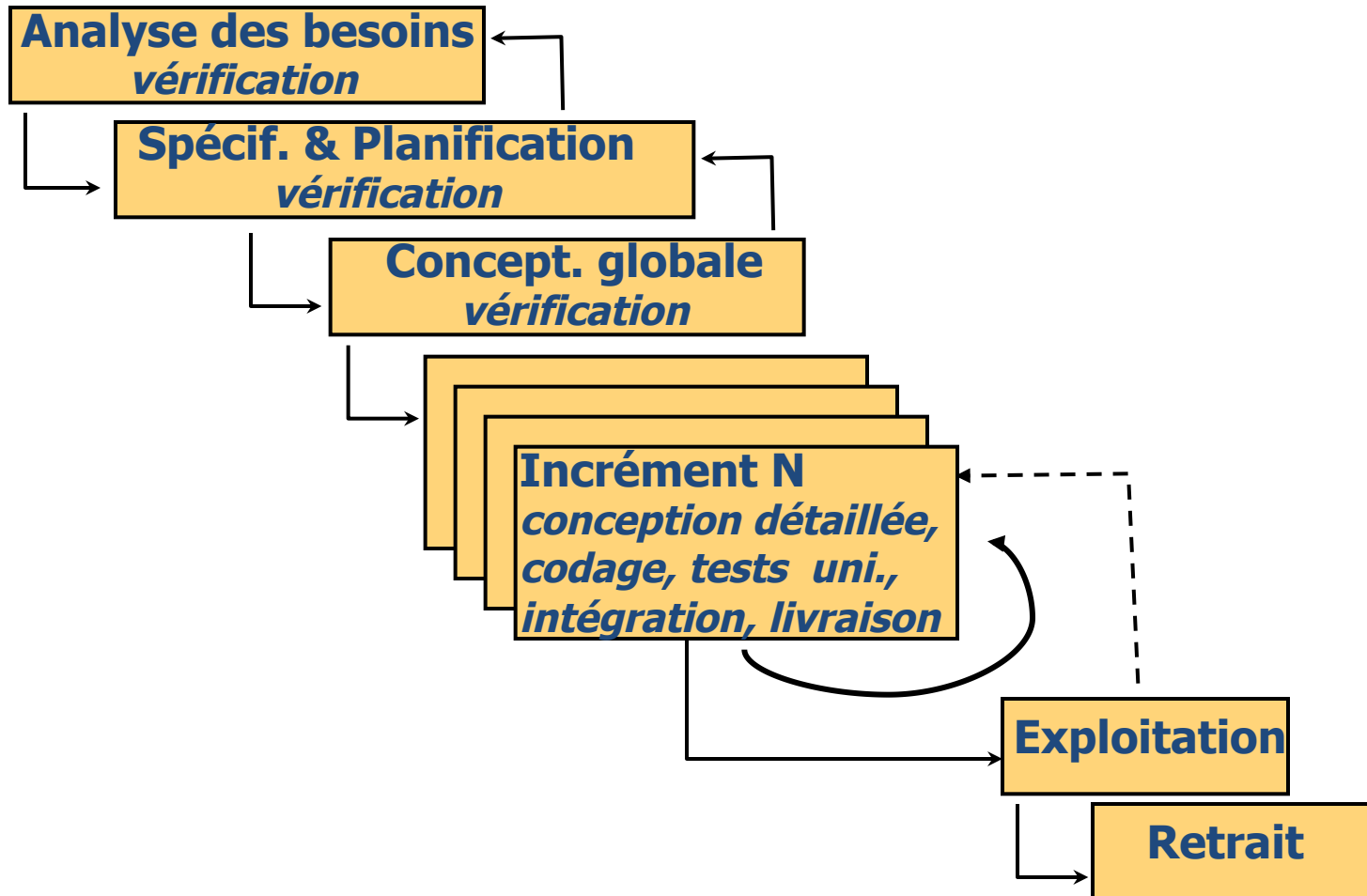


# Prototypage, RAD (suite)

- Mais :
  - Les objectifs sont uniquement généraux
  - Prototyper n'est pas spécifier
  - Les décisions rapides sont rarement de bonnes décisions
  - Le prototype évolutif donne-t-il le produit demandé ?
  - Les générateurs de code produisent-ils du code assez efficace ?

 *Projets petits ou à courte durée de vie*

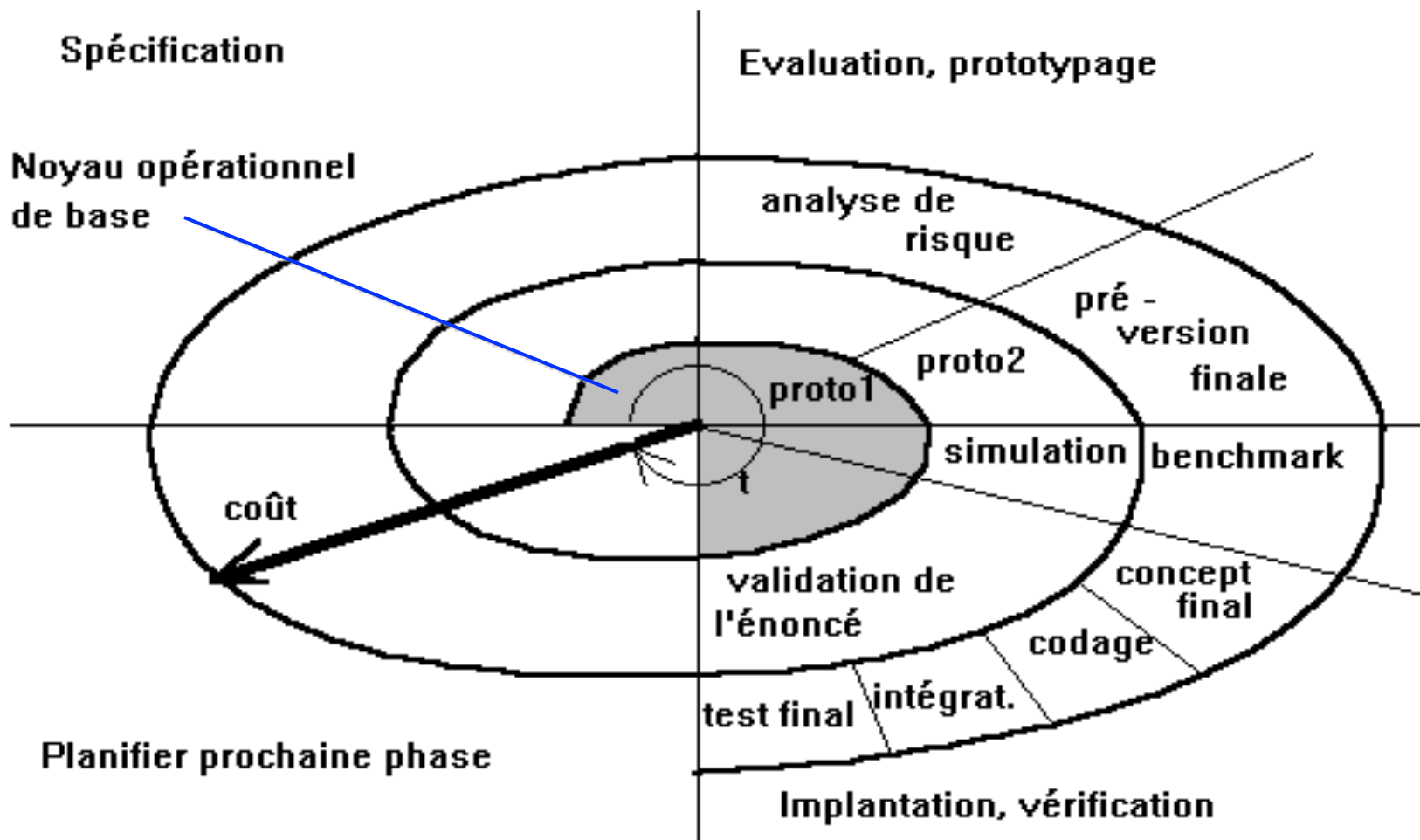
# Modèle incrémental



# Le développement incrémental

- combine des éléments des modèles linéaires et du prototypage
  - produit des incréments *livrables*
  - se concentre sur un produit opérationnel (pas de prototype jetable)
  - peut être utilisé quand il n'y a pas assez de ressources disponibles pour une livraison à temps
- ☞ *Le premier incrément est souvent le noyau*
- ☞ *Les incréments aident à gérer les risques techniques (matériel non disponible)*

# Modèle en spirale (Boehm, 1988)



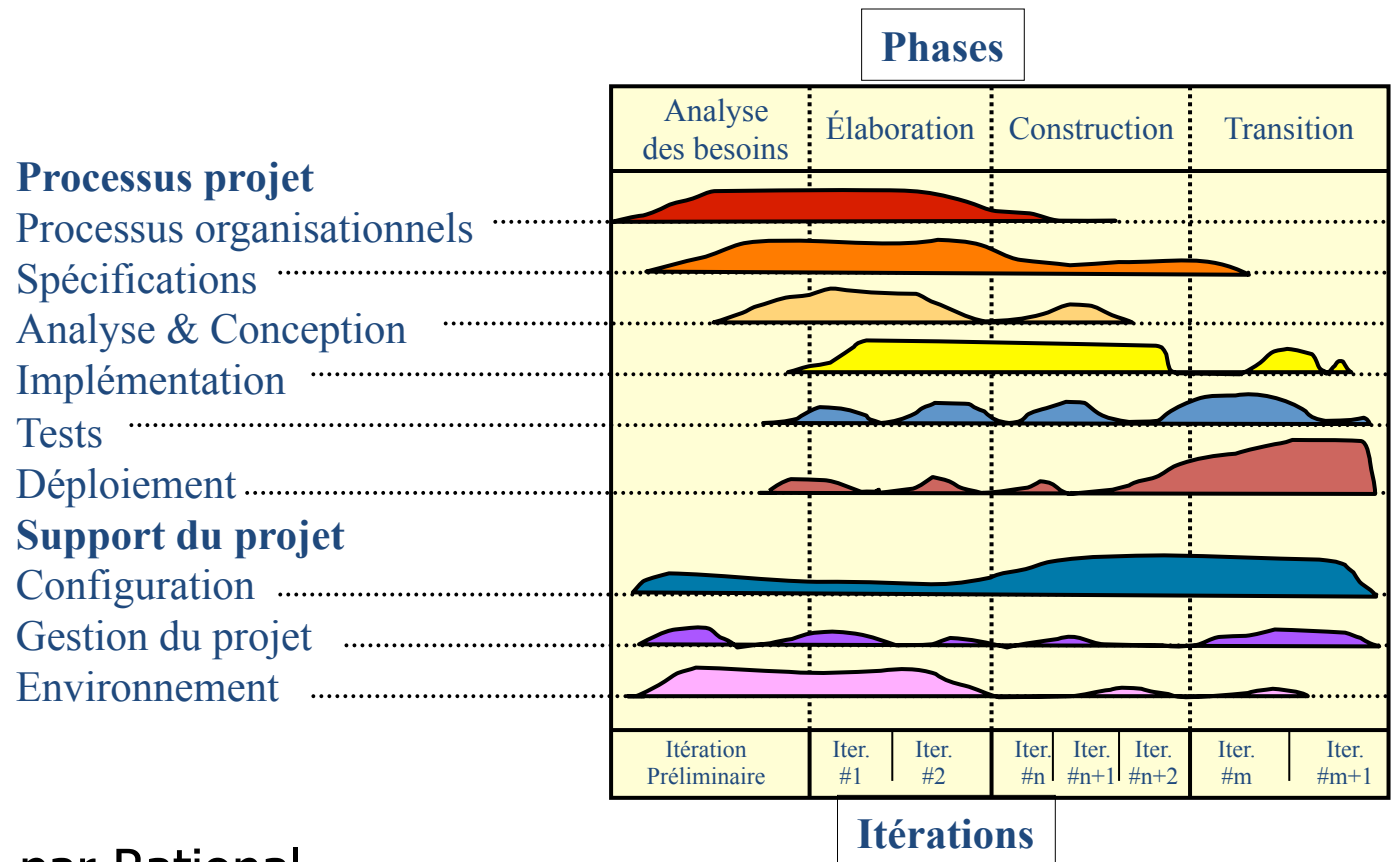
# Modèle en spirale (suite)

- Spécification : communiquer avec le client
- Analyse de risque : évaluation des risques techniques et des risques de gestion
- Implémentation et vérification : construire, tester, installer et fournir un support utilisateur
- Validation: obtenir des *retours*
- Planification : définir les ressources, la répartition dans le temps

# Modèle en spirale (suite)

- Couplage de la nature itérative du prototypage avec les aspects systématiques et contrôlés du modèle en cascade
  - Les premières itérations peuvent être des modèles *sur papier* ou des prototypes
  - Utilisation possible tout au long de la vie du produit
- ☞ *Réduit les risques si bien appliqué*
- ☞ *Les augmentent considérablement si le contrôle faiblit*

# RUP : *Rational Unified Process*

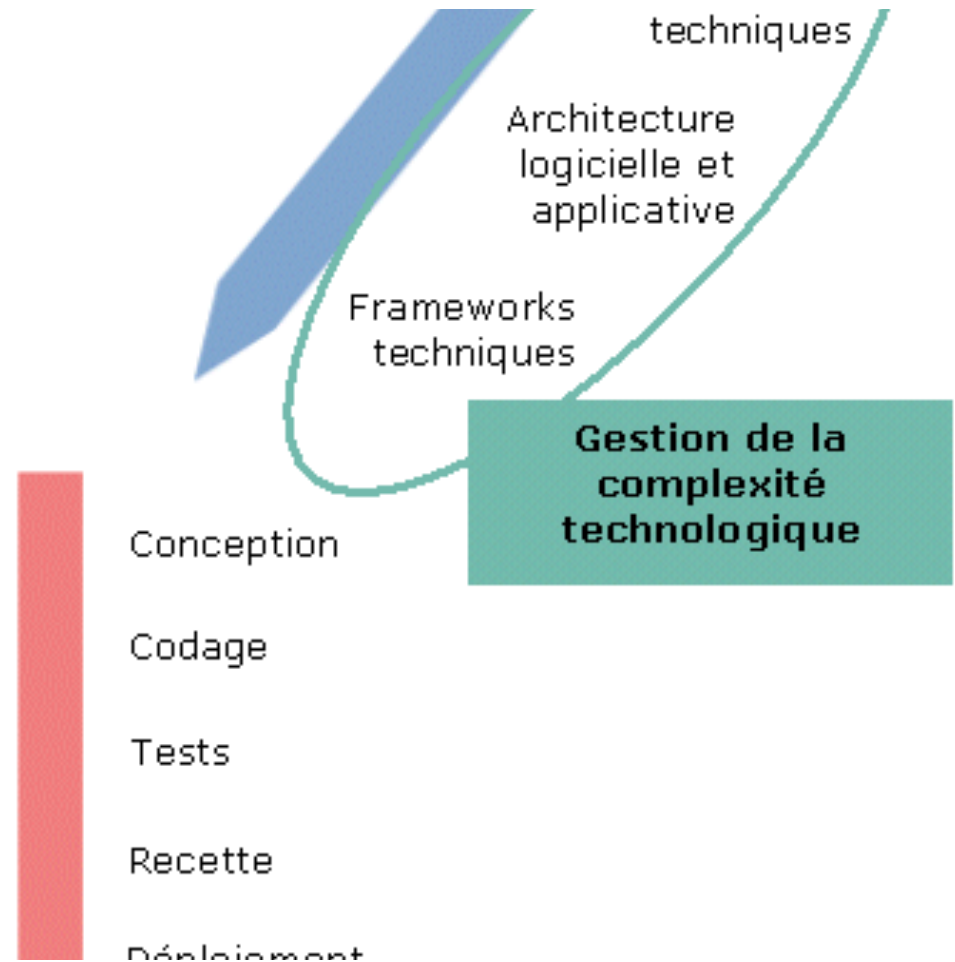


- Promu par Rational
- Le RUP est à la fois une méthodologie et un outil prêt à l'emploi (documents types partagés dans un référentiel Web)
- plutôt pour des projets de plus de 10 personnes

# 2TUP : *Two Track Unified Process*

- S'articule autour de l'architecture
- Propose un cycle de développement en Y
- Détaillé dans « UML en action »
- pour des projets de toutes tailles

**Phase de réalisation**





# eXtreme Programming (XP...)

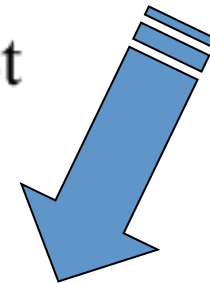
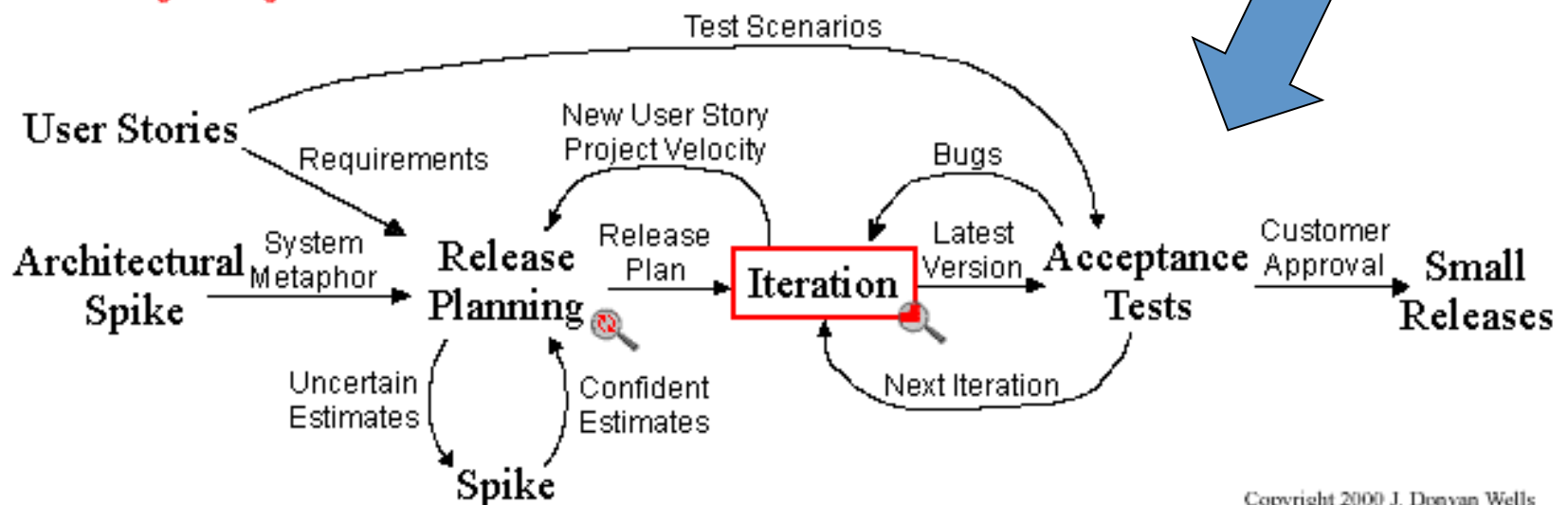
- Ensemble de « Bests Practices » de développement (travail en équipes, transfert de compétences...)
- plutôt pour des projets de moins de 10 personnes

4 Valeurs

- **Communication**
- **Simplicité**
- **Feedback**
- **Courage**



## Extreme Programming Project



# XP => Développement Agile

- Collaboration étroite entre équipe(s) de programmation et experts métier
  - Communication orale, pas écrite
  - Livraison fréquente de fonctionnalités déployables et utilisables (= qui apportent une valeur ajoutée)
  - Equipe auto-organisée et soudée
- *Test-Driven Development*
  - Ecrire les tests avant le code...

# Manifeste Agile : 12 principes

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile process harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face to face conversation.

# Manifeste Agile : 12 principes

7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity – the art of maximizing the amount of work not done – is essential
11. The best architectures, requirements, and designs emerge from self-organizing teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

# Scrum : principes

- Isolement de l'équipe de développement
  - l'équipe est isolée de toute influence extérieure qui pourrait lui nuire. Seules l'information et les tâches reliées au projet lui parviennent : pas d'évolution des besoins dans chaque sprint.
- Développement progressif
  - afin de forcer l'équipe à progresser, elle doit livrer une solution tous les 30 jours. Durant cette période de développement l'équipe se doit de livrer une série de fonctionnalités qui devront être opérationnelles à la fin des 30 jours.
- Pouvoir à l'équipe
  - l'équipe reçoit les pleins pouvoirs pour réaliser les fonctionnalités. C'est elle qui détient la responsabilité de décider comment atteindre ses objectifs. Sa seule contrainte est de livrer une solution qui convienne au client dans un délai de 30 jours.
- Contrôle du travail
  - le travail est contrôlé quotidiennement pour savoir si tout va bien pour les membres de l'équipe et à la fin des 30 jours de développement pour savoir si la solution répond au besoin du client.

# Scrum : rôles et pratiques

- Scrum Master
  - expert de l'application de Scrum
- Product owner
  - responsable officiel du projet
- Scrum Team
  - équipe projet.
- Customer
  - participe aux réunions liées aux fonctionnalités
- Management
  - prend les décisions
- Product Backlog
  - état courant des tâches à accomplir
- Effort Estimation
  - permanente, sur les entrées du backlog
- Sprint
  - itération de 30 jours
- Sprint Planning Meeting
  - réunion de décision des objectifs du prochain sprint et de la manière de les implémenter
- Sprint Backlog
  - Product Backlog limité au sprint en cours
- Daily Scrum meeting
  - ce qui a été fait, ce qui reste à faire, les problèmes
- Sprint Review Meeting
  - présentation des résultats du sprint

# Comparaison des 3 processus dans le vent

## Points forts

## Points faibles

	Points forts	Points faibles
<b>RUP</b>	<ul style="list-style-type: none"> <li>■ Itératif</li> <li>■ Spécifie le dialogue entre les différents intervenants du projet : les livrables, les plannings, les prototypes...</li> <li>■ Propose des modèles de documents, et des canevas pour des projets types</li> </ul>	<ul style="list-style-type: none"> <li>■ Coûteux à personnaliser</li> <li>■ Très axé processus, au détriment du développement : peu de place pour le code et la technologie</li> </ul>
<b>XP</b>	<ul style="list-style-type: none"> <li>■ Itératif</li> <li>■ Simple à mettre en œuvre</li> <li>■ Fait une large place aux aspects techniques : prototypes, règles de développement, tests...</li> <li>■ Innovant: programmation en duo...</li> </ul>	<ul style="list-style-type: none"> <li>■ Ne couvre pas les phases en amont et en aval au développement : capture des besoins, support, maintenance, tests d'intégration...</li> <li>■ Élude la phase d'analyse, si bien qu'on peut dépenser son énergie à faire et défaire</li> <li>■ Assez flou dans sa mise en œuvre: quels intervenants, quels livrables ?</li> </ul>
<b>2TUP</b>	<ul style="list-style-type: none"> <li>■ Itératif</li> <li>■ Fait une large place à la technologie et à la gestion du risque</li> <li>■ Définit les profils des intervenants, les livrables, les plannings, les prototypes</li> </ul>	<ul style="list-style-type: none"> <li>■ Plutôt superficiel sur les phases situées en amont et en aval du développement : capture des besoins, support, maintenance, gestion du changement...</li> <li>■ Ne propose pas de documents types</li> </ul>

# Les différents types de projet

<b>Durée</b>	<b>Personnes</b>	<b>Budget</b>	<b>Approche</b>
< à 1 an	1	< 100 K€	Documentation a posteriori Validation par le développeur Vie limitée
Env. 1 an	1 à 5	< 300 à 500 K€	Plusieurs phases (dont conception) Planning, réunions d'avancement Contrôle qualité interne et gestion de versions Prototypage
1 à 2 ans	6 à 15	< 5 M€	Etudes préliminaires et cycle en spirale Documents de suivi et d'anomalie, inspections Gestion de configurations Plans de validation et d'intégration
2 ans et plus	16 et plus	> 5 M€	Procédures de communication Recettes intermédiaires Contrôle qualité permanent Gestion des sous-projets et de la sous-traitance Tests de non-régression Effort de synthèse et base historique