Noël NOVELLI ; Université de la Méditerranée ; LIF et Département d'Informatique – Case 901 ; 163 avenue de Luminy – 13 288 MARSEILLE cedex 9

Documentation

- La documentation a pour but de laisser des traces du travail réalisé
- Pourquoi doit-on laisser des traces ?
  Est-ce important ?

Ce support est une réutilisation de diapositives des supports précédents

Documentation

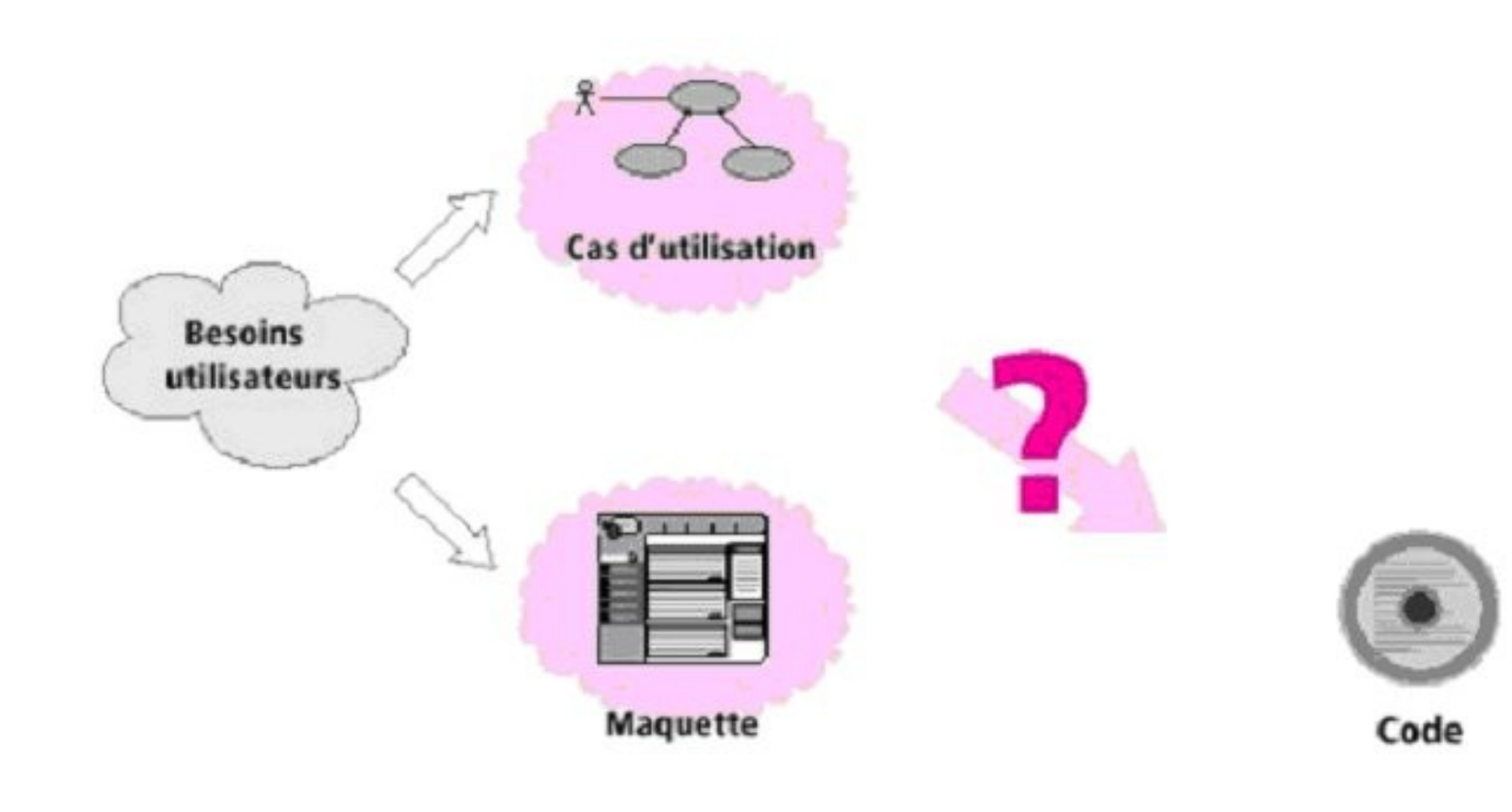
- Tous les documents doivent contenir un en-tête (le même modèle pour tous les documents)
- Dossiers d'analyse des besoins, de spécification, d'analyse du système, de conception, d'analyse technique, de développement (en plus des commentaires), de déploiement, de maintenance, d'utilisation, de test d'application et d'intégration, d'évolution.
- Dossier d'analyse financière
- Dossier de suivi du projet :
  - Planning, acteurs, activités, les artefacts, les workflows, les lotissements, les livrables, les tableaux de bord, CR de réunions interne et externe, les itérations.
- Garder à l'esprit l'enjeux du projet et les risques !

Côté client

Analyse des besoins

- Étape préalable si le client n'a qu'une idée peu précise du système à réaliser
  - Étude informelle des fonctionnalités (externes) du système sans considération technique : point de vue métier / utilisateur
  - Dialogue fournisseur / client en termes intelligibles pour ce dernier : l'aider à formaliser le problème à résoudre
  - Produire un document textuel avec schémas…
- Conduit généralement à la définition du **cahier des charges**

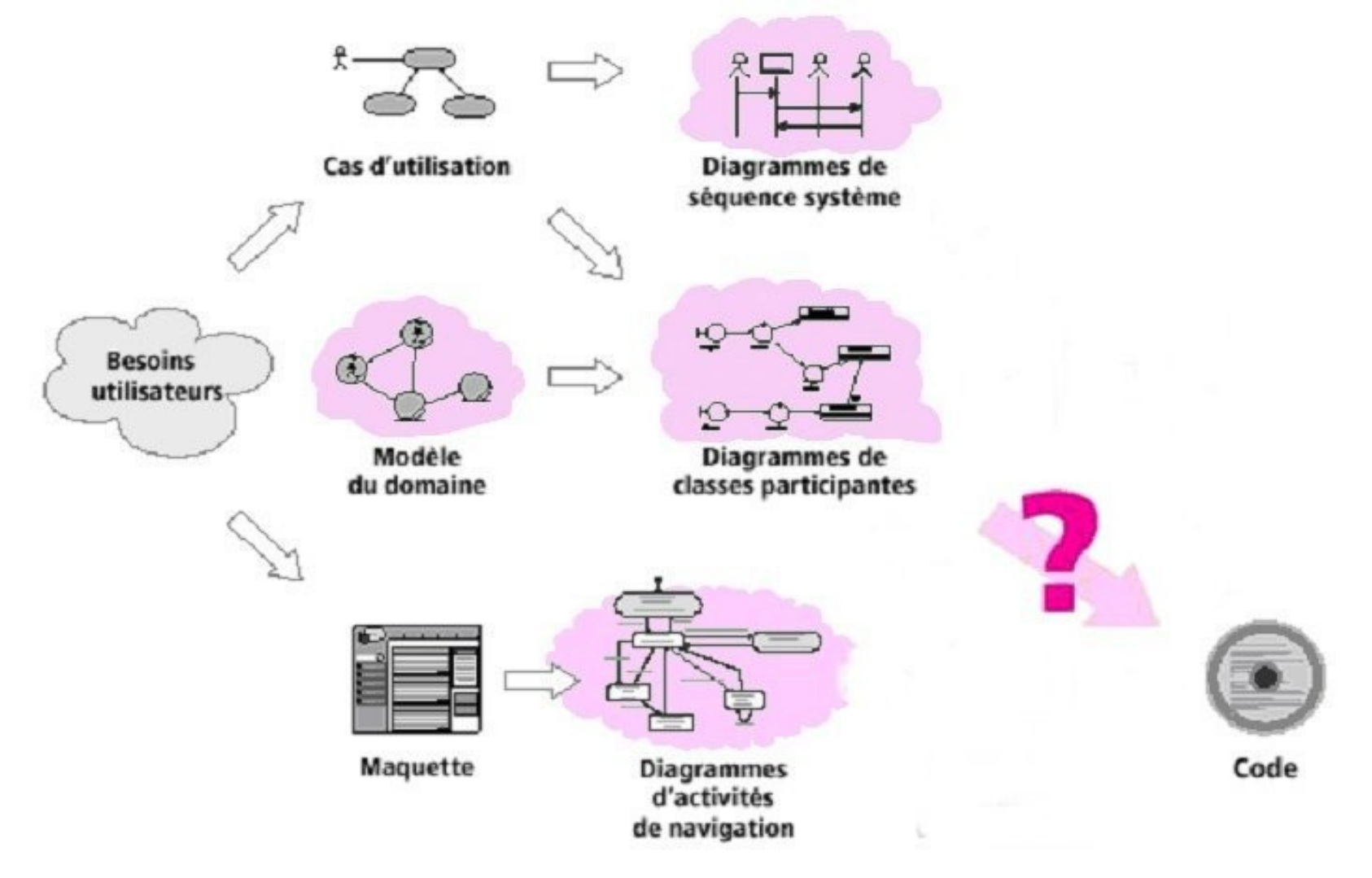
Spécifications



Spécifications

- **Ce que doit faire le système** (côté client)
  - Document précis spécifiant les **fonctionnalités** attendues
  - **Base du contrat commercial avec le client**
  - Document facile à comprendre par le client / utilisateur
  - Exemple : *définition de la frontière du système, description des fonctionnalités du système avec scénarios, interactions : enchaînements d'écran (cas d'utilisation de Jacobson)*
- **Les spécifications ne sont jamais complètes ni définitives**
  - Evolution du domaine, besoins supplémentaires…

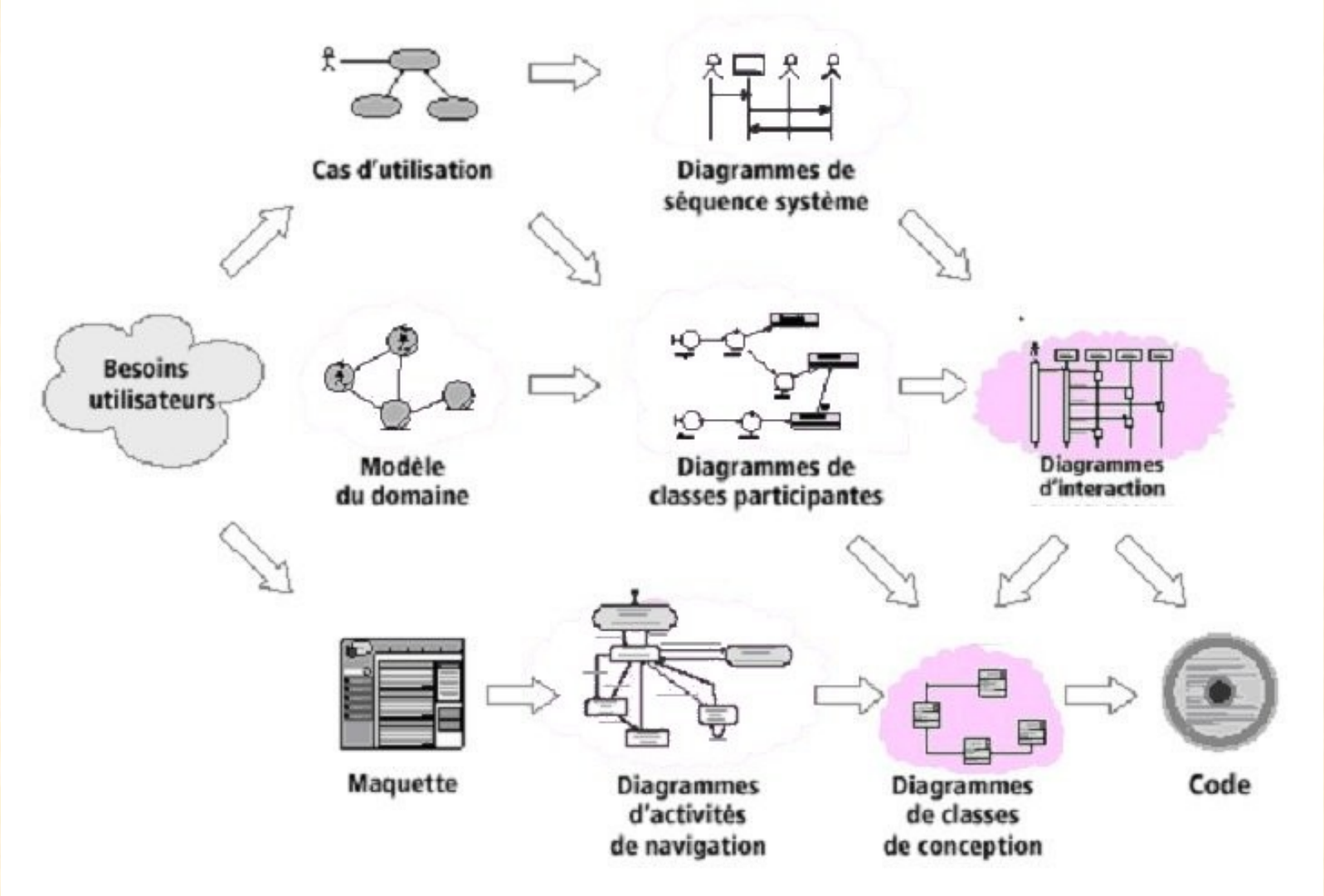
Analyse


Cas d'utilisation
Diagrammes de séquence système
Besoins utilisateurs
Modèle du domaine
Diagrammes de classes participantes
?
Maquette
Diagrammes d'activités de navigation
Code

Analyse du système

- **« Quoi faire » : comprendre et modéliser le métier**
  - Réflexion métier hors de toute considération technique
  - Reste un support de discussion avec le client / utilisateur
  - Premier modèle du système (niveau métier) Identifier les éléments intervenants (hors acteurs) et dans le système : fonctionnalités, structures et relations, états par lesquels ils passent suivant certains événements (diagramme de classes, de collaboration)
- **L'analyse n'est jamais complète mais elle doit être juste**

Conception


Cas d'utilisation
Diagrammes de séquence système
Besoins utilisateurs
Modèle du domaine
Diagrammes de classes participantes
Diagrammes d'interaction
Maquette
Diagrammes d'activités de navigation
Diagrammes de classes de conception
Code

Conception

- Comment faire le système : choix **techniques**
  - Choix d'une architecture technique (matériel, logiciel) suivant certaines priorités (facteurs qualités : robustesse, efficacité, portabilité…)
  - Expertise informatique : **hors** compréhension du client
  - **Modèle de l'architecture logicielle du système** Décomposition en sous-systèmes : application (interfaces), domaine (métier) et infrastructure (implémentation)
- Permet la définition des phases d'implémentation, de validation et de maintenance

Implémentation

- **Souvent trop de temps consacré au codage** au détriment des phases d'analyse et de conception : mauvaise pratique parfois très coûteuse…
- Savoir user de la réutilisation de composants, voire d'outils de génération de code (mise en place automatique du squelette du code à partir du modèle système)

⇒l'activité de développement sera de plus en plus tournée vers la
réutilisation de composants existants

Validation

- Tests de vérification
  - Vérification de la robustesse et cohérence du système en particulier dans le cas d'exception
  - Testeur ≠ concepteur ou programmeur
  - Logiciels de tests : toute ligne de code doit être testée !
- Recette
  - Validation client : accord avec les besoins
  - Une fois les tests de vérification satisfaits
  - Planification dès les spécifications : cahier de recettes
  - Activité souvent sous-estimée…

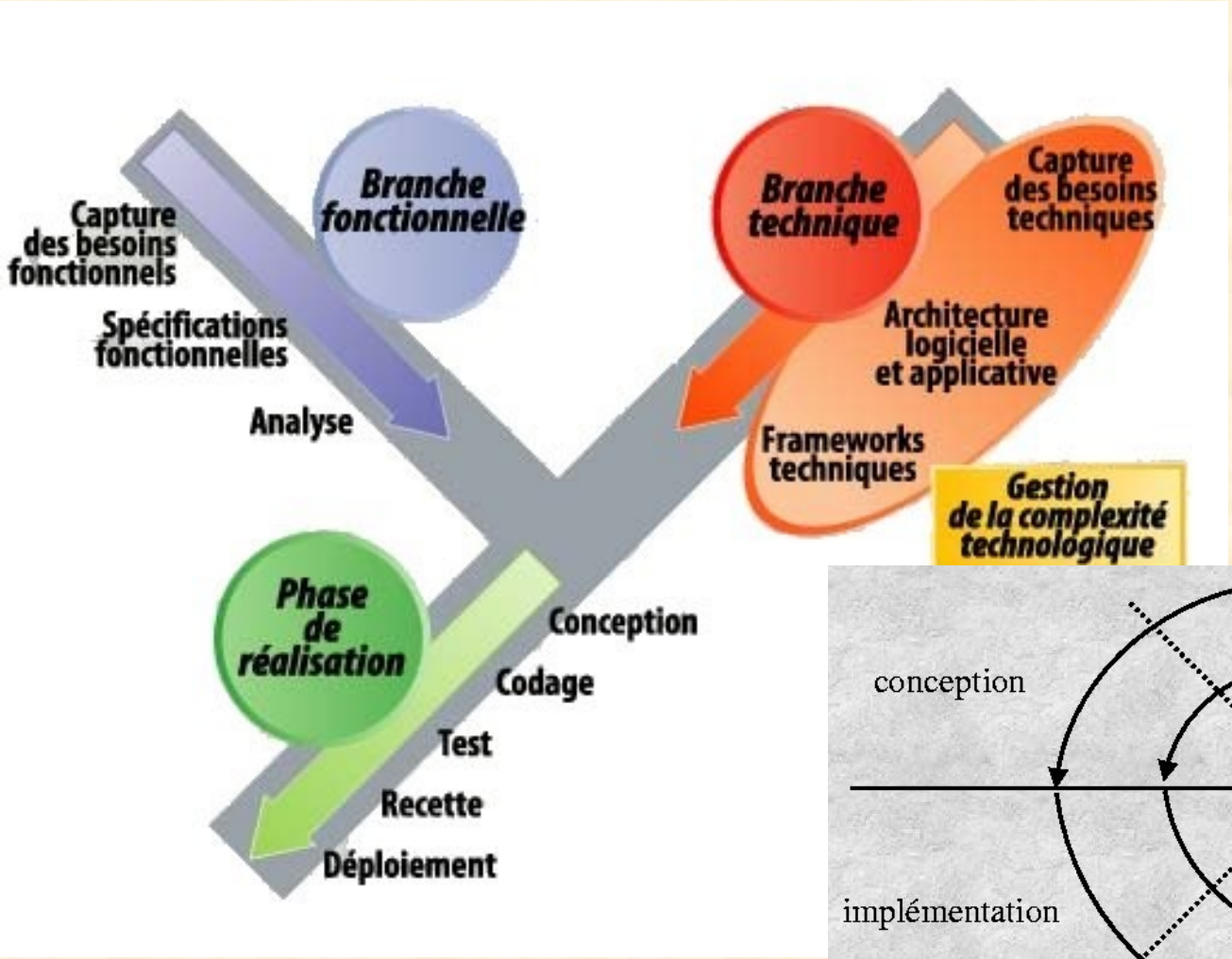
Maintenance

- Deux techniques de maintenance
  - Correction des erreurs du système
  - Demande d'évolution (modification de l'environnement technique, nouvelle fonctionnalité)
- Facteurs de qualité essentiels
  - Corrections : robustesse
  - Évolutions : modifiabilité, portabilité
- Étape longue, critique et coûteuse
  - 80 % de l'effort de certaines entreprises (pb de pratiques ?)

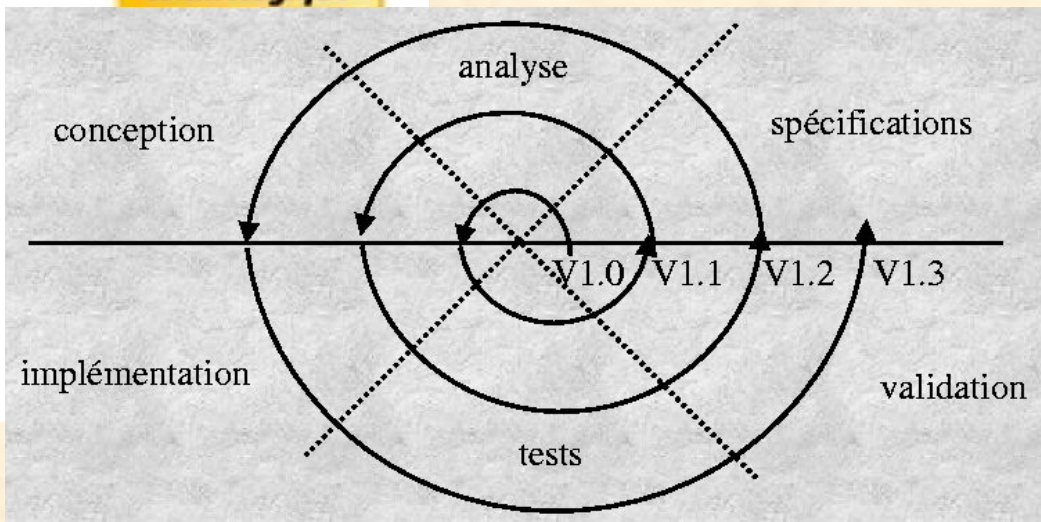
Lotissement et livrable

- Lotissement
  - Le fait de diviser l'application en plusieurs applications plus petites ce qui divise le traitement de la complexité de l'application globale
    - *Ceci s'applique donc à **toutes les phases** de réalisation du projet*
    - *Le développement **itératif** permet la validation régulière par les utilisateurs*
    - *Un lot ou ensemble de lots peut constituer un livrable*
- Livrable
  - Regroupement **d'artefacts** remis au client. Ceci est contractuel donc prévu !
    Exemple : dossier d'analyse fonctionnelle partiel et le codage correspondant avec tout ce qui va avec.
  - Prototypage
    - *Livraison d'un exécutable permettant une validation concrète*

Rappel



Effet tunnel ?



XP

- Principes d'XP :
  - un représentant du **client est intégré à plein temps** dans l'équipe de développement afin d'atteindre une réactivité optimale, aussi bien pour la définition du besoin que pour la validation des livraisons,
  - la programmation se fait en **binôme** ce qui permet l'appropriation de l'ensemble du code par l'équipe de développement et le partage de l'expertise,
  - l'écriture des **tests unitaires se fait avant l'étape de codage**, ce qui améliore leur efficacité et donc la qualité du code,
  - **l'intégration est continue** dès le début du projet, ce qui supprime les risques associés à cette discipline,
  - des **livraisons fréquentes** permettent d'accélérer la convergence du produit par rapport aux besoins des utilisateurs.