

# GPO2

## Risques

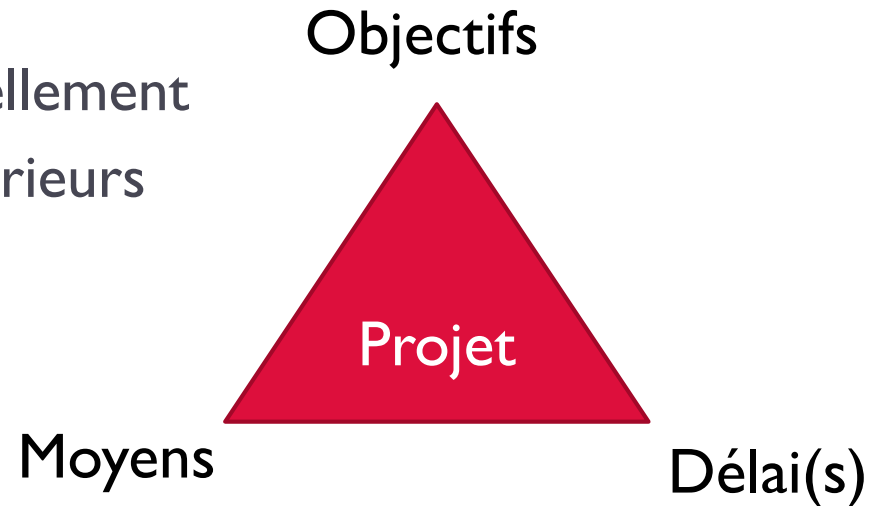
Jean-Marie Mottu  
IUT de Nantes – Département Informatique

# Le risque d'échecs dans l'accomplissement du projet

---

## ► Déséquilibre du triangle :

- Hors délais
- Objectifs non atteints ou partiellement
- Coûts (internes/externes) supérieurs



# Prévoir les risques dans l'accomplissement des tâches de développement d'une fonction

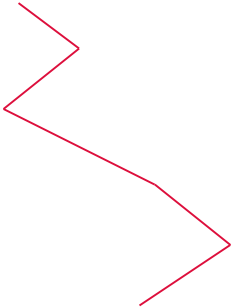
---

- ▶ **Facteurs de risques à évaluer :**
  - ▶ Taille de la fonction
  - ▶ Difficulté technique
  - ▶ Degré d'intégration
  - ▶ Configuration organisationnelle
  - ▶ Changement
  - ▶ Instabilité de l'équipe de projet
- ▶ **(se décline pour le projet et chaque fonction)**

# Prévoir les risques dans l'accomplissement des tâches de développement d'une fonction

---

## ► Permet de créer un profil de risque

Nature du risque	Degré du risque pour la fonction					
	0	1	2	3	4	5
Taille de la fonction Difficulté technique Degré d'intégration Configuration organisationnelle Changement Instabilité de l'équipe de projet						

- Un profil de risque faible est une ligne verticale à gauche (respectivement risque fort à droite)

# Prévoir les risques dans l'accomplissement des tâches de développement d'une fonction

---

## ► SWOT

• Strongness	• Weakness
• Opportunities	• Threats

# Le risque de dysfonctionnement du système développé

---

- ▶ **Le système livré fonctionne mal**
  - ▶ Incidence humaine, financière : cf. premier CM
  - ▶ Essentiellement des risques de pannes imprévues
    - ▶ Par exemple :
      - Risque de panne d'un système de vente : perte de CA
      - Risque de panne d'instrument de vol : crash aérien
- ▶ **Ou des comportements non prévus :**
  - ▶ Les programmes automatiques des bourses créent régulièrement des emballements des cours

# Prévoir les risques de défaillance(s) de la fonction livrée

---

Pour chaque fonction,

- ▶ lister :
  - ▶ Défaillances possibles
  - ▶ Causes potentielles
  - ▶ Effets potentiels des défaillances
  - ▶ Comment détecter ces défaillances
  
- ▶ évaluer :
  - ▶ Gravité
  - ▶ Fréquence

# Prévoir les risques de défaillance(s) de la fonction livrée

---

## ▶ Evaluer :

### ▶ Gravité

#### ▶ Sur une échelle de 1 à 4 :

□ Mineure, Majeure, Critique, Catastrophique

### ▶ Fréquence

#### ▶ Probabilité que la cause survienne

#### ▶ Sur une échelle de 1 à 4 :

□ Rare, modérée, élevée, très élevée

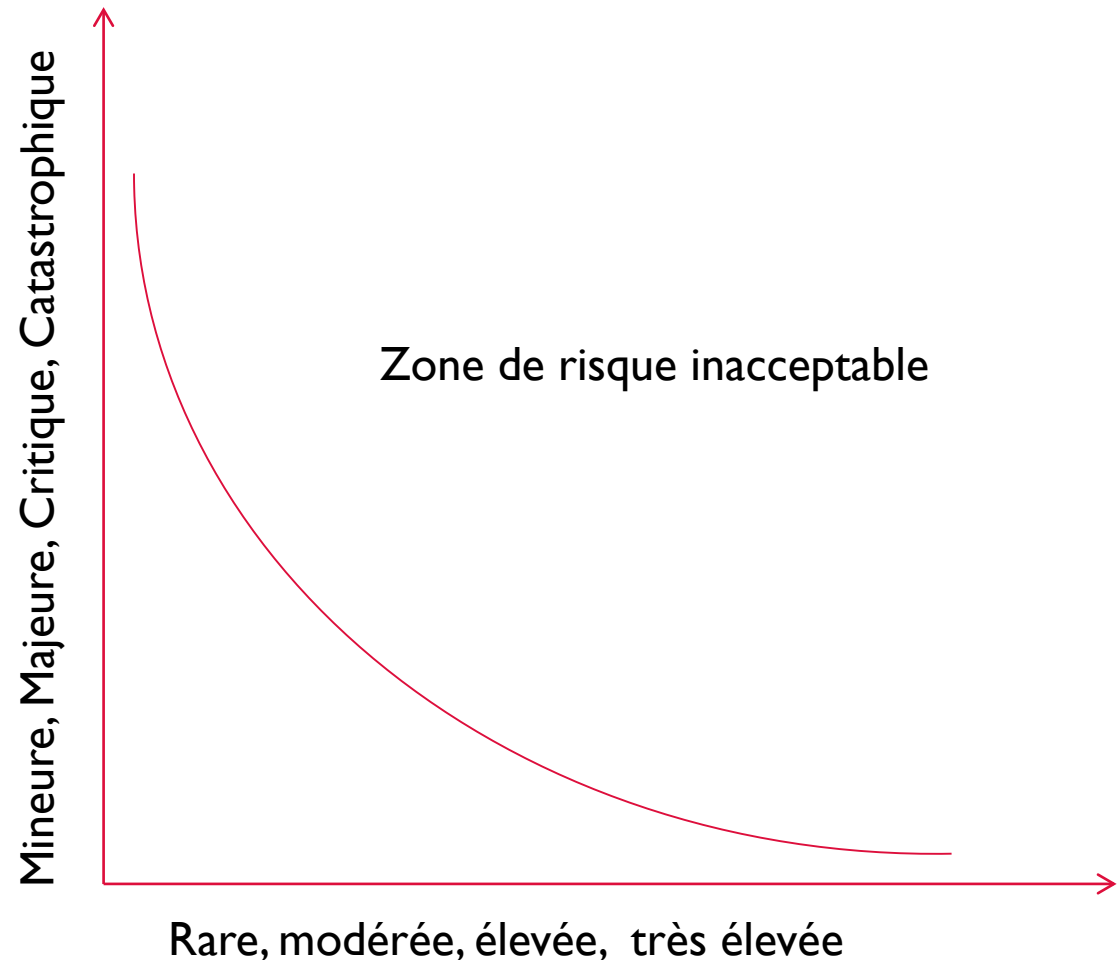


# Prévoir les risques de défaillance(s) de la fonction livrée

---

## ► Evaluer :

- Gravité
- Fréquence
- Chaque projet a une courbe variable, couramment :



# Prévoir les risques de défaillance(s) de la fonction livrée

---

- ▶ Pour chaque projet, on crée une/des matrice(s) de criticité
  - ▶ Chaque défaillance d'une fonction est placée dans une cellule
  - ▶ On grise les cellules de la zone de risque inacceptable

Fréquence Gravité	Rare,	modérée	élevée	très élevée
Mineure				
Majeure				
Critique				
Catastrophique				

# Prévoir les risques de défaillance(s) de la fonction livrée

## ► Evaluer :

- Gravité
- Fréquence

- Toutefois, cette évaluation est pondérée par un indice de non-détection :
  - Note de 1 à 10 (10 risque de non-détection)
  - Il peut amoindrir un risque

Fréquence Gravité	Rare,	modérée	élevée	très élevée
Mineure				
Majeure				
Critique				
Catastrophique	FI (non- détection 2/10)			

# Maîtriser les risques à différentes portées

---

- ▶ Ainsi il faut pour chaque fonction développée :
  - ▶ Prévoir les risques dans l'accomplissement des tâches de développement de la fonction
  - ▶ Prévoir les risques de défaillance(s) de la fonction livrée

# Démarche projet

Jean-Marie Mottu  
IUT de Nantes – Département Informatique

# Démarche Projet

## Etape 0

---

- ▶ **C'est l'émergence d'une idée qui fait naître le projet**
  - ▶ Identifier un problème
  - ▶ Trouver l'inspiration dans l'existant
  - ▶ Penser à une innovation

# Démarche Projet

## Etape 0

---

- ▶ **Analyse qualitative : QQOQCC - P**
  - ▶ Qui ?
  - ▶ Quoi ?
  - ▶ Où ?
  - ▶ Quand ?
  - ▶ Comment ?
  - ▶ Combien ?
- ▶ et Pourquoi tout ça ?

# Démarche Projet

## 6 étapes de l'AFNOR

---

1. Etudes préliminaires
2. Conception
3. Définition
4. Construction
5. Mise en route
6. Transfert à l'exploitation



# Démarche Projet

## 6 étapes de l'AFNOR

---

### 1. Etudes préliminaires

- ▶ Quels sont le problème, la solution, l'idée ?
- ▶ Qu'est-ce qui existe (comment faire mieux) ?
- ▶ Est-ce faisable ?

### 2. Conception

- ▶ Elaboration du cahier des charges fonctionnel
- ▶ Analyse de tous les détails qui formeront le produit et des conditions pour pouvoir le produire
- ▶ Permet de décider du lancement de l'exécution du projet

### 3. Définition

### 4. Construction

### 5. Mise en route

### 6. Transfert à l'exploitation

---

# Démarche Projet

## 6 étapes de l'AFNOR

---

1. Etudes préliminaires
2. Conception
3. Définition
  - ▶ Recherche et choix des solutions
4. Construction
  - ▶ Mise en œuvre jusqu'à la conformité avec la définition
5. Mise en route
  - ▶ Déploiement
6. Transfert à l'exploitation
  - ▶ Mise en marche
  - ▶ Montée en charge progressive et contrôle des objectifs

# Déclinaisons

---

1. Avant-projet
2. Cadrage
3. Etudes préalables/Spécification/Analyse
4. Etudes détaillées/Conception
5. Réalisation/Mise en œuvre/Développement/Tests
6. Déploiement/Installation/Mise en service/Stabilisation
7. Après-projet
  1. Montée en charge
  2. SAV
  3. Réglages
  4. Correctif/Patch



# GPI Cycle de développement



Jean-Marie Mottu  
IUT de Nantes – Département Informatique

# Synoptique de conduite de projet informatique

---

Phases standards	Effectuées en principe par	Phases d'un projet informatique
Etudes préliminaires	MOA	Avant-projet
Conception	MOA (& AMOA)	-> production du cahier des charges
Définition	(A)MOA-MOE	Projet Informatique
Construction	MOE	-> découpages en phases selon un cycle de développement
Mise en route	MOE	
Transfert à l'exploitation	MOE-MOA	Après-projet
		-> livraison du produit

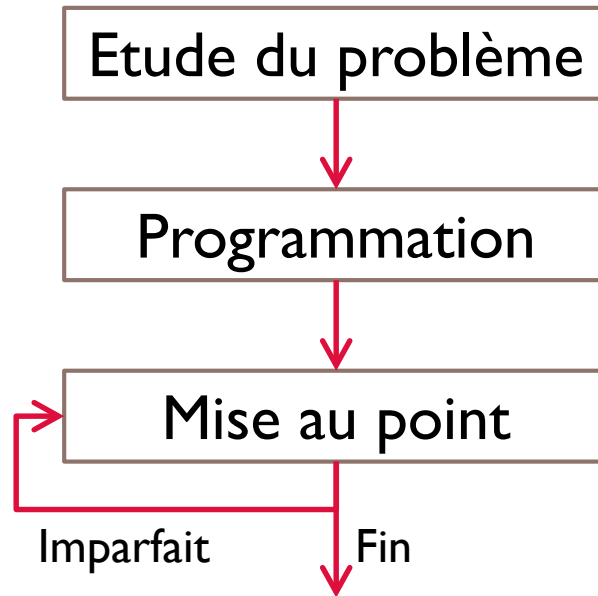
# Modèles de cycle de développement informatique

---

- ▶ Etapes classiques d'un développement informatique
  - ▶ Analyse des besoins et spécification
  - ▶ Conception architecturale et détaillée
  - ▶ Implémentation
  - ▶ Vérification et Validation
- ▶ Organisées en cycle définissant l'enchaînement des grandes activités d'un projet
- ▶ Selon différents modèles à différents niveaux de
  - ▶ Prédiction/Adaptation
  - ▶ itération
  - ▶ formalisation

# Modèle code-and-fix

---



# Modèle code-and-fix

---

## ► Pratique la plus simple

### ► Gestion envisageable-pour des projets

- Soit très simples
- Soit prospectifs

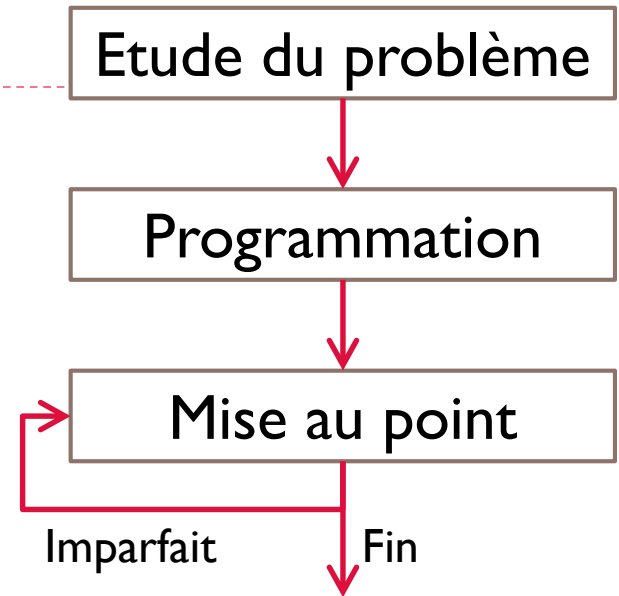
### ► Gestion plus souvent subie que choisie

- « on verra bien en le faisant »

### ► La logique de ce modèle est qu'on ne cernerait vraiment le besoin et la faisabilité qu'en mettant au point le programme

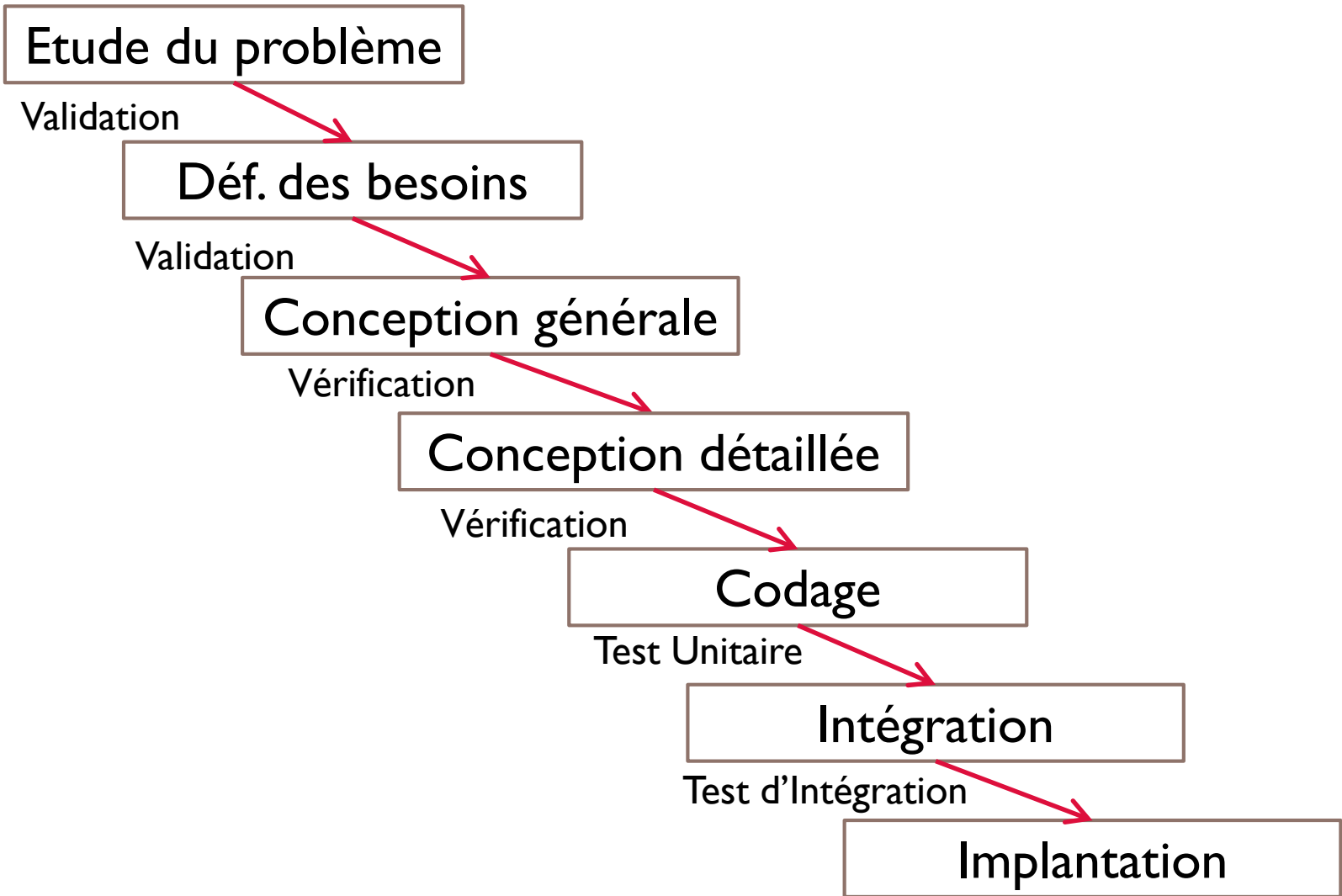
- Cette idée n'est pas fausse et est à la base des modèles agiles
- Attention par contre à sa mise en œuvre
  - Un projet complexe imposera trop de phases de mise au point et un risque de régression mal maîtrisé

« Mieux vaudrait prévenir que guérir »





# Modèle en cascade



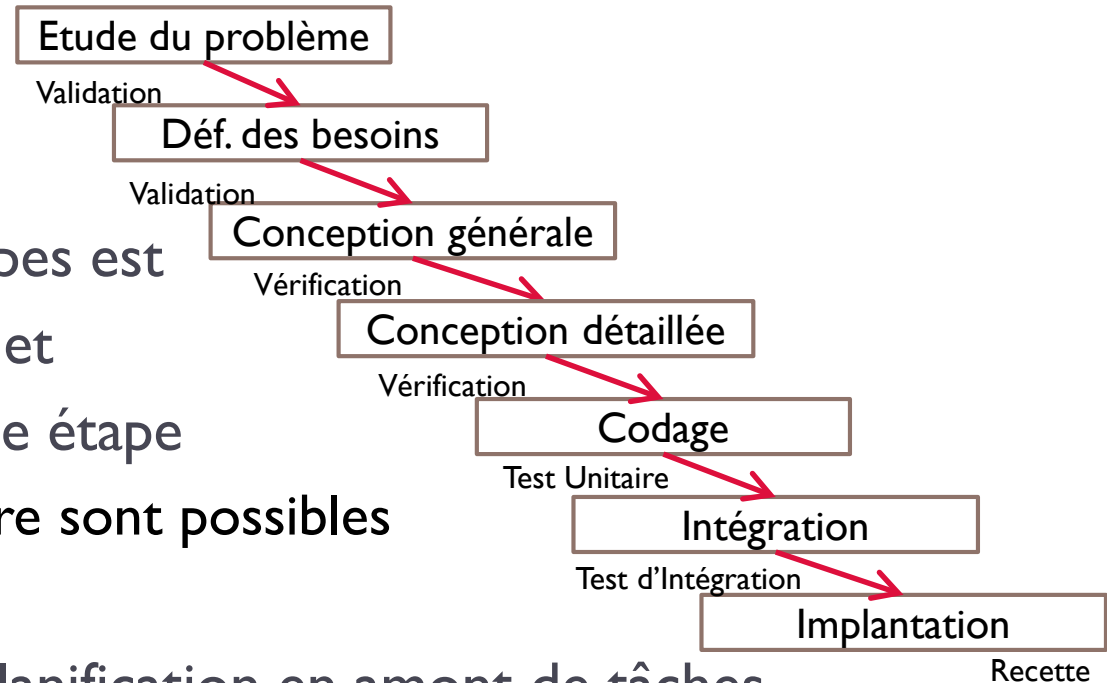
Recette

# Modèle en cascade

- ▶ Waterfall model

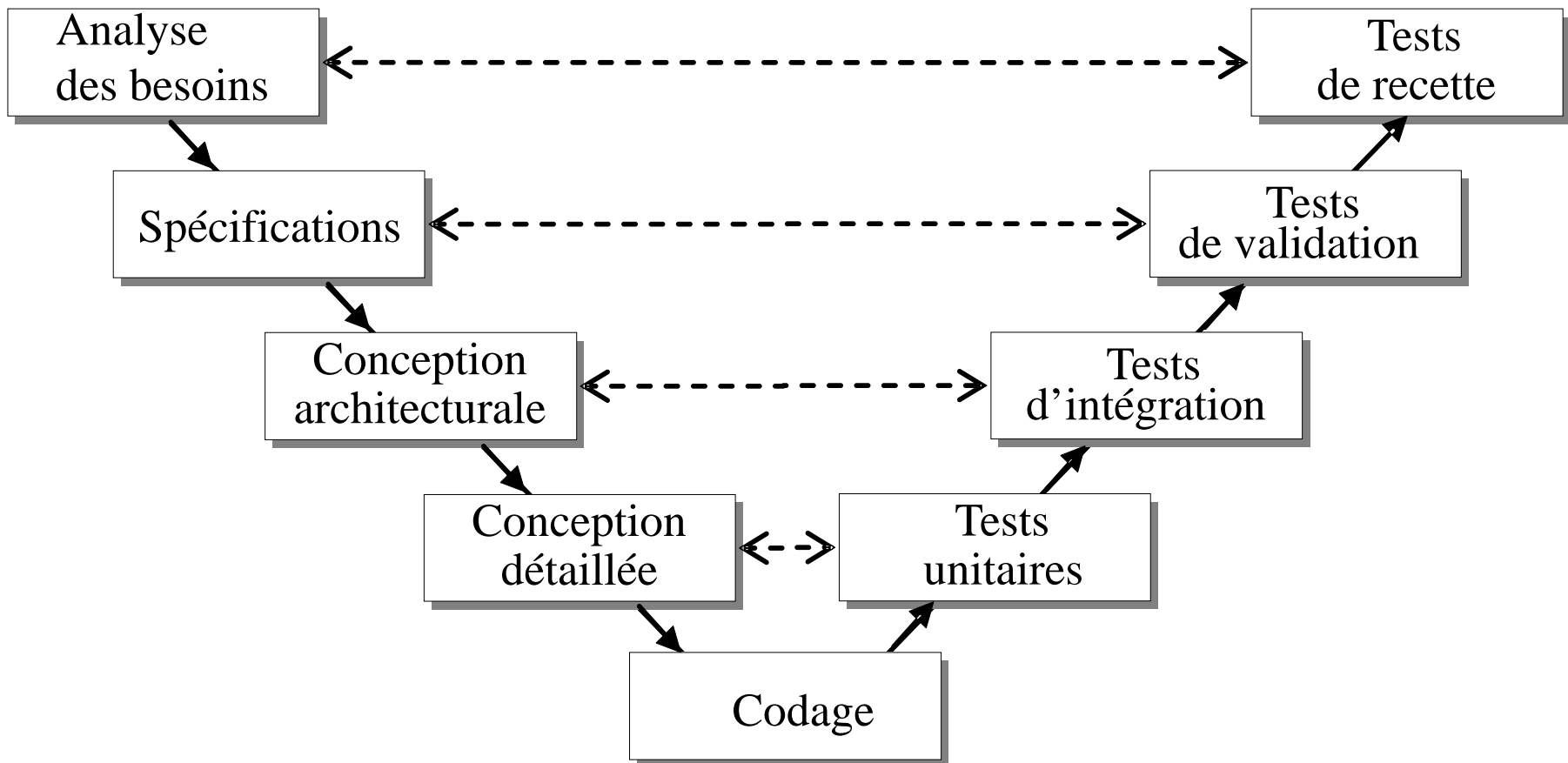
- ▶ Modèle prédictif

- ▶ La succession des étapes est prévue au début du projet
- ▶ Contrôle entre chaque étape
  - ▶ Les retours en arrière sont possibles mais contraignants
- ▶ Directement lié à la planification en amont de tâches successives

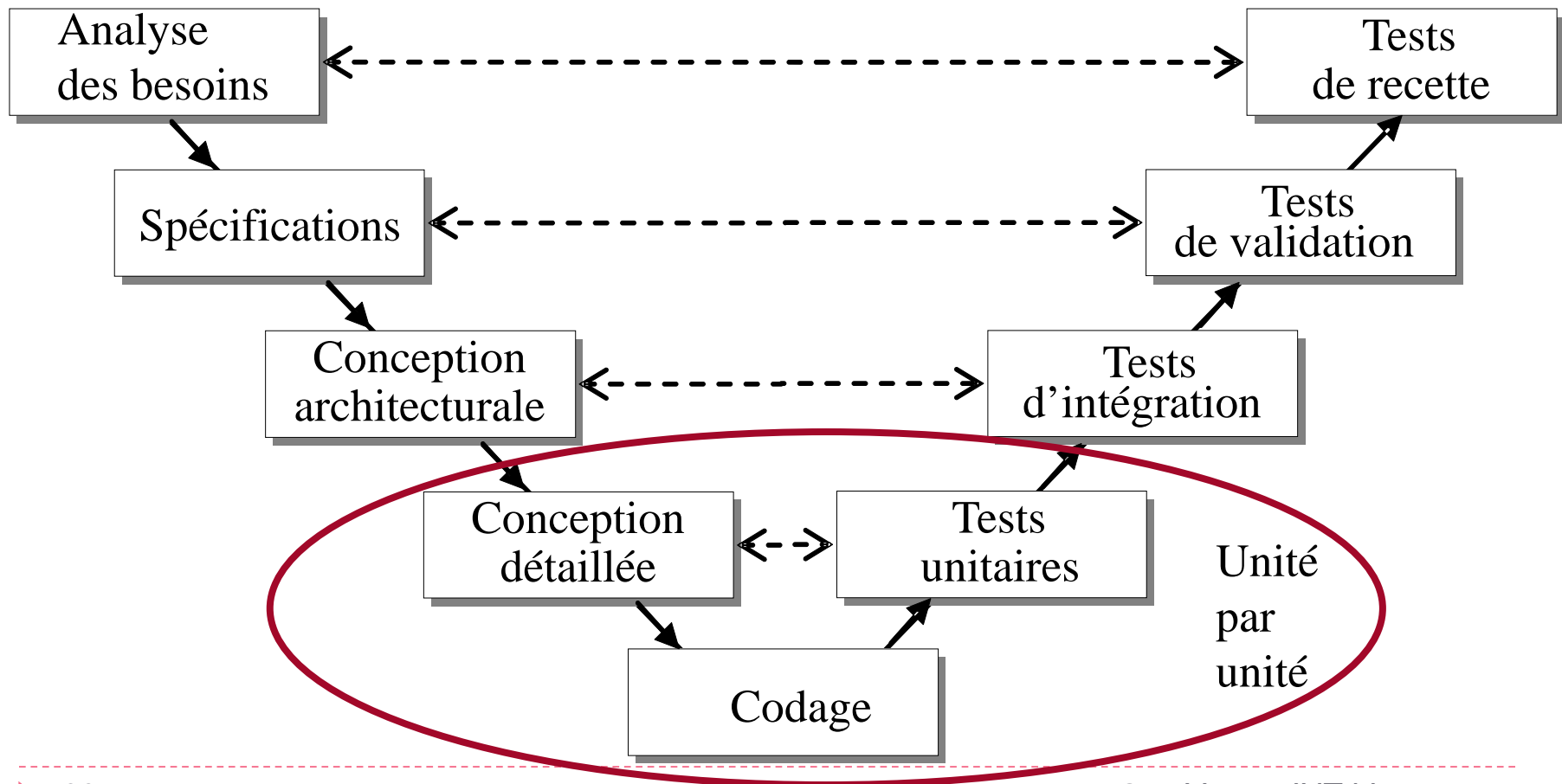


# Modèle en V

---

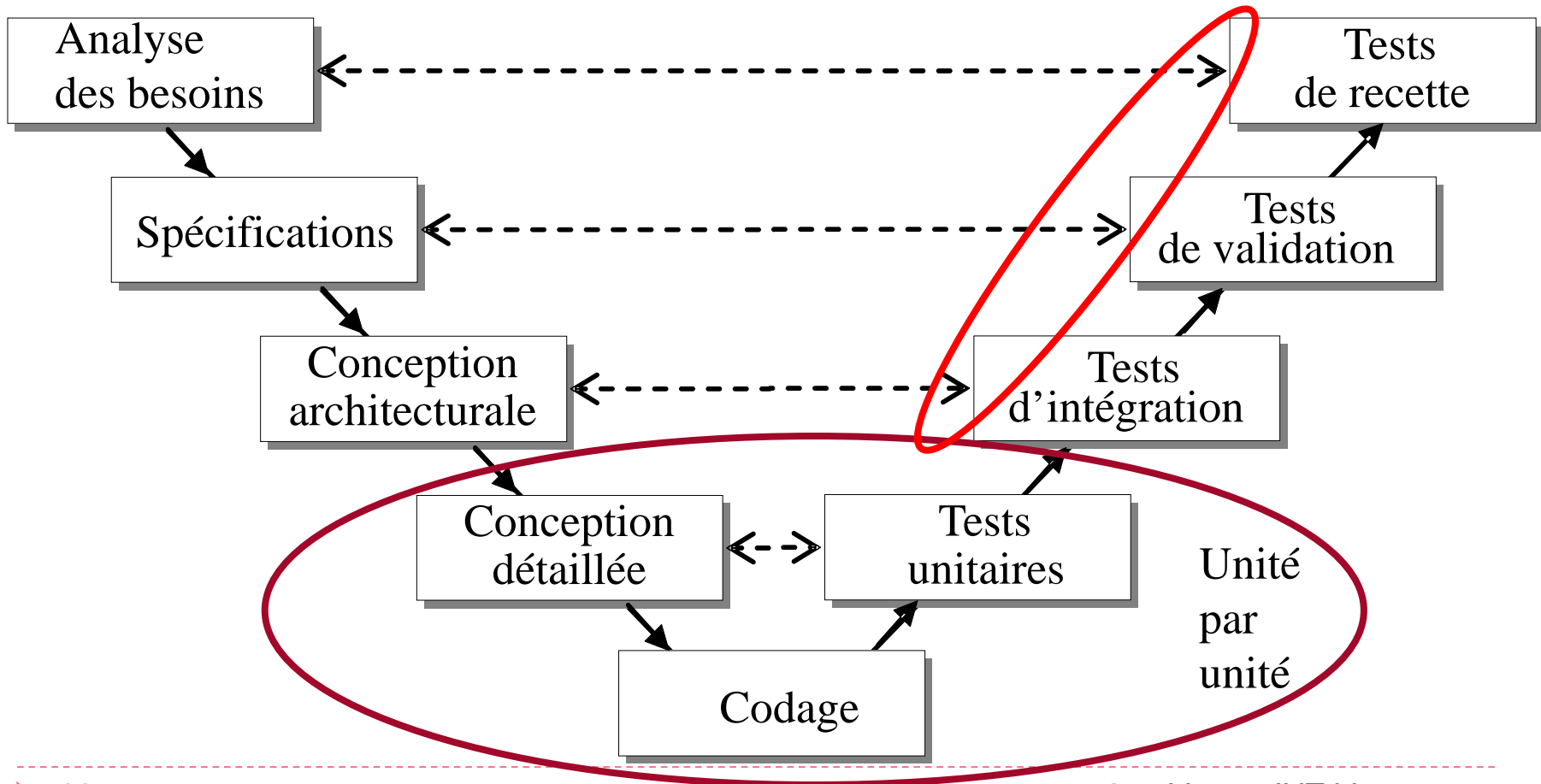


# Modèle en V



# Modèle en V

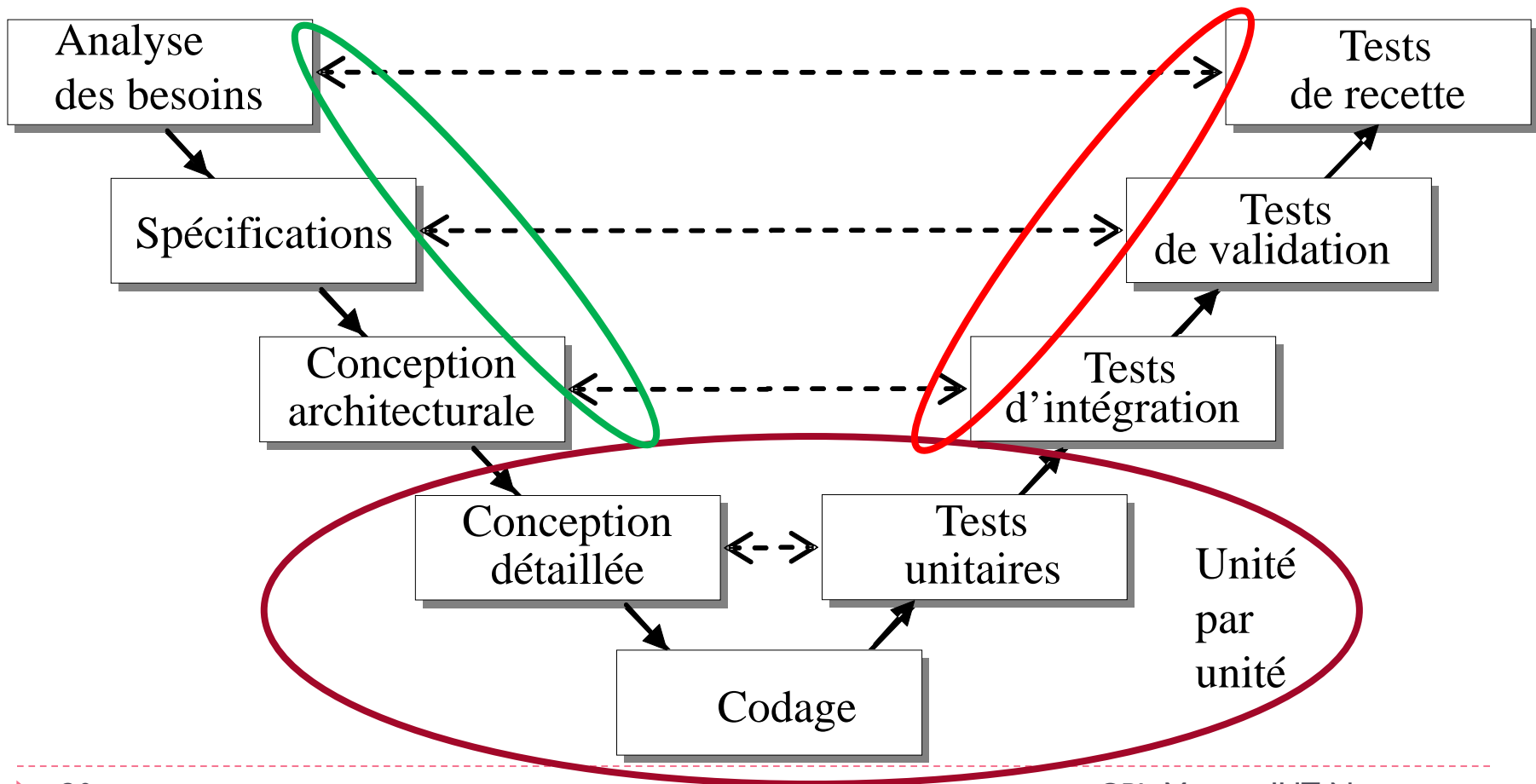
En amont on anticipe  
le test



# Modèle en V

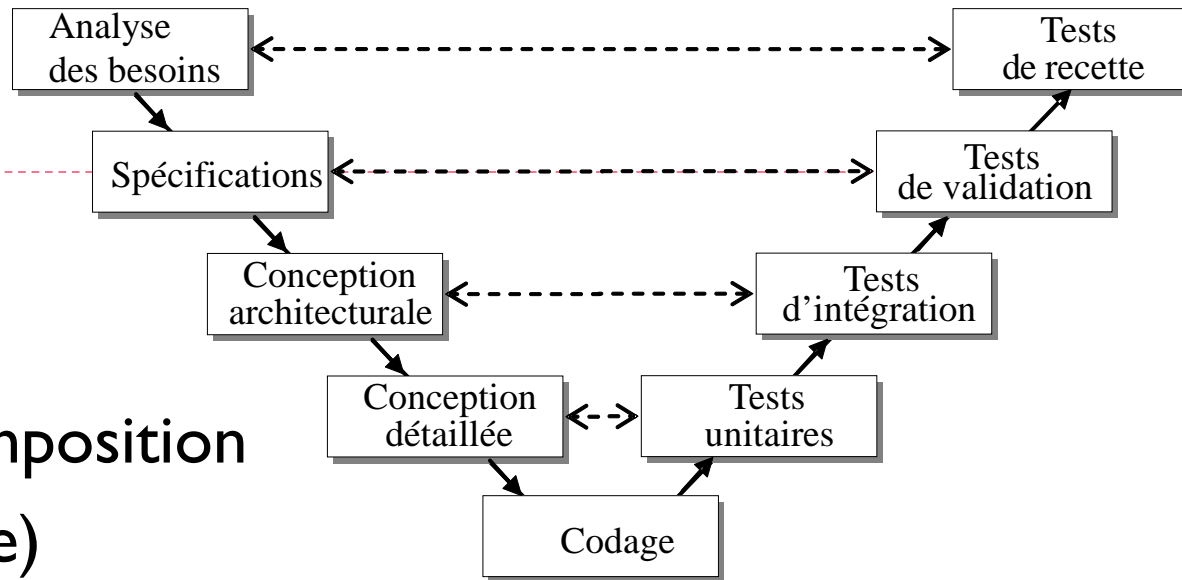
## En amont on anticipe le test

En aval on exploite/remet en cause  
les étapes amonts pour/par le test



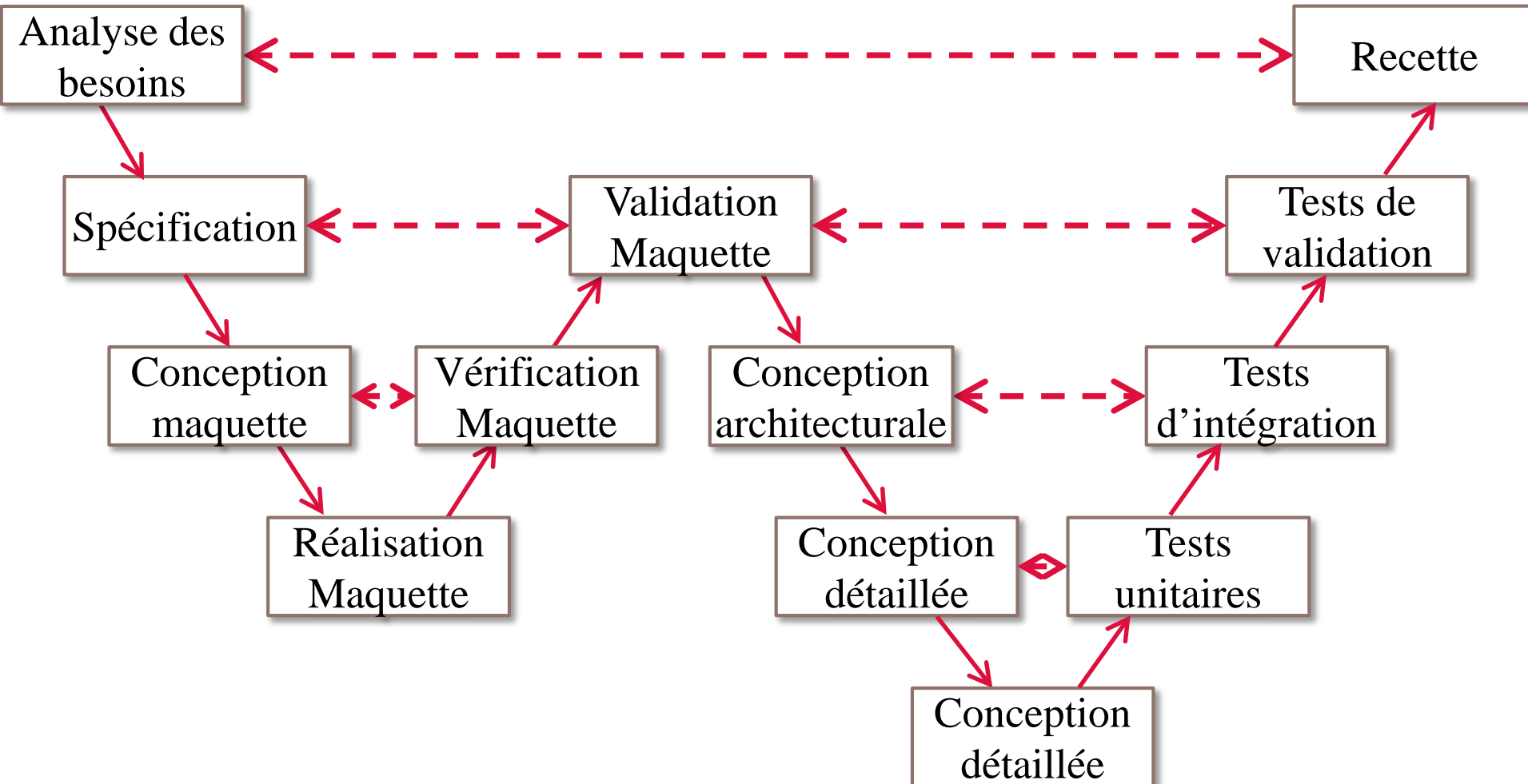
# Modèle en V

- ▶ Standard depuis les années 80
- ▶ Profite d'une décomposition en unité (parallélisable)
- ▶ Réactivité améliorée
  - ▶ Anticipation des tests (et donc des problèmes à survenir)
  - ▶ Limite les retours (à priori on sait d'où viennent les problèmes)
- ▶ Approche prédictive
  - ▶ Phases amonts contraignantes
  - ▶ Le système ne prend forme qu'une fois les unités intégrées



# Modèle en W

## ► Evolution introduisant une phase de maquettage en amont





# Qualité et Critique des modèles prédictifs

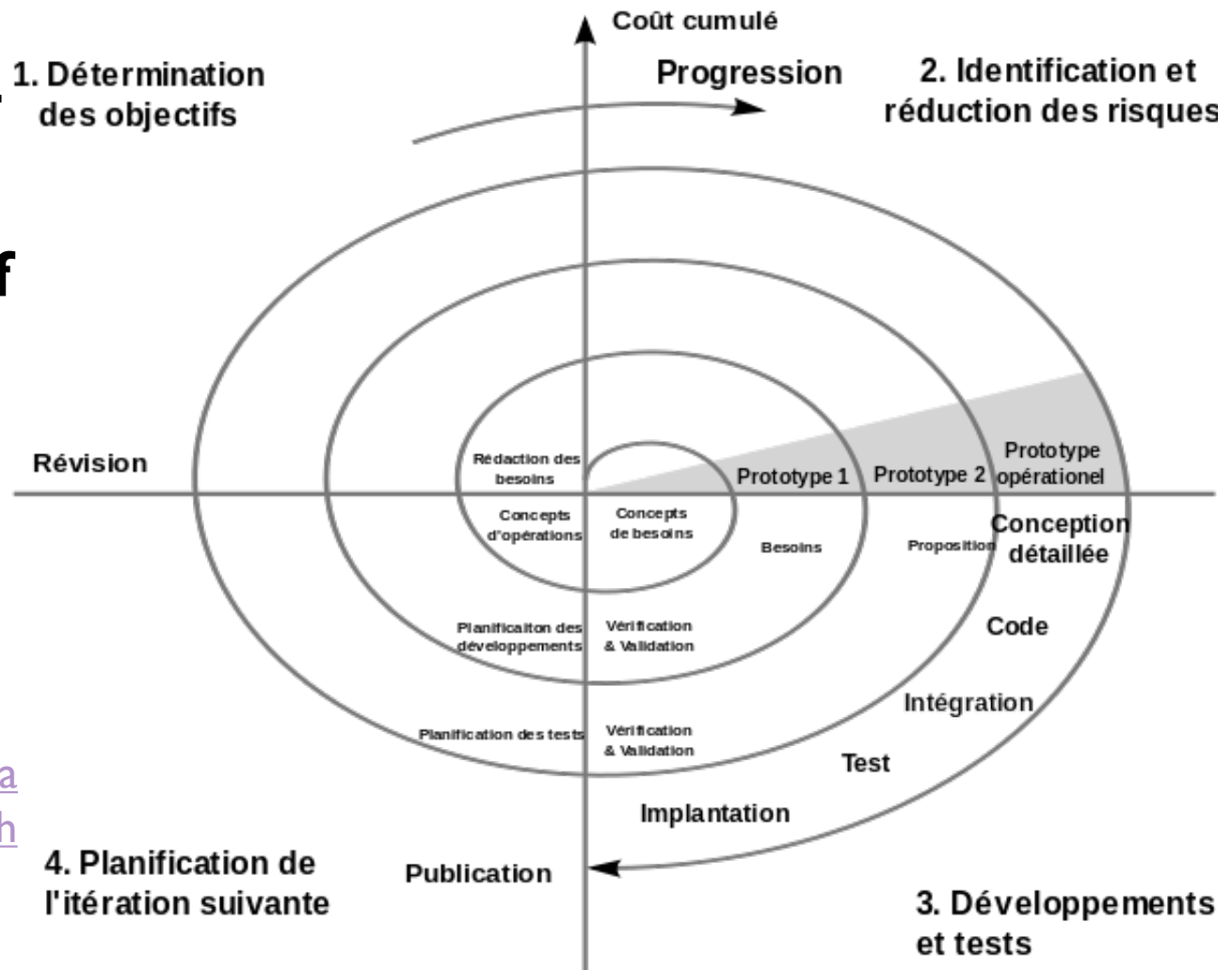
---

- ▶ Rigueur
- ▶ Méthodologie éprouvée
  
- ▶ Effet tunnel
- ▶ Principalement séquentiel
  - ▶ découpage en différentes unités
- ▶ Jalons avec des versions intermédiaires tardives
- ▶ Des facteurs de risque peuvent n'être levés que tard
  - ▶ (test de performance, ergonomie)
- ▶ Pour prédire, il faut beaucoup documenter avant et pendant

# Modèle en spirale

► Un premier modèle **itératif** issue du cycle en V

► Transition d'un mode **prédictif** à un mode **adaptatif**



[https://commons.wikimedia.org/wiki/File:Spirale\\_\(Boehm,\\_1988\).svg](https://commons.wikimedia.org/wiki/File:Spirale_(Boehm,_1988).svg)



# Méthodes Agile

## Le Manifeste Agile

---

- ▶ Rédigé en 2001 par 17 experts
- ▶ 4 valeurs
  - ▶ **Les individus et leurs interactions**
    - ▶ plus que les processus et les outils
  - ▶ **Un logiciel opérationnel**
    - ▶ plus qu'une documentation pléthorique
  - ▶ **La collaboration avec le client**
    - ▶ plus que la négociation de contrat
  - ▶ **L'adaptation face au changement**
    - ▶ plus que le suivi d'un plan

# Le Manifeste Agile

---

- ▶ 12 principes <https://agilemanifesto.org/principles.html>
  - ▶ On retiendra
    - ▶ Beaucoup d'échanges,
    - ▶ Plus de code fonctionnel, plus souvent,
    - ▶ Eviter le superflus,
    - ▶ Accepter les changements.
- ▶ Au programme l'an prochain

# Quelques noms de méthodes appliquées de développement

---

- ▶ Merise (méthode d'analyse orientée SGBD)
  - ▶ associée au modèle Entité-association
- ▶ SADT (Structured Analysis and Design Technique)
- ▶ RUP (Rational Unified Process)
  - ▶ associée au langage UML
- ▶ Méthodes dites “agiles”
  - ▶ Développement Rapide d'applications (RAD)
  - ▶ eXtreme Programming (XP)
  - ▶ Scrum
  - ▶ Lean IT
  - ▶ Kanban
  - ▶ Test Driven Development
- ▶ Etc.