

# Méthodes de conduite de projet

Où les enseignements prennent leur place

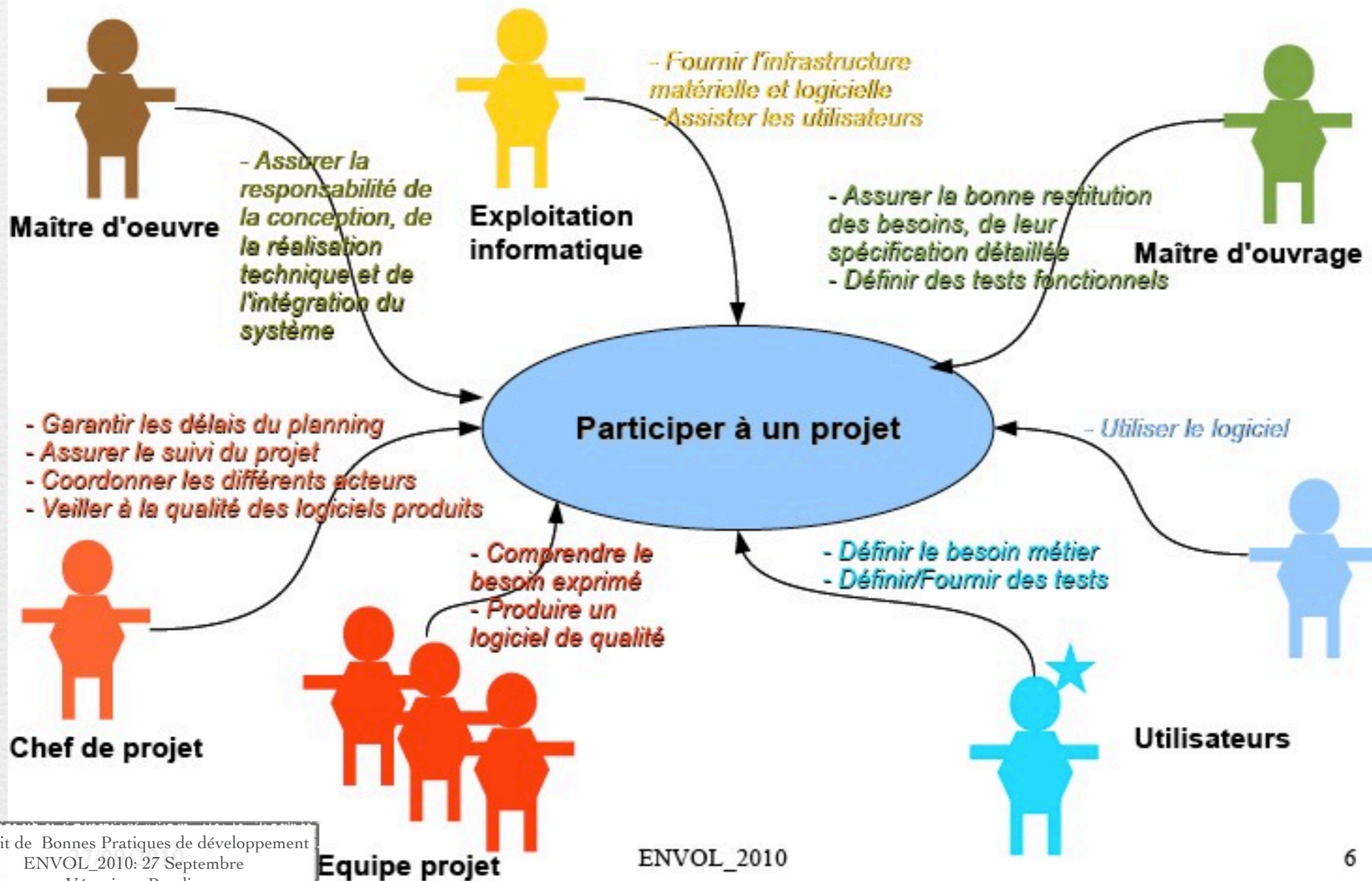
Mireille Blay-Fornarino  
IUT Nice-Sophia Antipolis  
[blay@polytech.unice.fr](mailto:blay@polytech.unice.fr)  
<http://www.polytech.unice.fr/~blay>

Site web du module : <http://anubis.polytech.unice.fr/iutl>

# Projet en général : qq définitions

- **Projet** : ensemble d'actions à entreprendre afin de répondre à un besoin défini (avec une qualité suffisante), dans un délai fixé, mobilisant des ressources humaines et matérielles, possédant un coût.
- **Maître d'ouvrage** : personne physique ou morale propriétaire de l'ouvrage. Il détermine les objectifs, le budget et les délais de réalisation.
- **Maître d'oeuvre** : personne physique ou morale qui reçoit mission du maître d'ouvrage pour assurer la conception et la réalisation de l'ouvrage.
- **Conduite de projet** : organisation *méthodologique* mise en oeuvre pour faire en sorte que l'ouvrage réalisé par le maître d'oeuvre réponde aux attentes du maître d'ouvrage dans les contraintes de délai, coût et qualité.

# I.I PROJET INFORMATIQUE



Extrait de Bonnes Pratiques de développement  
ENVOL\_2010: 27 Septembre  
Véronique Baudin  
Violaine Louvet

# PROJET INFORMATIQUE

- Projet = ensemble fini d'activités et d'actions dont l'objectif est de répondre à un besoin défini en respectant des contraintes de temps et de coûts.
- Objectif décliné en 5 aspects
  - Fonctionnel : répondre à un besoin
  - Technique : respecter des spécifications et des contraintes de mise en oeuvre
  - Organisationnel : respecter un mode de fonctionnement de la structure
  - Respect des délais : respecter les échéances du planning
  - Respect des coûts : respecter le budget fixé

# Quelques causes des problèmes de développement logiciel

- Mauvaise interprétation des demandes des utilisateurs finaux
- Surspécification
- Incapacité à tenir compte des changements du cahier des charges
- Construction d'un processus/algorithme non fiable
- Membre(s) de l'équipe isolé(s): incapable(s) de déterminer qui a changé quoi, quand, où et pourquoi
- Procédures de tests coûteuses ou inefficaces
- Découverte tardive de problèmes
- *Faible qualité du logiciel*








# UML : «No silver bullet»

- Connaître UML n'est pas suffisant pour réaliser de bonnes conceptions
  - Il faut en plus savoir maîtriser/contrôler le processus de développement d'un projet
- Cycle de vie du logiciel
  - Période entre l'idée du logiciel et sa mise en service

# Cycle de vie du logiciel

- 1) Activités
- 2) Différents types de cycles

# Activités du cycle de vie du logiciel

-  **1) Définition des besoins (Requirements)**
  -  Expression des besoins dans le langage du client
  
-  **2) Spécifications**
  -  Traduction des besoins dans un langage plus informatique
  -  Description du système d'un point de vue extérieur
    - ▶ Ce qu'il doit faire
    - ▶ Pas comment il le fait
  -  Spécifications fonctionnelles et non-fonctionnelles
  -  Doit rester accessible au client (contrat)



# Activités du cycle de vie du logiciel

## 2) Spécifications (suite)

### Spécifications fonctionnelles



- ▶ Les fonctions/services rendus par le système

### Spécifications non-fonctionnelles

- ▶ Utilisabilité
- ▶ Fiabilité (reliability)
- ▶ Performance
- ▶ Supportabilité (maintenabilité)
- ▶ Conditions d'implémentation, de déploiement...
- ▶ Interface
- ▶ Conditions d'exploitation
- ▶ Conditionnement
- ▶ Aspects juridiques
- ▶ Aspects financiers...

# Activités du cycle de vie du logiciel



## 3) Conception

-  Traduction des spécifications en termes d'artefacts logiciels
-  Peut être plus ou moins détaillée

## 4) Codage

-  Traduction de la conception en code

## 5) Test unitaires

-  Test de chaque module individuellement
-  Généralement de la responsabilité du développeur du module

## 6) Tests d'intégration



-  Test de la composition de plusieurs modules (sous-systèmes)

# Activités du cycle de vie du logiciel

## 7) Validation




-  Test du système final par rapport aux spécifications

## 8) Recette/Mise en exploitation

-  Acceptation du système final
-  Peut être l'objet d'une procédure formelle et parfois officielle

## 9) Gestion des changements, gestion de configuration

## Gestion de projet




-  Orthogonale à toutes ces activités
-  Gestion du temps, des coûts, des équipes
-  Gestion des relations avec les clients...

# Cycle de vie du logiciel

- 1) Activités
- 2) Différents types de cycles

# Modèles de cycle de vie d'un logiciel

## Modèles de cycle de vie

-  organiser les différentes activités du cycle de vie pour l'obtention d'un logiciel fiable, adaptable et efficace
-  guider le développeur dans ses activités techniques
-  fournir des moyens pour gérer le développement et la maintenance
  - ▶ ressources, délais, avancement, etc.

## *Modèles linéaires*

-  en cascade et variantes

## *Modèles non linéaires*

-  en spirale, incrémentaux, itératifs

# Modèle en cascade (70)

## Analyse

- Expression du besoin
- Analyse détaillée

## Conception

- Etude technique préalable
- Conception préliminaire
- Conception détaillée

## Réalisation

- Codage
- Mise au point
- Tests unitaires

## Intégration

- Intégration
- Tests d'Intégration

## Validation

- Validation
- Mise en œuvre

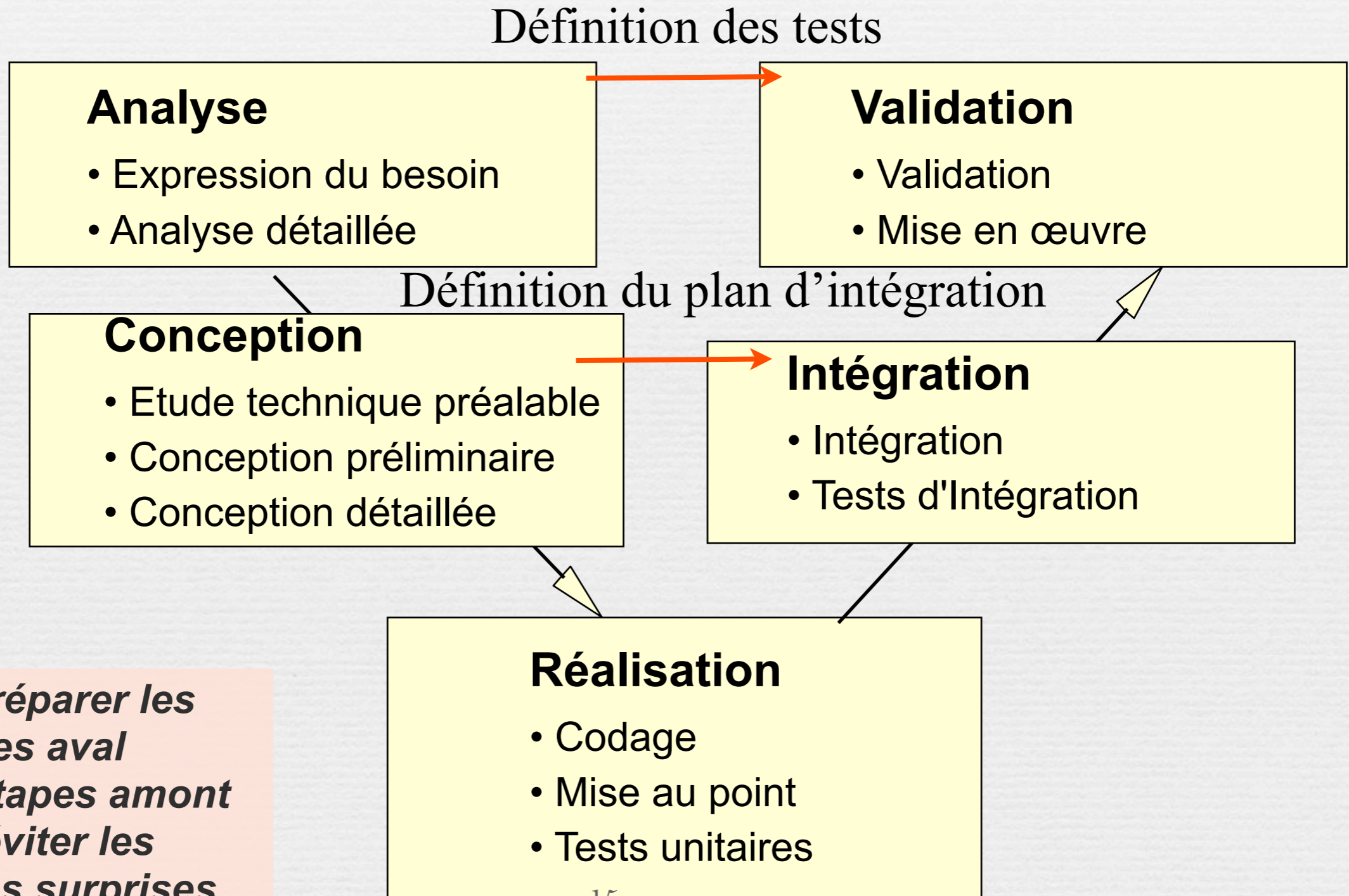
- Les vrais projets suivent rarement un développement séquentiel
- Établir tous les besoins au début d'un projet est difficile
- Le produit apparaît tard, les tests sont tardifs.

Seulement applicable pour les projets qui sont bien compris et maîtrisés

# Modèle en « V »

## → Cycle de vie normalisé AFNOR

Variante du cycle en « cascade »



*Vise à préparer les étapes aval lors des étapes amont pour éviter les mauvaises surprises*

# Modèle en « V »

## Le cycle en V

- permet une meilleure anticipation
- présente des tests bien structurés

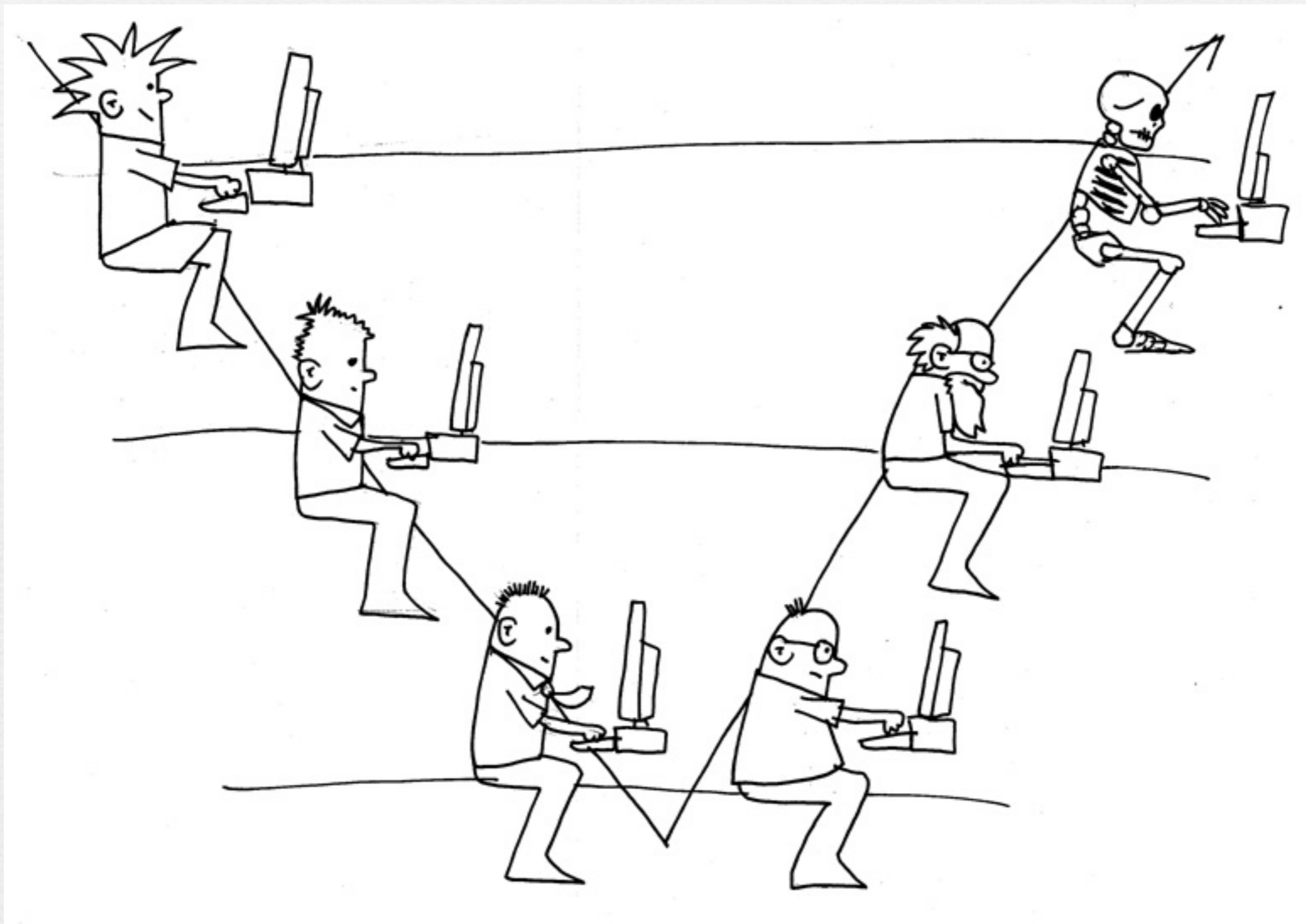
## Mais

- le cadre de développement est rigide
- la durée est souvent trop longue
- le produit apparaît très tard (validation finale tardive très coûteuse s'il y a des erreurs)

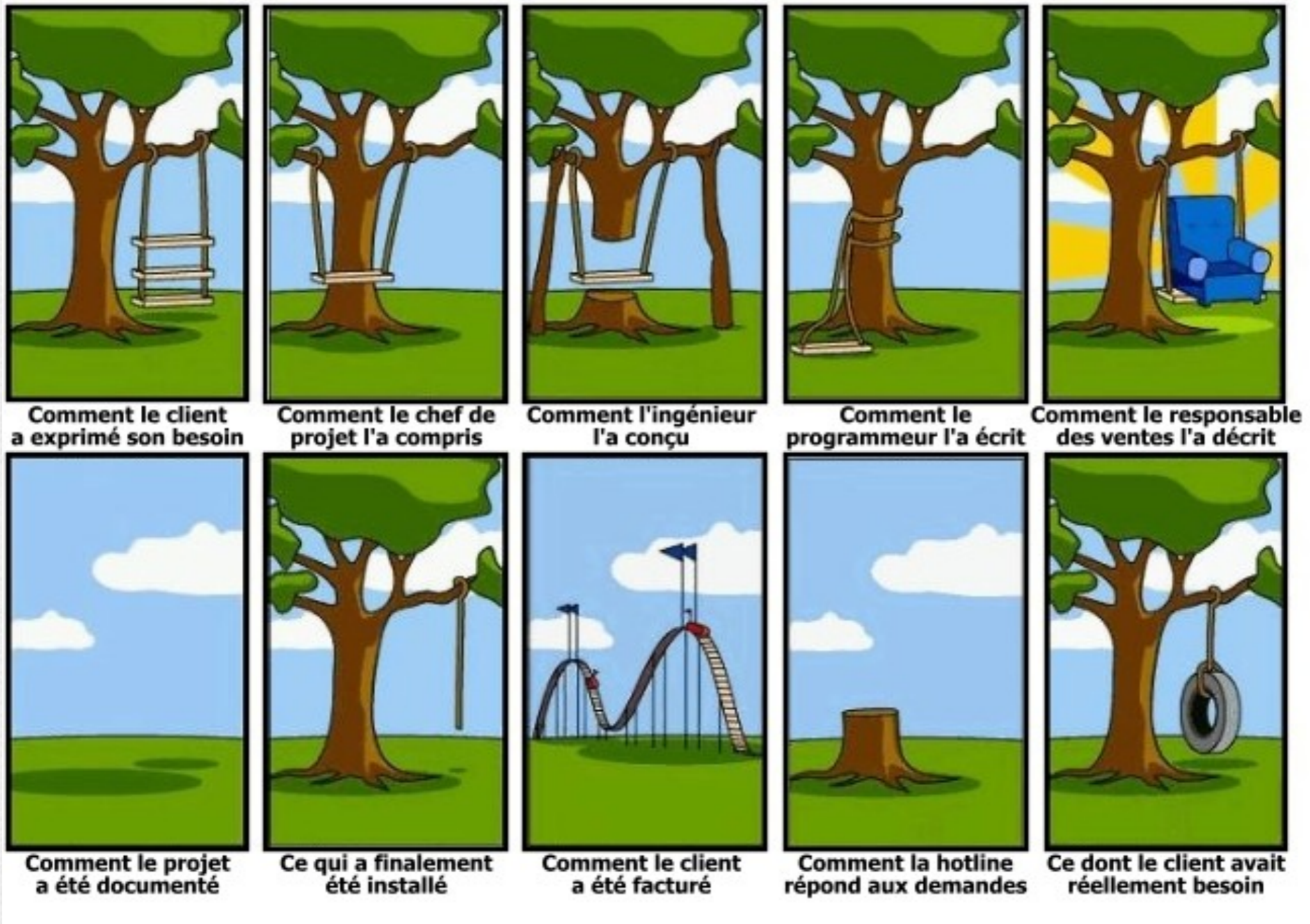
**Variante : W** (validation d'un maquette avant conception)



# Modèle en « V »

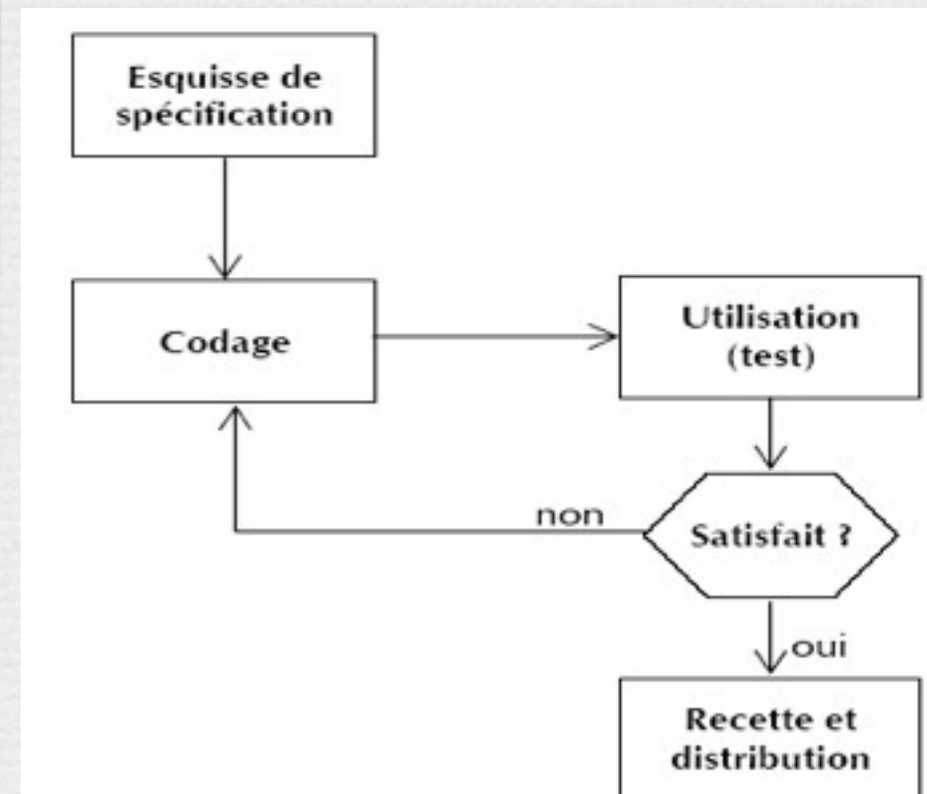
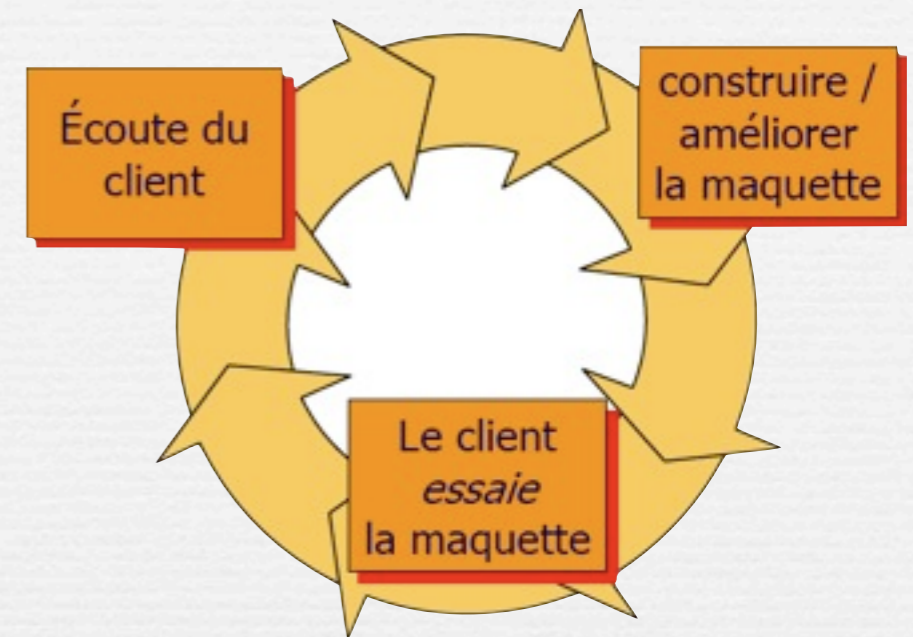


# Problèmes avec le processus classique...



# Cycle exploratoire Prototypage RAD\*

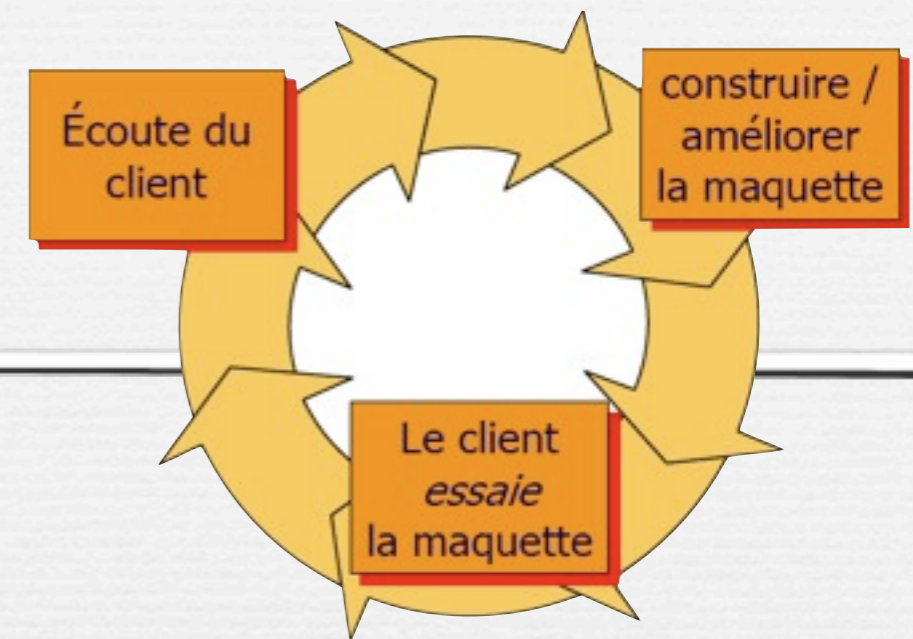
- Discuter et interagir avec l'utilisateur
- Vérifier l'efficacité réelle d'un algorithme
- Vérifier des choix spécifiques d'IHM
- Souvent utilisé pour identifier les besoins
  - Prototype jetable (moins de risque ?)
- Souvent implémenté par des générateurs de code
  - Prototype évolutif



# Cycle exploratoire, Prototypage, RAD\*

## 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 ?



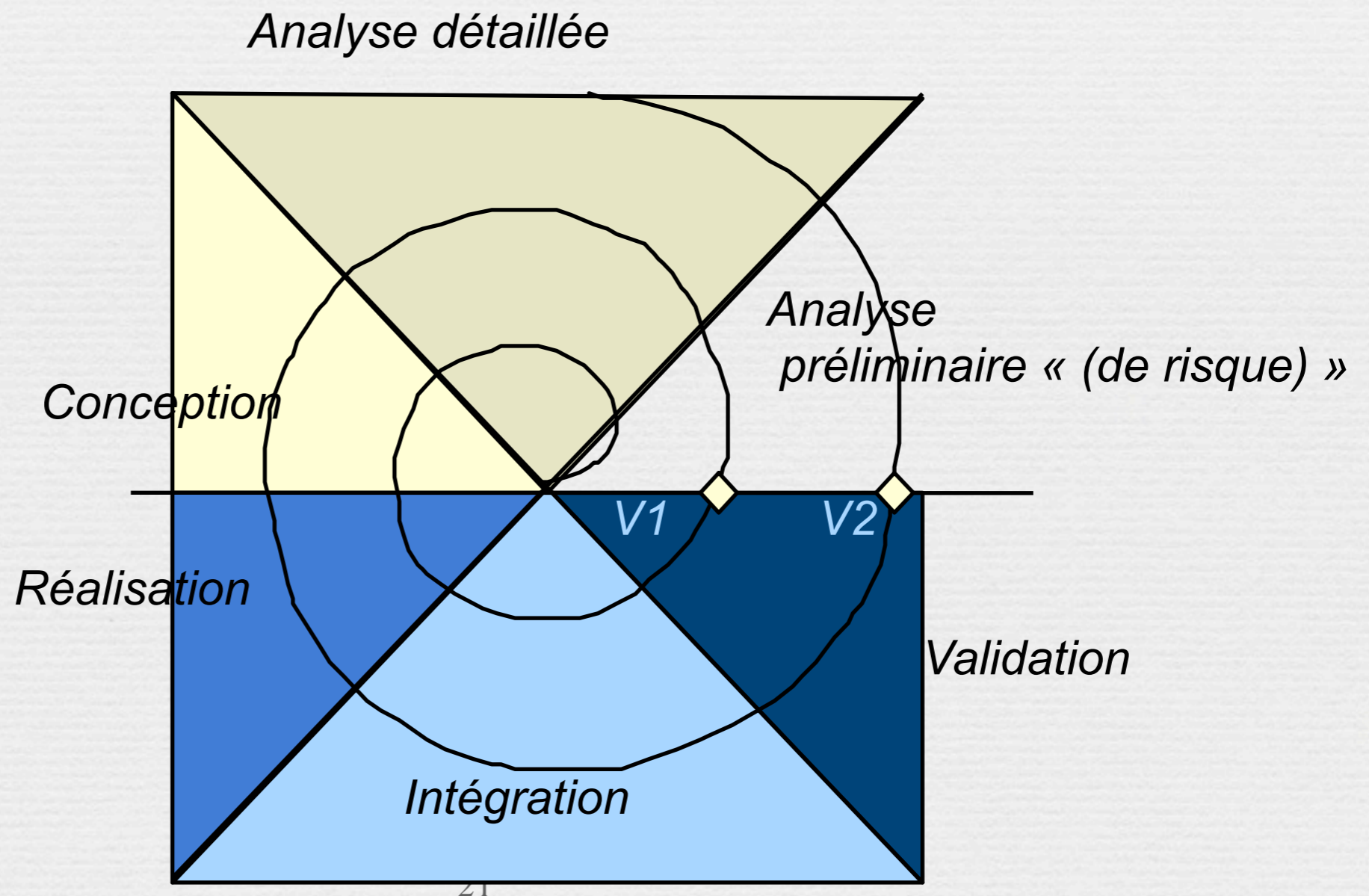
Admissible tel quel pour

- du prototypage
- de très petits projets développés
- par de très petites équipes
- avec de très petits enjeux

Cependant, retour en force dans les processus « agiles » mais sous une forme beaucoup plus élaborée

# Modèle en « spirale » (Boehm 88)

Couplage de la nature itérative du prototypage avec les aspects systématiques et contrôlés du modèle en cascade.



# Modèle en « spirale »

- ➔ Bien adapté aux développements innovants
  - les progrès sont tangibles : c'est du logiciel qui « tourne » et pas seulement des kilos de documents
  - réduit les risques si bien appliqué : possibilité de s'arrêter « à temps », i.e. avant que l'irréalisabilité du projet ait créé un gouffre financier
  
- ➔ Moins simple à manager
  - difficile à gérer en situation contractuelle
  - mal contrôlé => on retombe dans le *hacking*

# Modèle en « spirale »

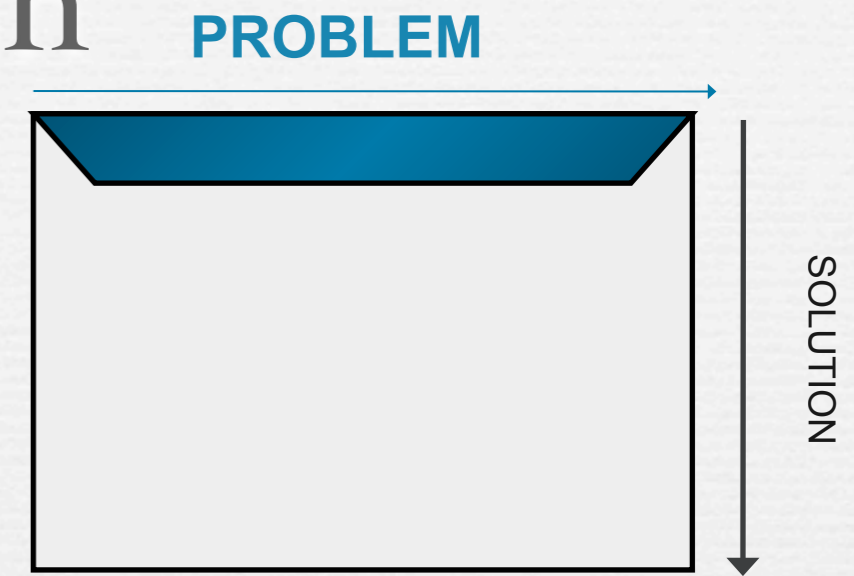
- ➔ Bien adapté aux développements innovants
  - les progrès sont tangibles : c'est du logiciel qui « tourne » et pas seulement des kilos de documents
  - réduit les risques si bien appliqué : possibilité de s'arrêter « à temps », i.e. avant que l'irréalisabilité du projet ait créé un gouffre financier
  
- ➔ Moins simple à manager
  - difficile à gérer en situation contractuelle
  - mal contrôlé => on retombe dans le *hacking*

**À la base de tous les processus « agiles »**

# Stratégies d'itération

## Large et peu profonde

- ▶ Analyse de l'ensemble du domaine
  - Tous les cas d'utilisation sont étoffés
- ▶ Définition d'une architecture générale



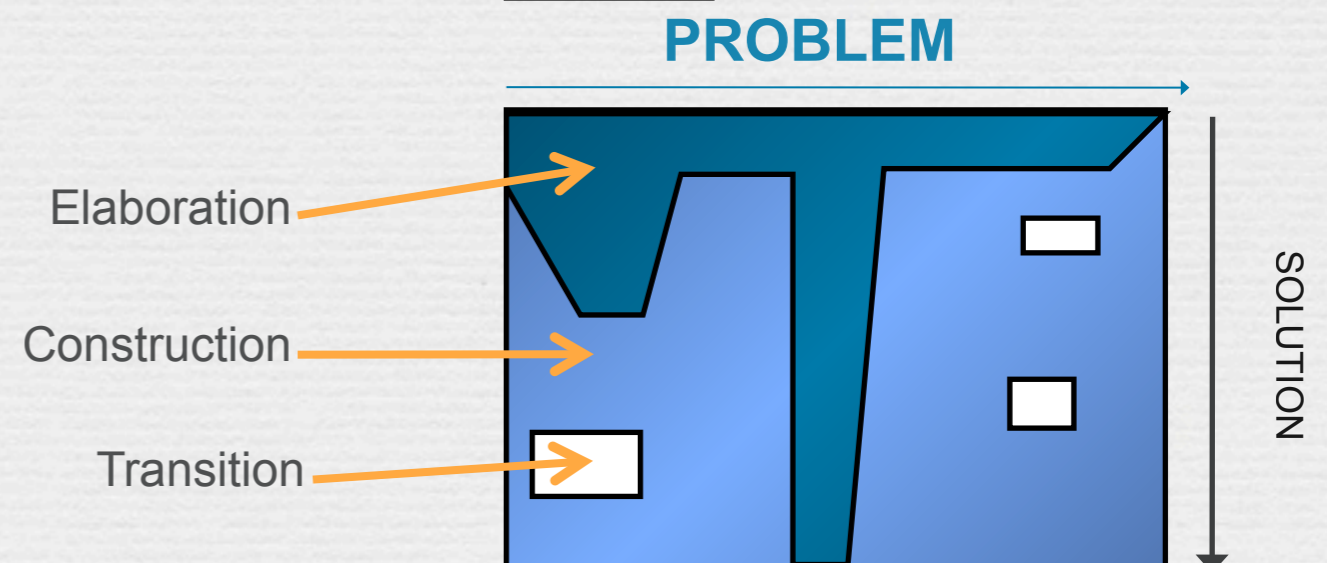
## Etroite et profonde

- ▶ Analyse en profondeur d'une partie du problème
- ▶ Développement de cette partie



## Hybride

- ▶ Un mélange des stratégies





# Méthode

# Méthode/Processus

## ● Méthode :

- guide +/- formalisé, démarche reproductible pour obtenir des solutions fiables
- capitalise l'expérience de projets antérieurs

## ● Une méthode définit

- des concepts de modélisation / les **artefacts** à produire
- activités et chronologie des activités
- un ensemble de règles et de conseils pour tous les participants

– notation + démarche (+ outils)  
– façon de modéliser et façon de travailler

# Notion d'artefact

- Un artefact est un élément produit ou modifié dans le cadre d'un processus de développement
- Il peut s'agir d'un document, un diagramme, un compte rendu de réunion...
- ...mais aussi un code source, un écran, une base de données...

# Outils et Méthode

● Une méthode spécifique

● des activités

● des artefacts à réaliser

● Il est souvent vital de disposer d'outil(s) soutenant le processus

● en pilotant / permettant les activités

● en gérant les artefacts du projet

● Les outils peuvent être plus ou moins

● intégrés à la méthode

● interopérables

● achetés / fabriqués / transformés...

- Outils de planification
- Outils de gestion des versions
- Outils de gestion de documentation
- Outils de maquettage
- Outils de gestion des tests
- Outils de modélisation
  - \* pro, rétro, roundtrip
- Ateliers de développement logiciel
- Outils de vérification

# Fourniture des applications : hier et aujourd'hui



Extrait de : Agilité informatique : un guide de la maîtrise du cycle<sup>28</sup> de vie des applications par HP

# Évolution des méthodes

## ● Origine : fin des années 60

- problèmes de qualité et de productivité dans les grandes entreprises, mauvaise communication utilisateurs / informaticiens
- méthodes = guides pour l'analyse et aide à la représentation du futur SI
- conception par découpage en sous-problèmes, analytico-fonctionnelle
- méthodes d'analyse structurée

## ● Ensuite

- conception par modélisation : « construire le SI, c'est construire sa base de données »
- méthodes globales qui séparent données et traitements

# Évolution des méthodes

## **Maintenant**

 conception pour et par réutilisation

- ▶ Frameworks, Design Patterns, bibliothèques de classes

 méthodes

- ▶ exploitant un capital d'expériences
- ▶ unifiées par une notation commune (par ex. UML)
- ▶ procédant de manière incrémentale
- ▶ validant par simulation effective



146.38

00:12:46 10

02.33 [XMIT]



09:07:25 09:54:01





146.38

00:12:46 10

02.33 [XMIT]

# III. Pratiques contemporaines

09:07:15 09:54:01

# Un nouveau problème...



# Un nouveau problème...

- Il ne s'agit pas de remplacer la cascade



# Un nouveau problème...

- Il ne s'agit pas de remplacer la cascade
- Mais de ne pas la suivre aveuglément



# Un nouveau problème...

- Il ne s'agit pas de remplacer la cascade
- Mais de ne pas la suivre aveuglément
  - ⇒ Ce que d'ailleurs à peu près personne ne faisait plus



# Un nouveau problème...

- Il ne s'agit pas de remplacer la cascade
- Mais de ne pas la suivre aveuglément
  - ⇒ Ce que d'ailleurs à peu près personne ne faisait plus
- Et de prescrire des méthodes de management complémentaires assurant la maximisation des apprentissages



# Un nouveau problème...

- Il ne s'agit pas de remplacer la cascade
- Mais de ne pas la suivre aveuglément
  - ⇒ Ce que d'ailleurs à peu près personne ne faisait plus
- Et de prescrire des méthodes de management complémentaires assurant la maximisation des apprentissages
- C'est l'origine des avancées méthodologiques contemporaines



# Un nouveau problème...

- Il ne s'agit pas de remplacer la cascade
- Mais de ne pas la suivre aveuglément
  - ⇒ Ce que d'ailleurs à peu près personne ne faisait plus
- Et de prescrire des méthodes de management complémentaires assurant la maximisation des apprentissages
- C'est l'origine des avancées méthodologiques contemporaines
  - ⇒ Logiciel libre



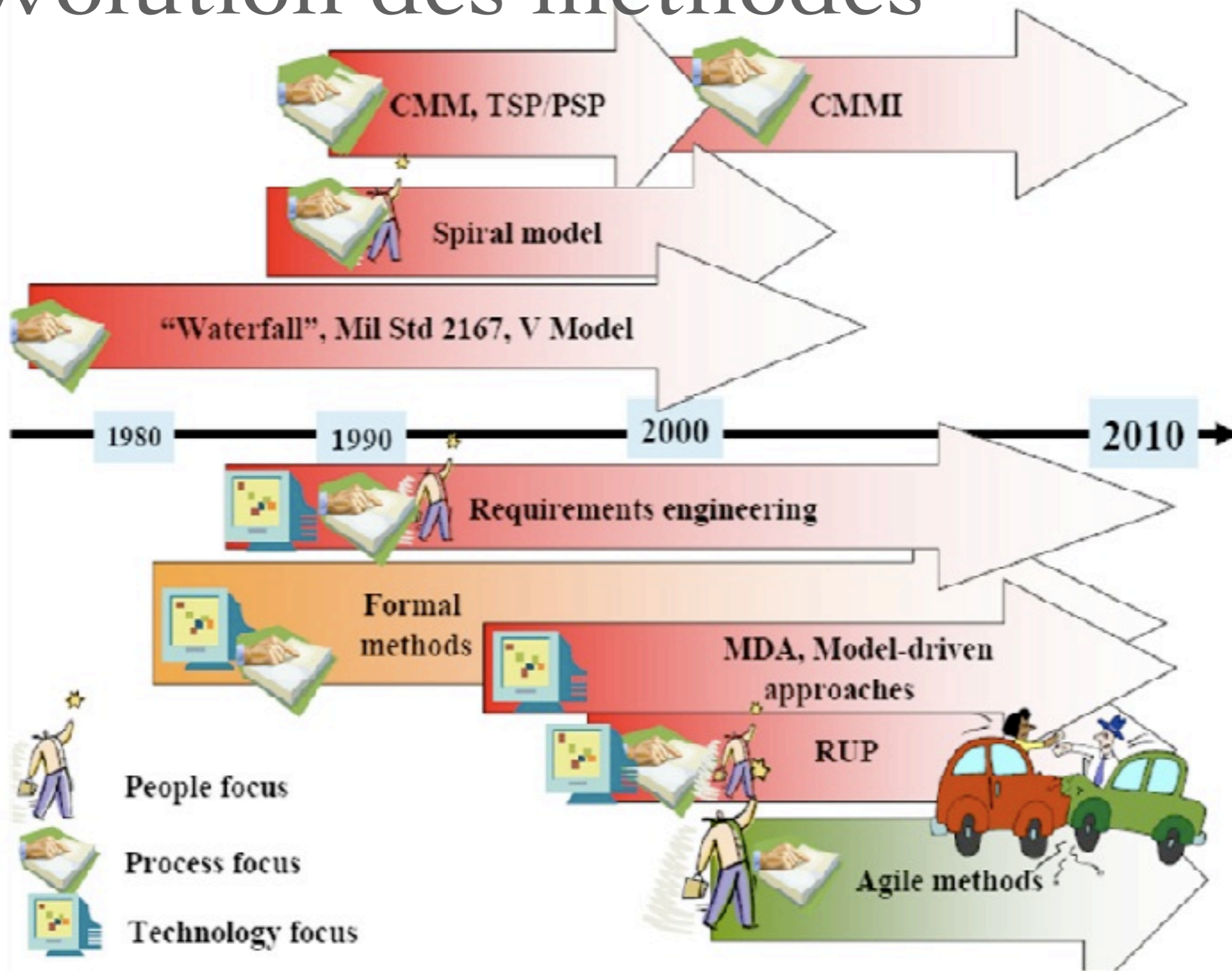


# Un nouveau problème...

- Il ne s'agit pas de remplacer la cascade
- Mais de ne pas la suivre aveuglément
  - ⇒ Ce que d'ailleurs à peu près personne ne faisait plus
- Et de prescrire des méthodes de management complémentaires assurant la maximisation des apprentissages
- C'est l'origine des avancées méthodologiques contemporaines
  - ⇒ Logiciel libre
  - ⇒ Méthodes agiles



# Évolution des méthodes

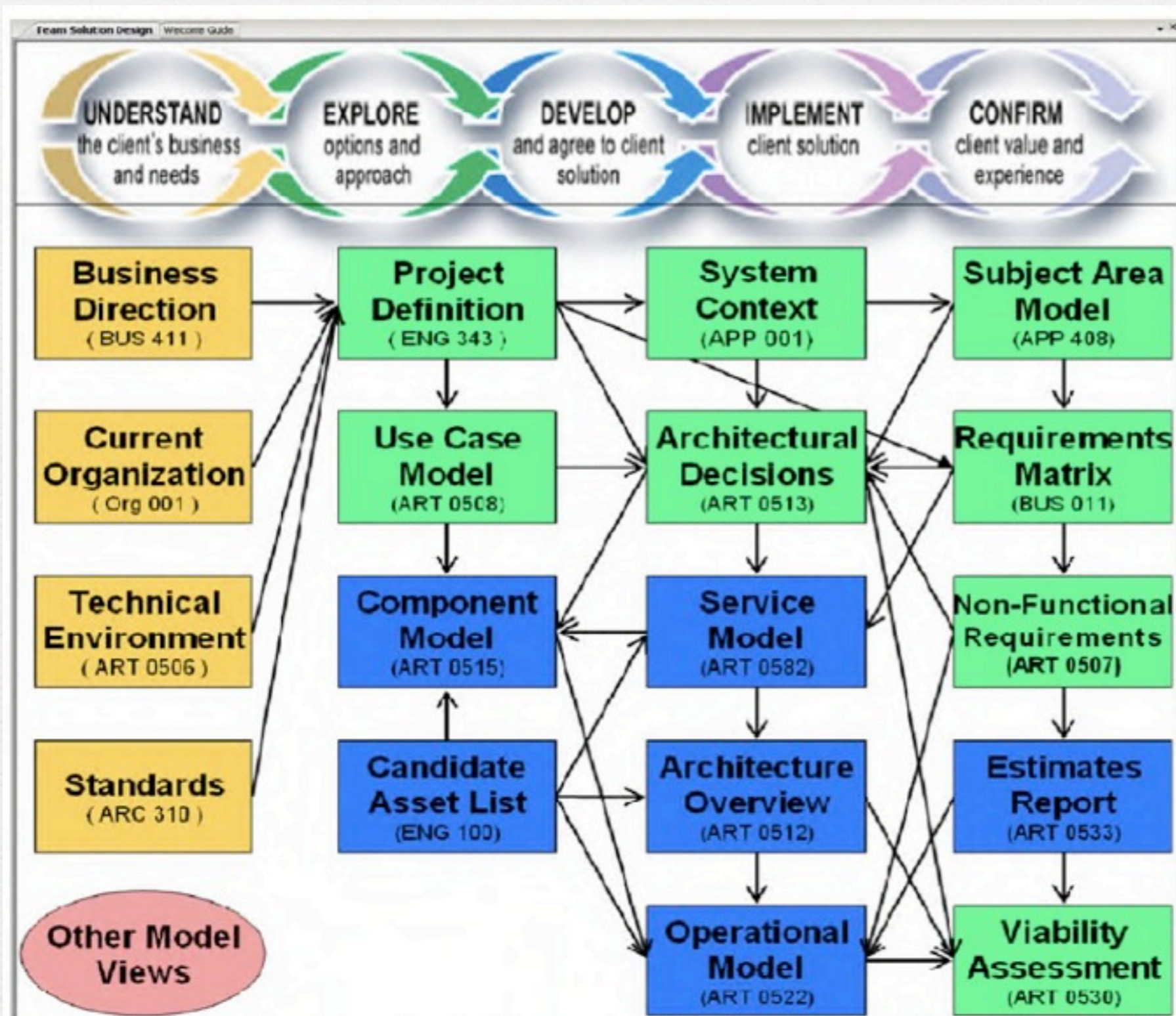


<http://www.entreprise-agile.com/HistoAgile.pdf>

Figure 4. — Vision globale (source NCR 2007)

# Évolution des méthodes

IBM methodology : IBM Team Solution Design.



Extrait rapport  
de stage,  
M2 IFI AL,  
2011,  
Mohamed  
Zouhaier  
Ramadhane,  
IBM La Gaude

# Bibliographie

UNIVERSITE PARIS XII -ISIAG , MASTER 2, METHODOLOGIE ET CONDUITE DE PROJETS, CHAPITRE 3

Génie Logiciel Orienté Objets, Philippe Collet, Master 1 Informatique, 2007-2008

Processus de conception de SI M1 MIAGE - SIMA - 2005-2006 Yannick Prié UFR Informatique - Université Claude Bernard Lyon 1

Méthodes de conduite de projet, Tester, optimiser, structurer ses applications Jean David Olekhnovitch, [jd@olek.fr](mailto:jd@olek.fr) - [www.olek.fr](http://www.olek.fr)

Les cours IBM sur le RUP

Conduite de projet, Méthode d'analyse et de conception, Processus unifié, G. Picard, SMA/G2I/ENS Mines Saint-Etienne [gauthier.picard@emse.fr](mailto:gauthier.picard@emse.fr), Octobre 2009

Processus Unifié : [www2.lifl.fr/~clerbout/.../Cours4-ProcessusUnifie.pdf](http://www2.lifl.fr/~clerbout/.../Cours4-ProcessusUnifie.pdf)

*eXtreme Programming & Scrum Practices, Embrace Change, Naresh Jain*

*La Gestion de projets (informatiques) et la question de la collaboration; G. Beauvallet, Octobre 2007, telecom paris.*