

Méthodologie de la production d'applications

Mireille Blay-Fornarino
septembre 2014

<https://mbf-iut.i3s.unice.fr/>

Que pensez-vous de ?

Cher client,

nous étions très occupé et n'avons pas envoyé à temps notre reporting, nous en sommes désolé.

Malheureusement, au vu du temps restants et des projets parallèles à réaliser, nous pensons abandonner des versions planifiées dans les livrables.

Celles qui seraient impactées sont la 0.8 et la 0.7.

Concrètement l'objectif de notre groupe sera toujours atteint mais l'intégration finale avec les autres groupes sera fortement compromise et une couverture complète de tests ne pourra pas être assurée côté serveur (les tests côté client sont déjà abandonnés).

Nous aurions voulu respecter notre cahier des charges initiales ainsi que notre liste de livrables. Malheureusement le manque de temps joue toujours en notre désavantage.

Que pensez-vous de ?

Cher client,

Vous nous avez commandé le 1er janvier une voiture neuve. Comme beaucoup d'autres clients nous ont aussi commandé des voitures, nous avons pris du retard. Nous en sommes désolés. Cependant la voiture coûte toujours le même prix.

Nous comptons finir à temps l'assemblage, avec la plupart des options que vous avez demandées, sauf certaines (référez-vous à la liste des versions prévues pour savoir lesquelles seront manquantes).

Cependant, nous n'avons pas encore fait les essais moteur, et nous n'aurons pas le temps de faire les essais routiers non plus.

Objectifs

- ➔ O1 : Analyser les besoins pour la conception et l'évolution des systèmes d'information dans une organisation
- ➔ O2 : Organiser et gérer un projet
- ✓ Approche par les méthodes Agiles

Organisation des TDS

- ➔ 1 étude de cas : **Système de Transport Intelligent**
- ➔ 4 phases, avec des livrables à chaque fin de phase.
 1. Analyse => 19 septembre
 2. Conception et développement par 2 Sprints => 14 novembre
 3. Retrospective et qualité => 22 novembre
 4. Intégration => Fin du module fin décembre
- ➔ Des sous-groupes de 4 étudiants avec un chef de projet par sous-groupe.

Evaluation du module

- Un examen (1,25)
- Une note de TD obtenue par moyenne pondérée des différents livrables. (1,5)
 - Voir le site pour les détails
- Une note de Participation aux TDs (0,25)

Outils associés





- La forge de l'IUT étendu avec un KANBAN
- Visual Paradigm pour la modélisation
- Eclipse avec plusieurs plugins pour les développements.

Méthodes de conduite de projet

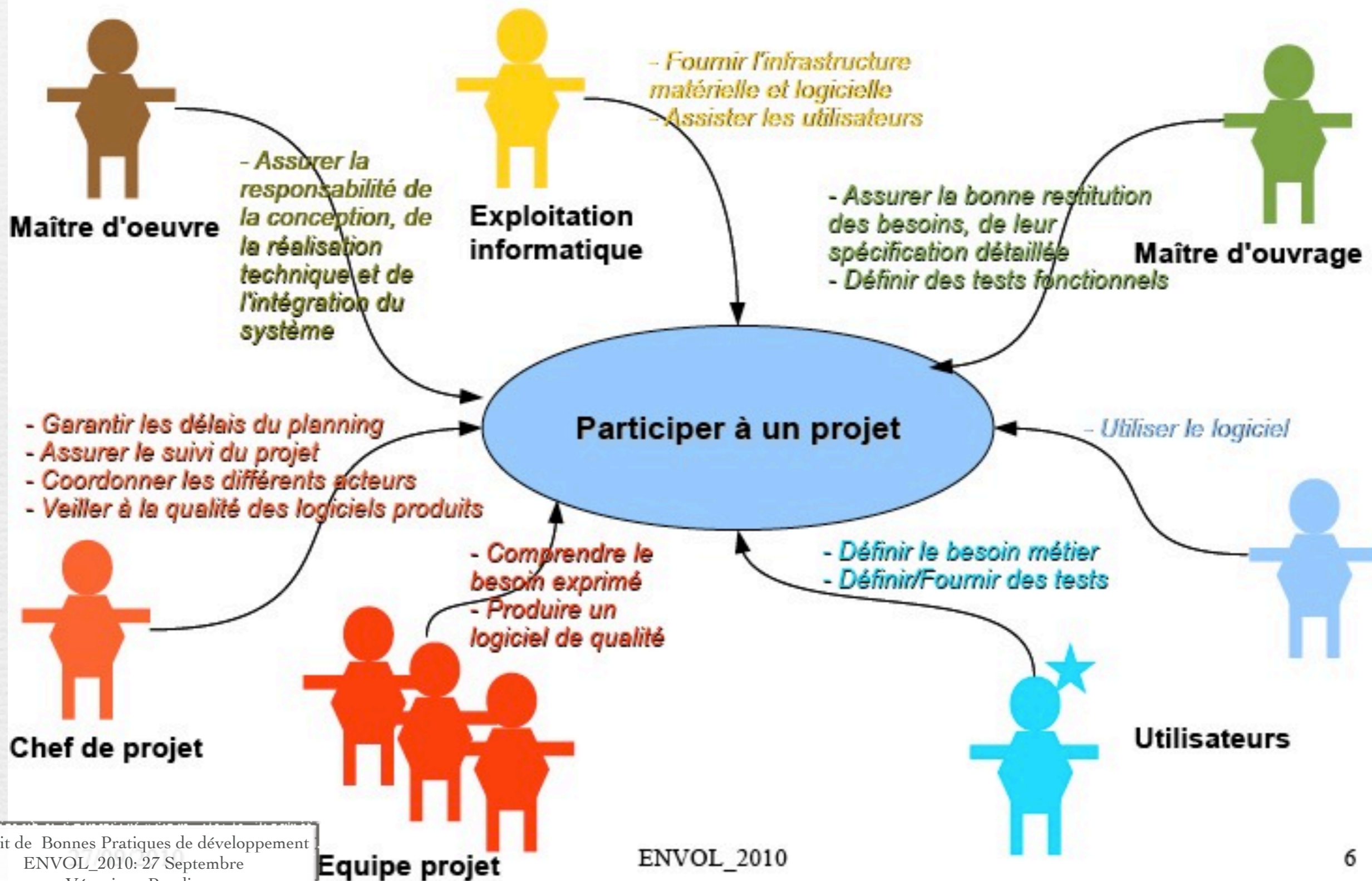
Mireille Blay-Fornarino
IUT
blay@unice.fr

Site web du module : <https://mbf-iut.i3s.unice.fr/>

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



Discussion: symptômes et causes profondes

- * Quels sont les symptômes de problèmes de développement de logiciels?
- * Quelles sont les causes profondes de ces symptômes ?

Symptômes de problèmes de développement logiciel

- * Non réponse aux besoins
- * Non couverture des exigences
- * Mauvaise intégration
- * Difficile à maintenir
- * Découverte tardive de défauts
- * Mauvaise qualité en terme de service rendu ou d'expérience utilisateur
- * Mauvaises performances lors de la charge
- * Aucun effort de coordination de l'équipe
- * Problème de construction des versions finales

Causes des symptômes des problèmes de développement logiciel

- * Prise en compte des exigences insuffisantes
- * Communications ambiguës
- * Architectures fragiles
- * Complexité écrasante
- * Incohérences non détectées
- * Faiblesse des tests
- * Evaluation subjective
- * Développement en cascade
- * Absence de contrôle du changement
- * Pas d'automatisation

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(1/4)

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(2/4)

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 (3/4)



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(4/4)

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.

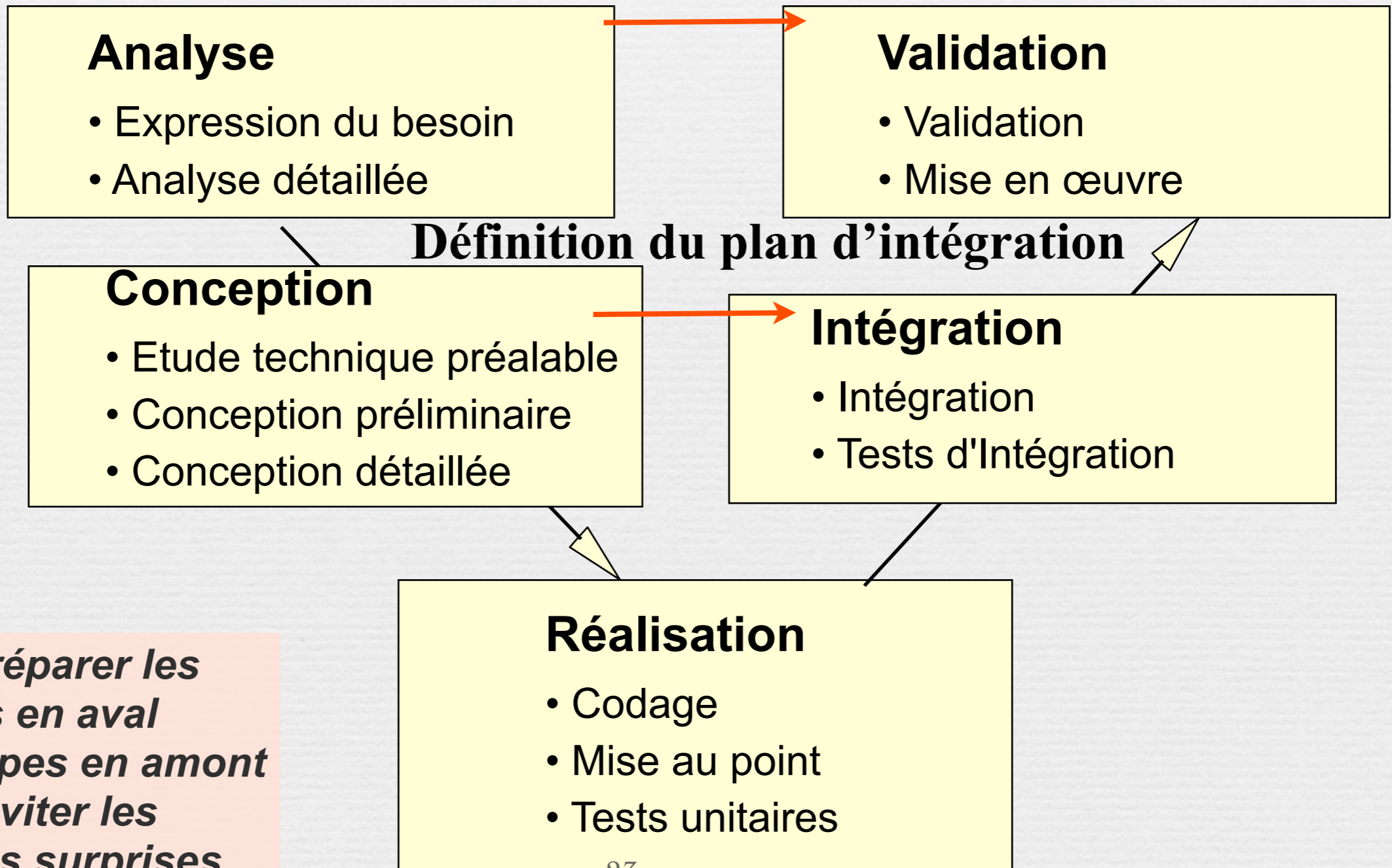
Seulement applicable pour les projets qui sont bien compris et maîtrisés

Modèle en « V »

→ Cycle de vie normalisé AFNOR

Variante du cycle en « cascade »

Définition des tests



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

Modèle en « V »

Le cycle en V

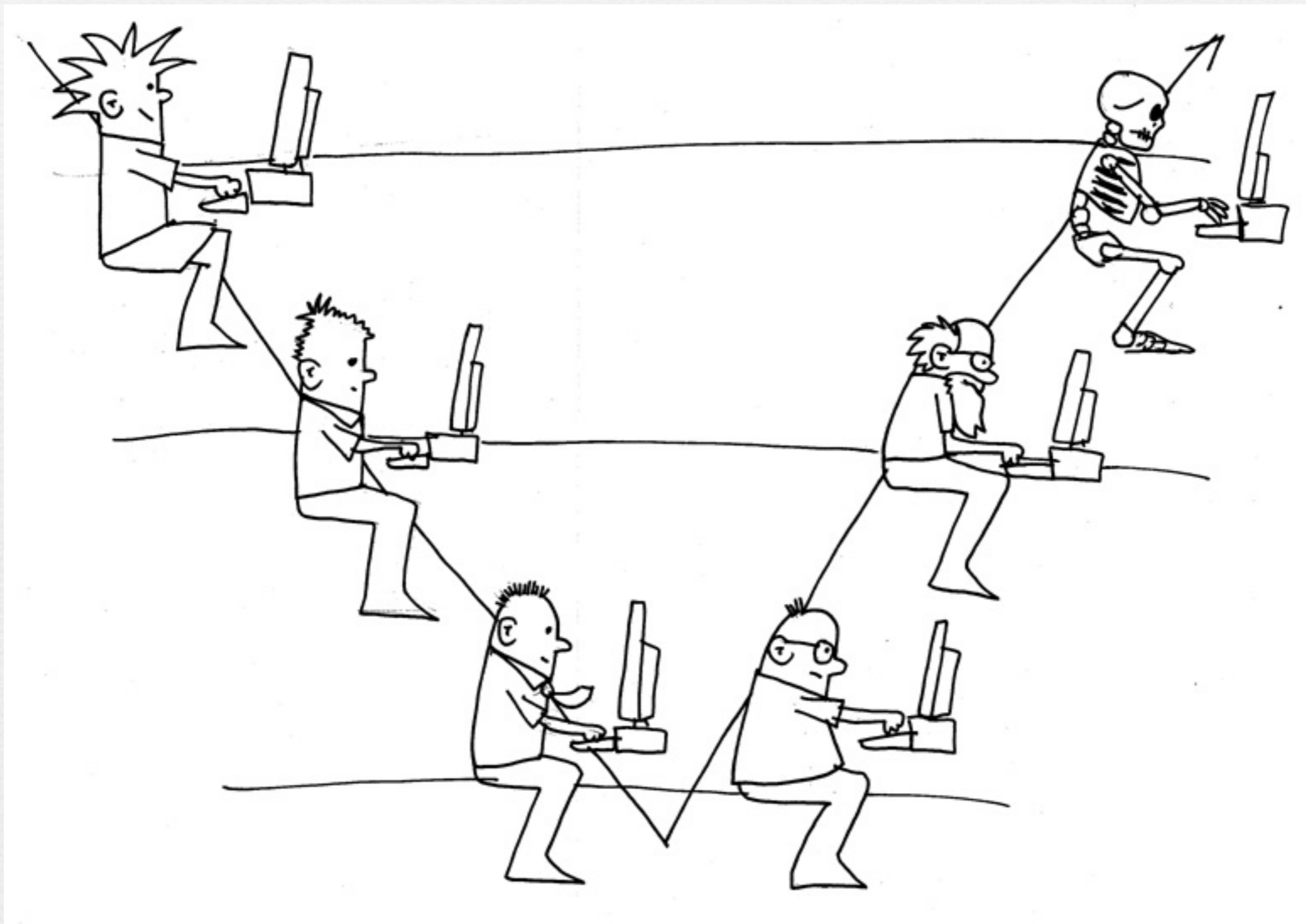
- permet une meilleure anticipation
- présente des tests bien structurés

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...



Comment le client a exprimé son besoin



Comment le chef de projet l'a compris



Comment l'ingénieur l'a conçu



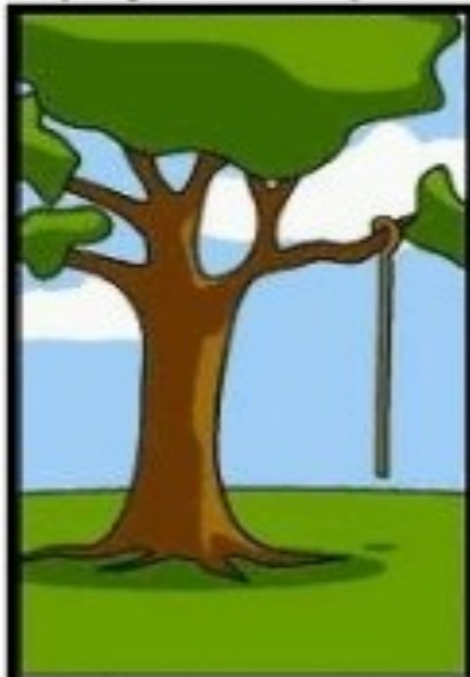
Comment le programmeur l'a écrit



Comment le responsable des ventes l'a décrit



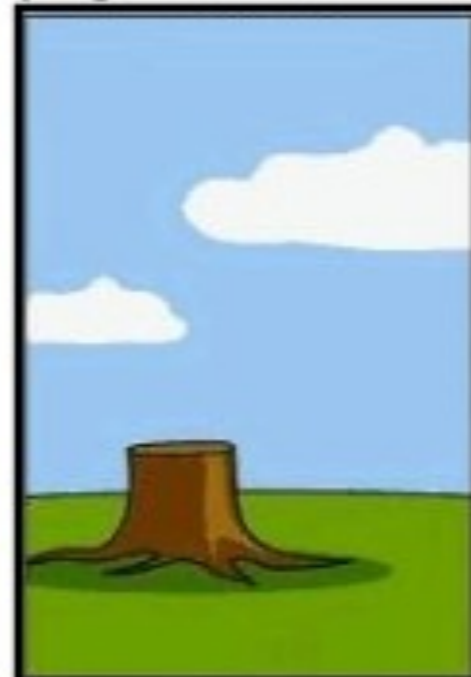
Comment le projet a été documenté



Ce qui a finalement été installé



Comment le client a été facturé



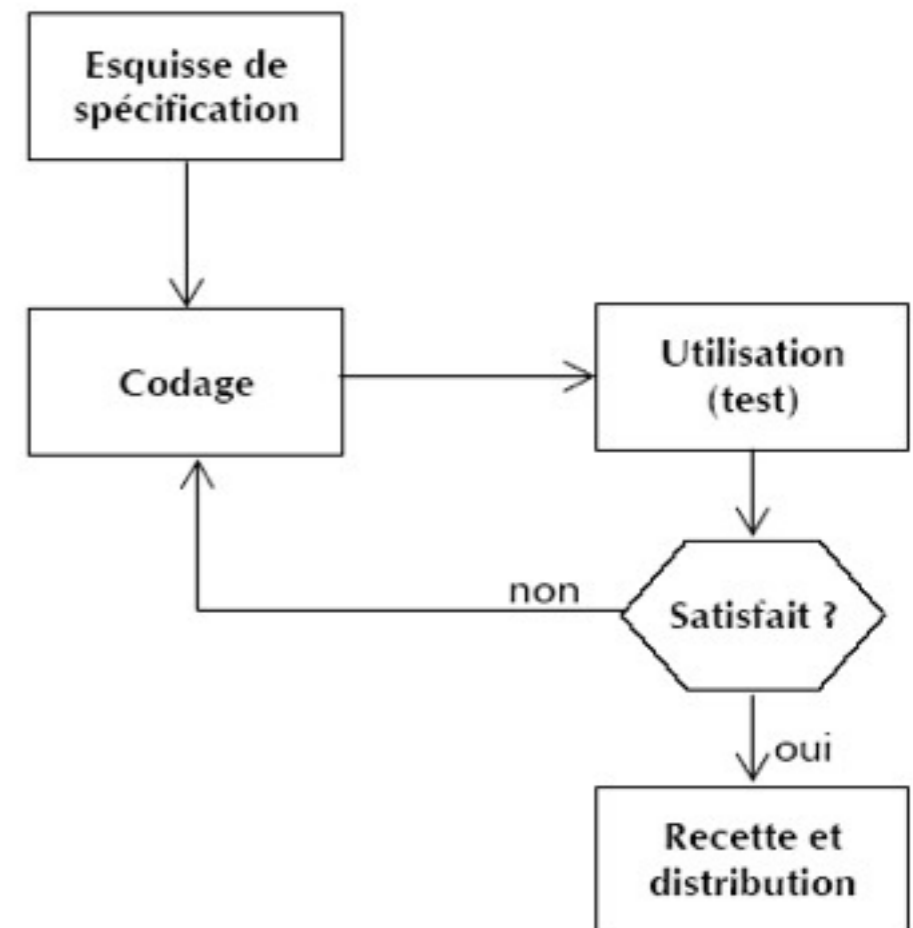
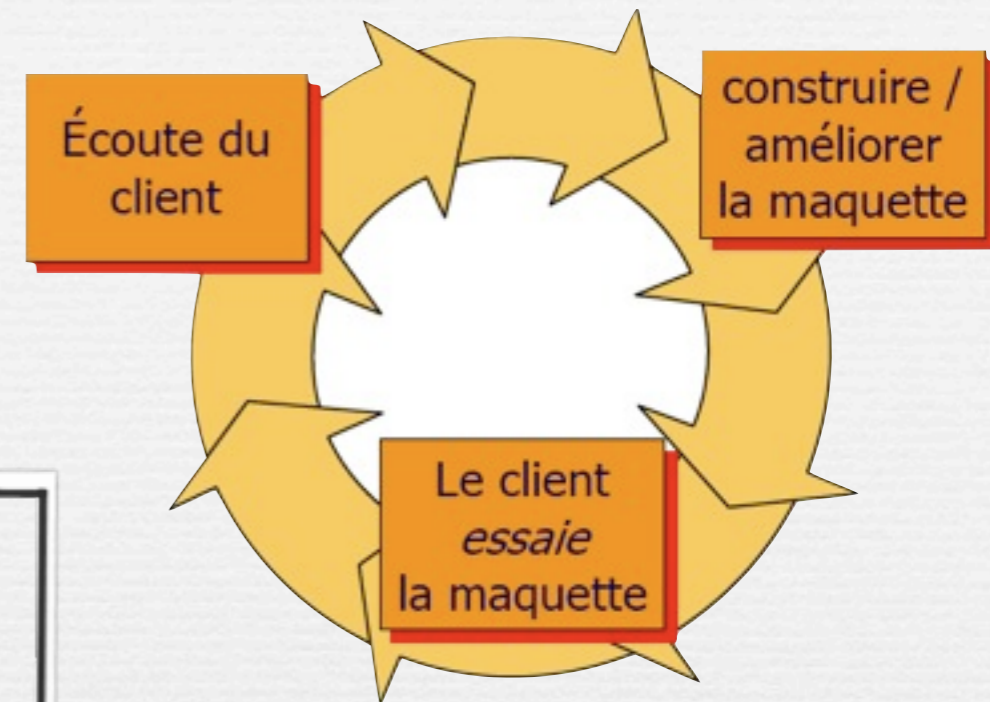
Comment la hotline répond aux demandes



Ce dont le client avait réellement besoin

Cycle exploratoire Prototypage RAD*

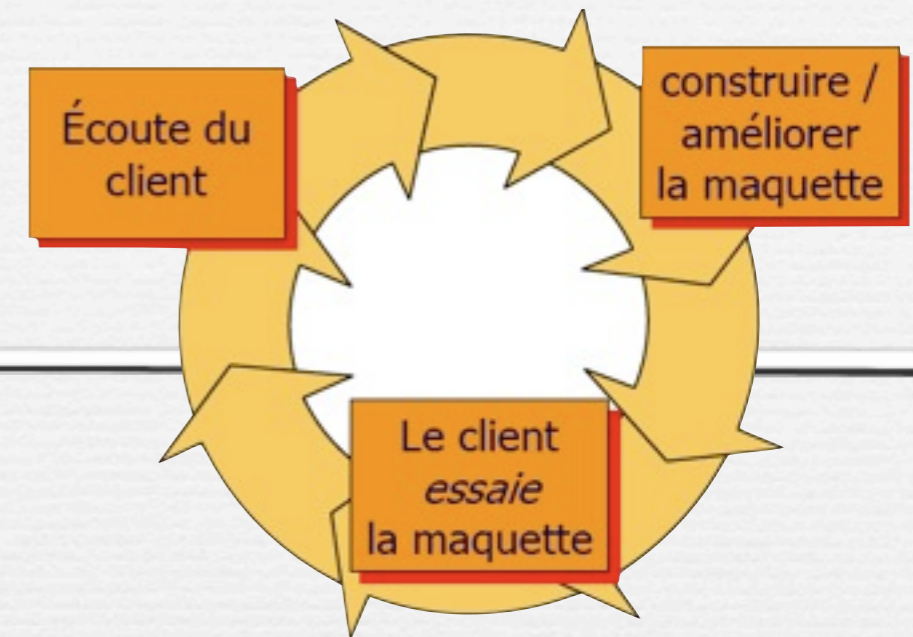
- Discuter et interagir avec l'utilisateur
- Vérifier l'efficacité réelle d'un algorithme
- Vérifier des choix spécifiques d'IHM
- Souvent utilisé pour identifier les besoins
 - Prototype jetable (moins de risque ?)
- Souvent implémenté par des générateurs de code
 - Prototype évolutif



Cycle exploratoire, Prototypage, RAD*

MAIS

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



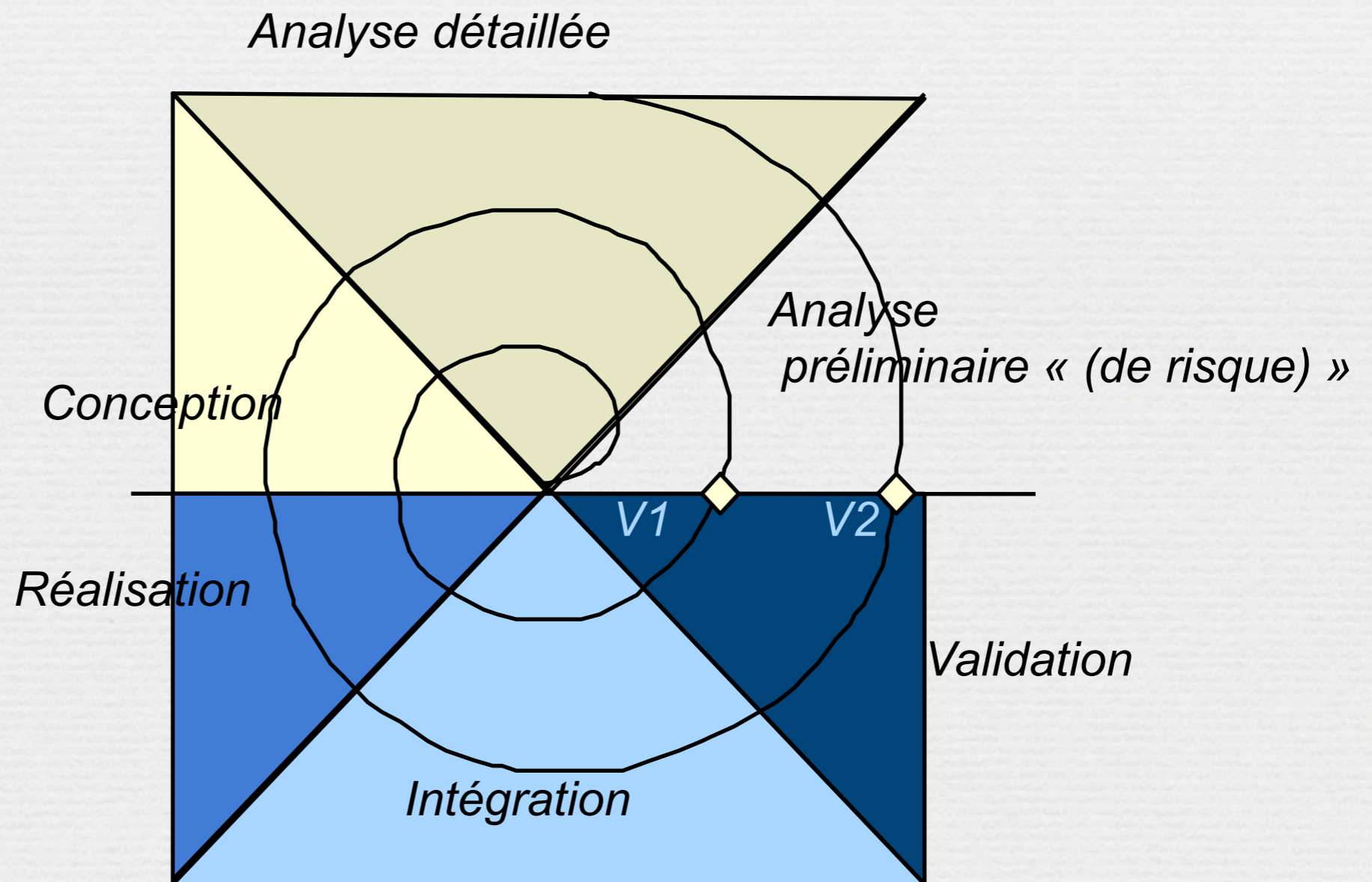
Admissible tel quel pour

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

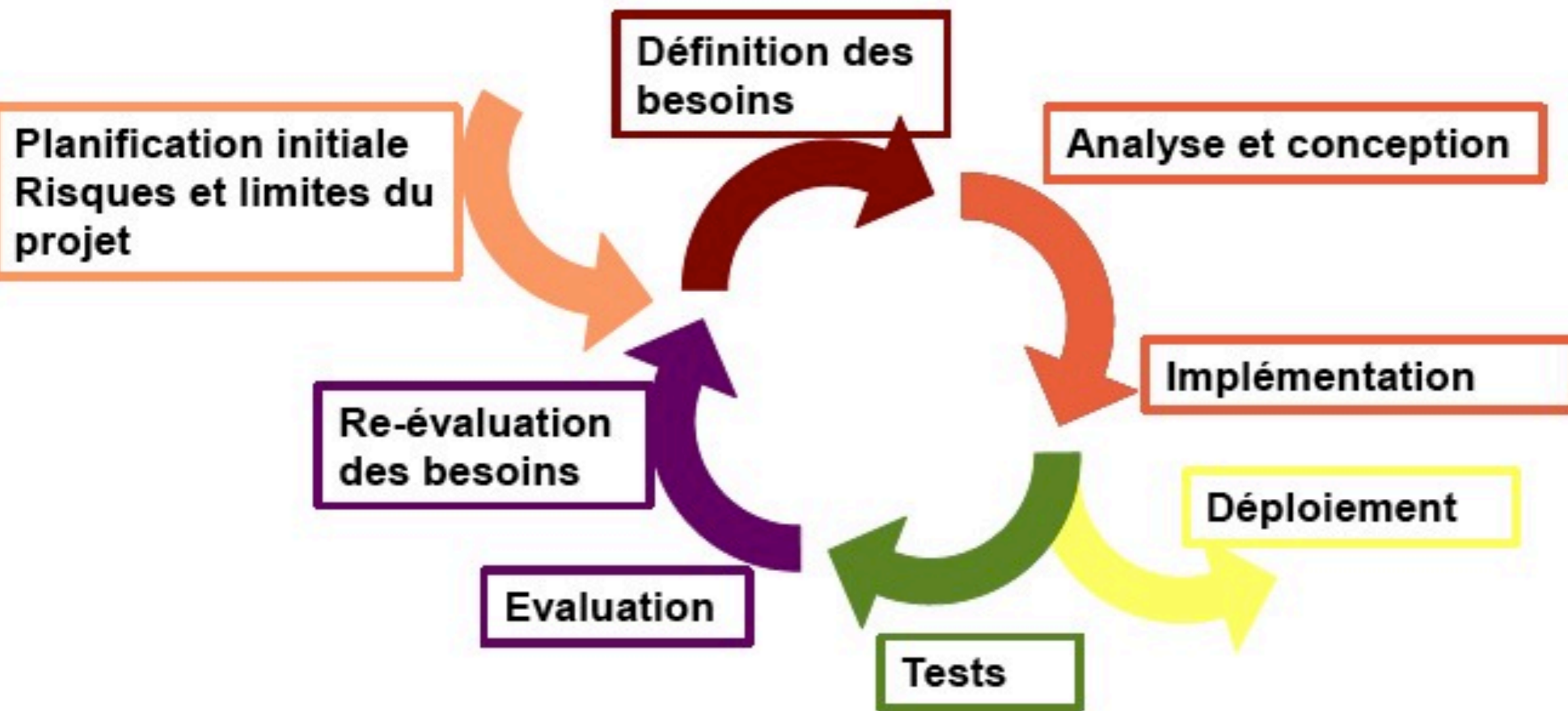
Cependant, retour en force dans les processus « agiles » mais sous une forme beaucoup plus élaborée

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



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

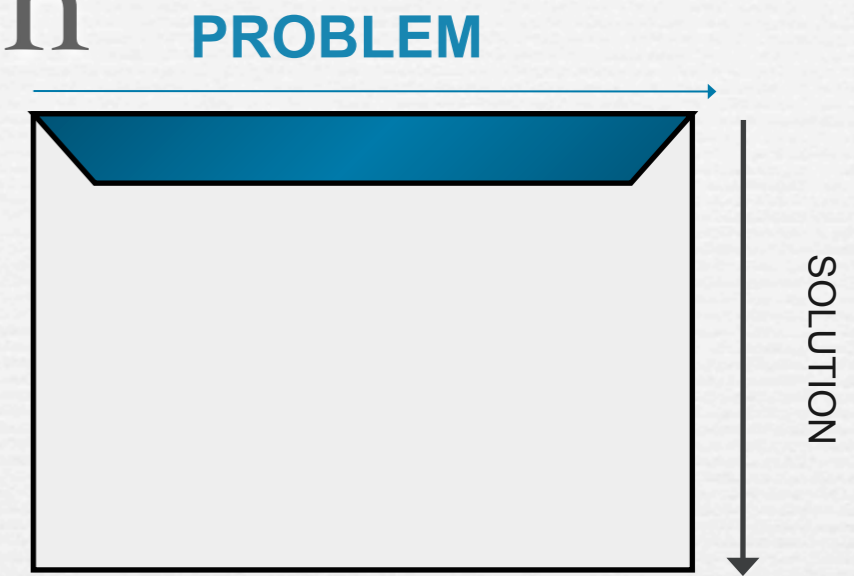
- ➔ Moins simple à manager
 - 🌐 difficile à gérer en situation contractuelle
 - 🌐 mal contrôlé => on retombe dans le *hacking*

À la base de tous les processus « agiles »

Stratégies d'itération

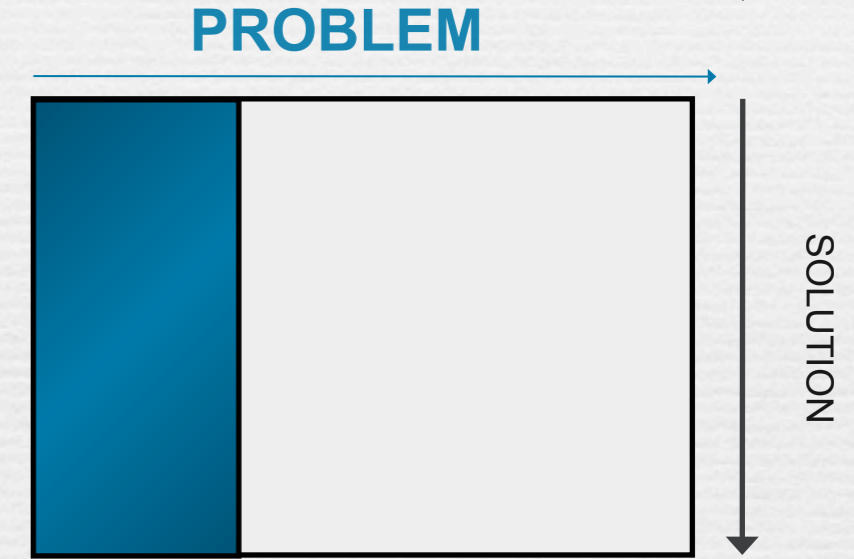
Large et peu profonde

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



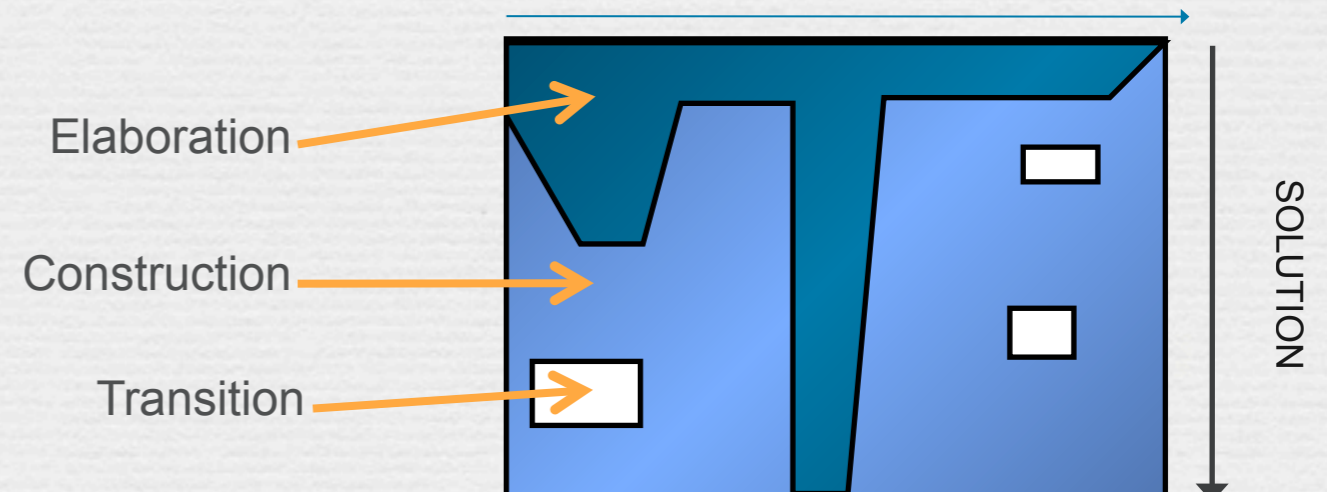
Etroite et profonde

- ▶ 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

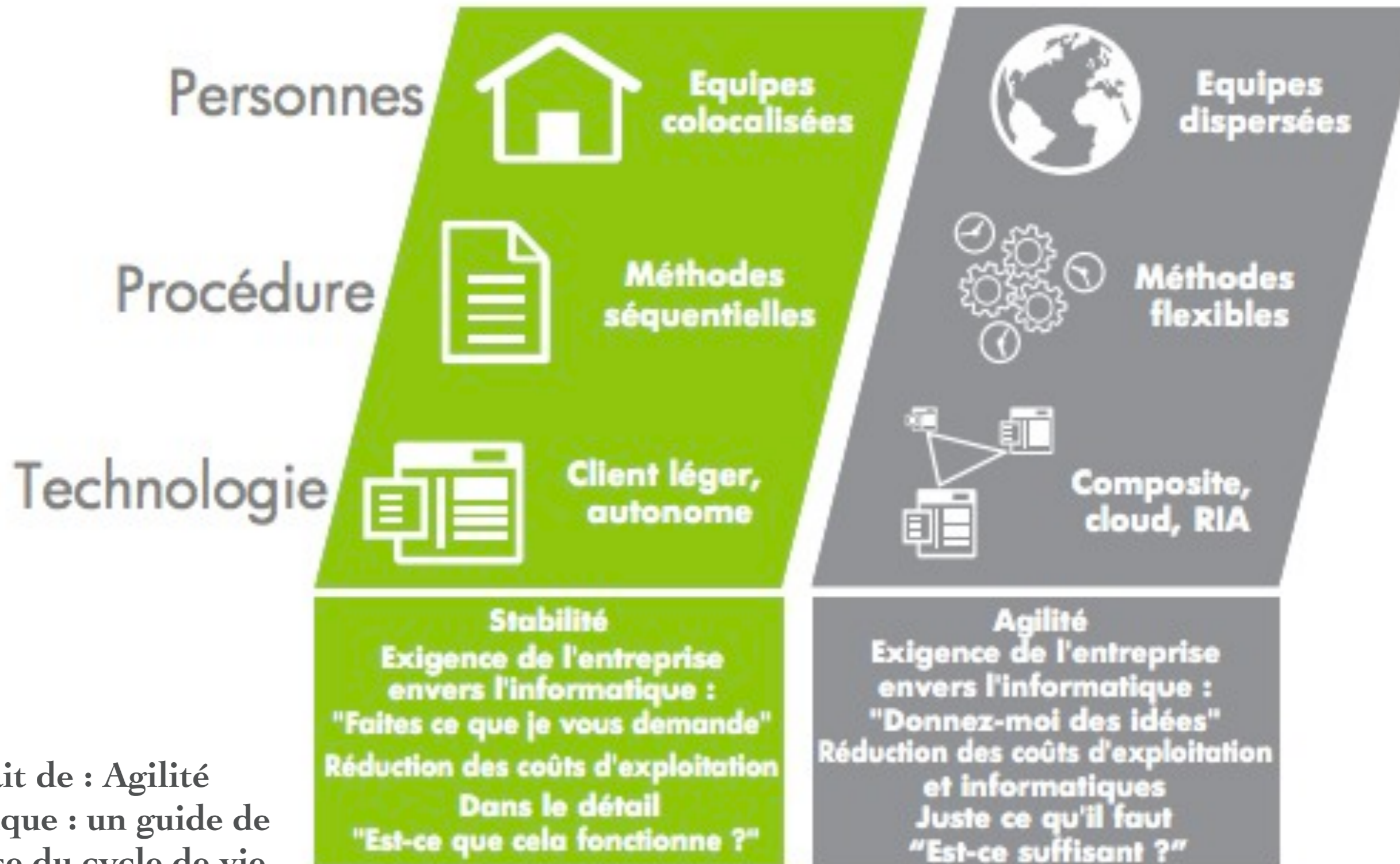
- Un artefact est un élément produit ou modifié dans le cadre d'un processus de développement
- Il peut s'agir d'un document, un diagramme, un compte rendu de réunion...
- ...mais aussi un code source, un écran, une base de données...

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 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]

III. Pratiques contemporaines

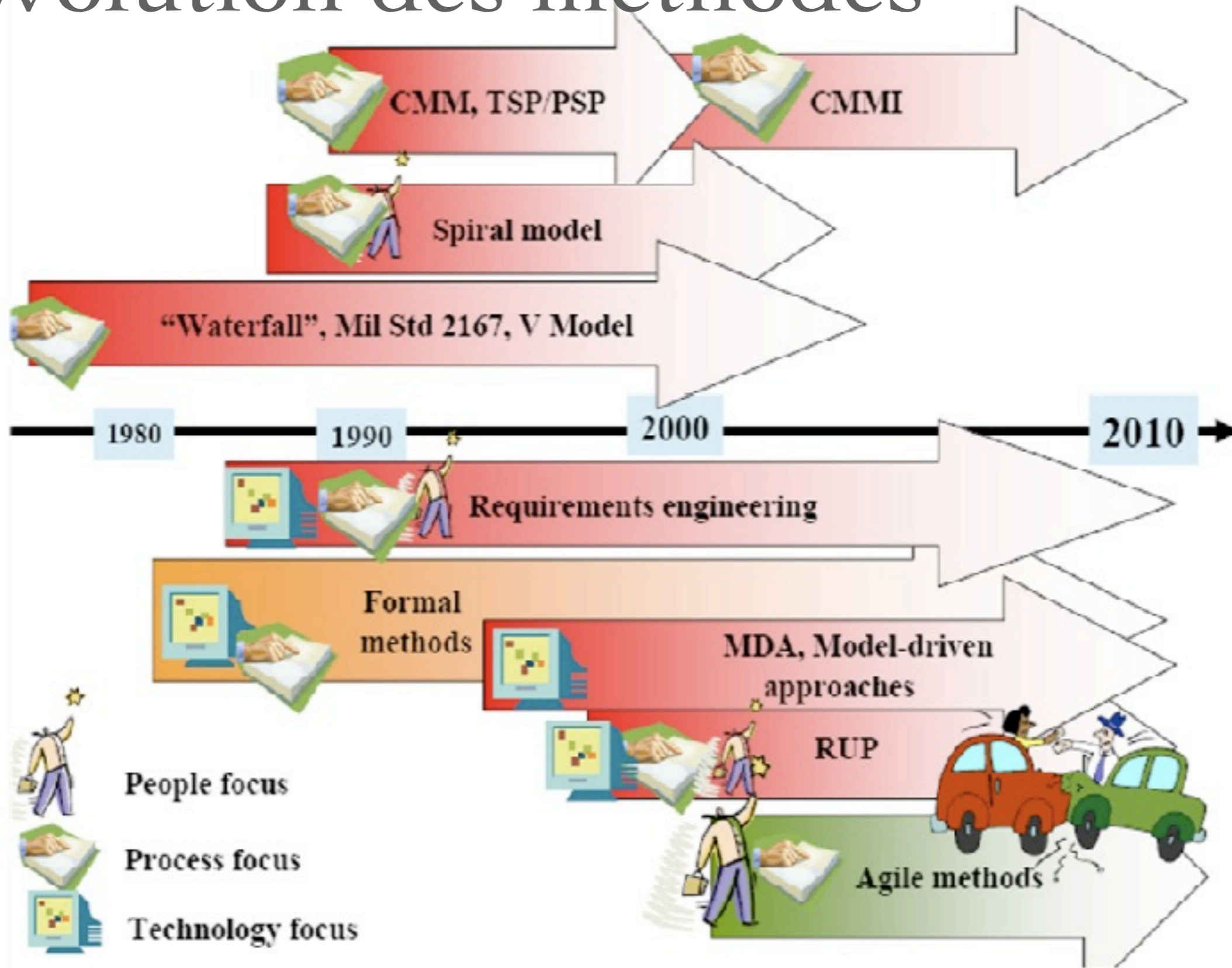
09:07:25 09:54:01

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

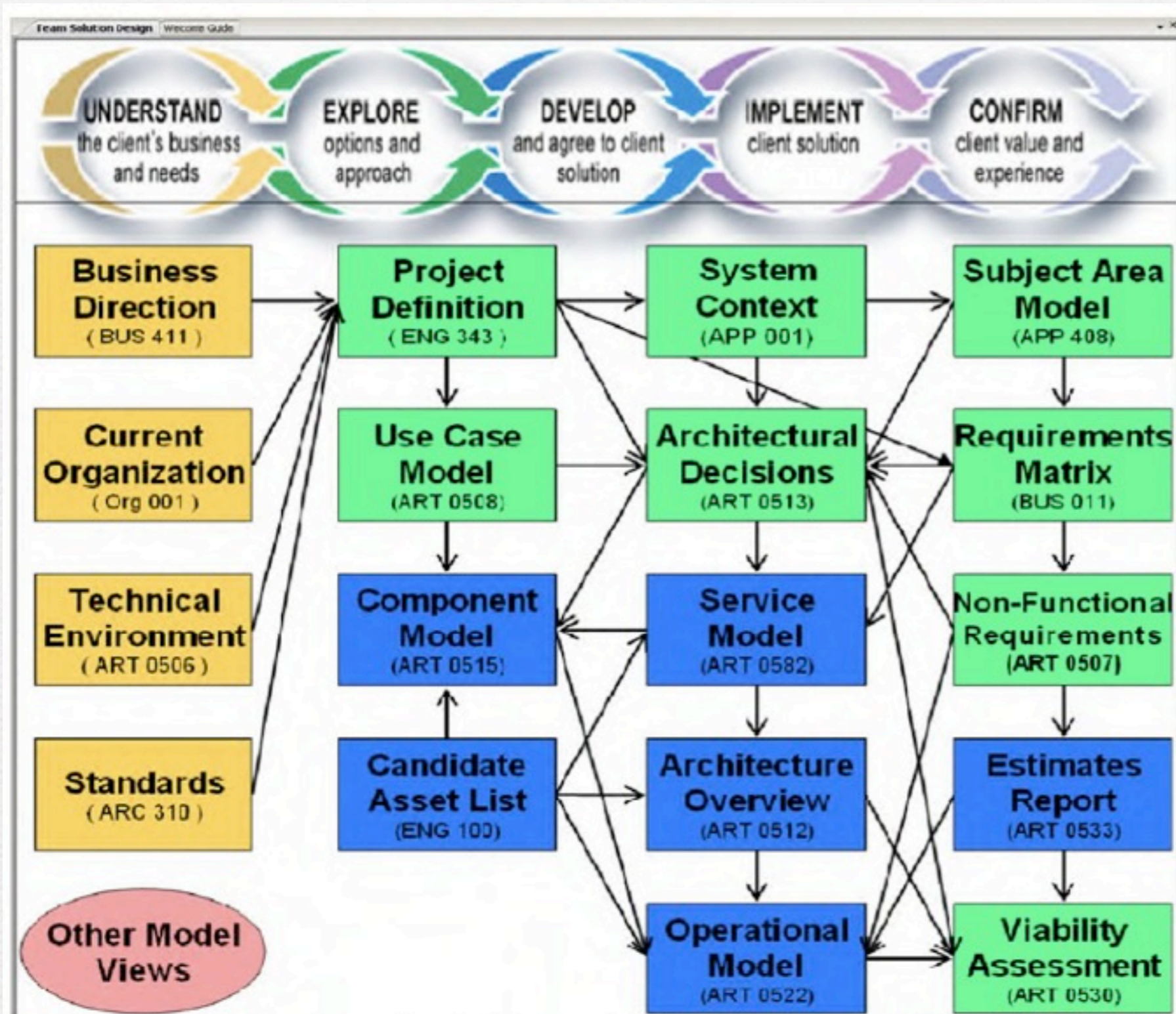


<http://www.entreprise-agile.com/HistoAgile.pdf>

Figure 4. — Vision globale (source NCR 2007)

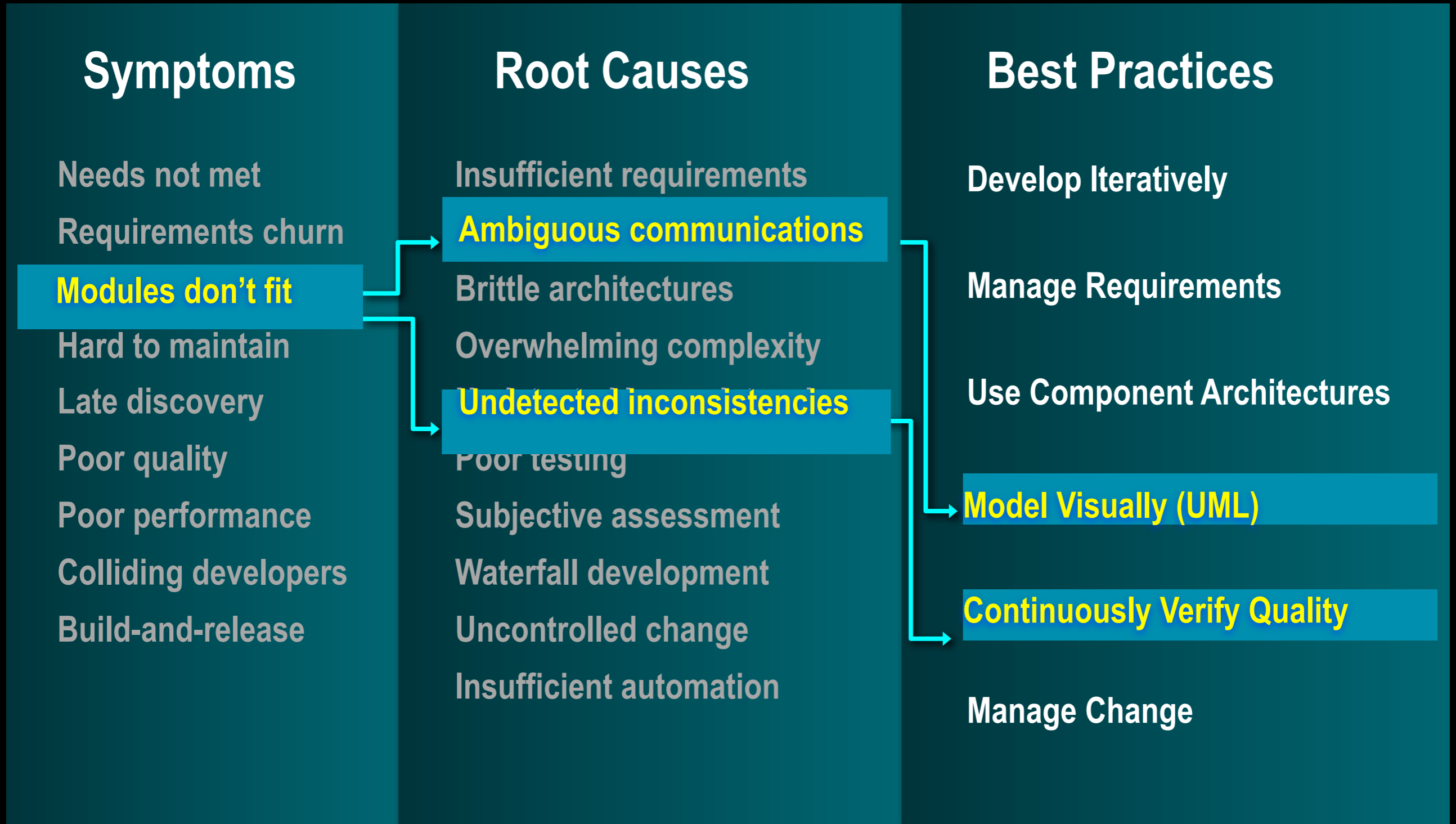
Évolution des méthodes

IBM methodology : IBM Team Solution Design.



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

Trace Symptoms to Root Causes



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 - 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, Octobre 2009

Processus Unifié : 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.