



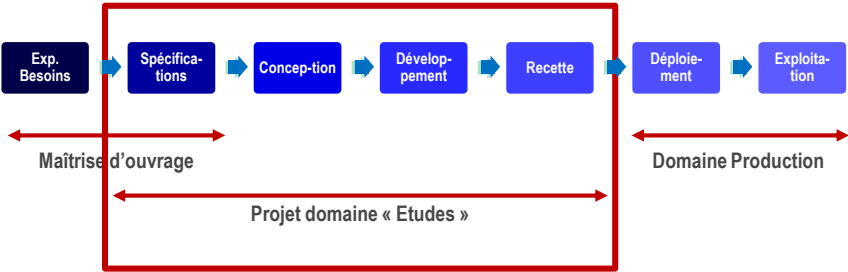
1

Sommaire

- Les disciplines standards d'un projet: spécifications, conception, développement, recette,...
- Les méthodes « classiques » (Cascade, V, ...).
- Les méthodes « itératives » (RUP, 2TUP).
- Les méthodes Agile.
- Exemples.

2

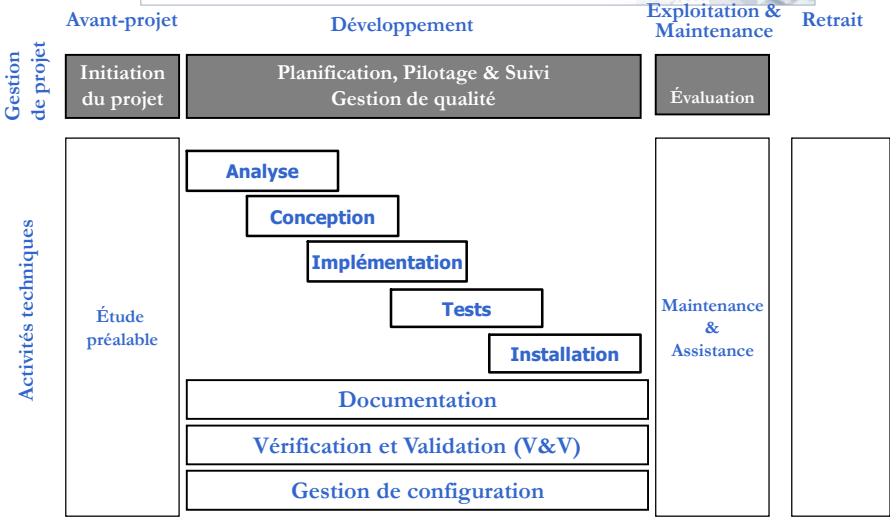
Les disciplines d'un projet: vue d'ensemble



Le cycle de vie du projet ne concerne que la réalisation du produit informatique

3

Cycle de vie du logiciel (CVL)





LES DISCIPLINES D'UN PROJET

5

Les disciplines d'un projet

Etude de la faisabilité

- Déterminer si le développement proposé vaut la peine d'être mis en œuvre, compte tenu de attentes et de la difficulté de développement.

⇒ Etude de marché : déterminer s'il existe un marché potentiel pour le produit.

6

Les disciplines d'un projet

SPÉCIFICATIONS

- Spécifications = manière dont l'application va implémenter les besoins (étape précédente = expression de besoin)
- Définir le périmètre de l'application :
 - au niveau fonctionnel : activités métier à informatiser (souvent sous UML)
 - Navigation dans l'application
 - Description détaillée des écrans : maquettes, règles de gestion écran
 - Organisation des traitements
 - Description détaillée des traitements « batch »
 - Paramétrage et contrôle des traitements
 - au niveau technique : interfaces techniques du projet (normes, flux de données, interfaces matérielles (info indus, imprimantes, équipements mobiles, ...))

7

Les disciplines d'un projet

SPÉCIFICATIONS : LIVRABLES

- Spécifications Fonctionnelles Générales (**SFG**) - option
- Spécifications Fonctionnelles Détaillées (**SFD**) - obligatoire
- Spécifications Techniques Générales (**STG**) - option
- Spécifications Techniques Détaillées (**STD**) - option

8

Les disciplines d'un projet

ARCHITECTURE TECHNIQUE

- Peut être optionnelle (si l'architecture est déjà en place et maîtrisée)
- Nécessaire s'il faut industrialiser les développements sur de gros projets
- Tâches :
 - Sélectionner avec soin les outils de développement
 - Récupérer un framework existant ou le créer
 - Mettre en place des fonctions standard pour les DAO
 - Identifier et contourner les contraintes techniques (ex : projets J2EE : comportements différents selon les navigateurs)
 - Réaliser des modèles d'implémentation qui seront largement utilisés, exemple :
 - Écran de saisie, navigation standard dans l'écran
 - Fenêtres standard d'erreurs et gestion standard des retours d'erreur

9

Les disciplines d'un projet

ARCHITECTURE TECHNIQUE : LIVRABLES

DOSSIER D'ARCHITECTURE LOGICIELLE

Les disciplines d'un projet

CONCEPTION TECHNIQUE

- Conception Technique = Passerelle entre SFD et Code
- Point d'entrée des développeurs
- Tâches :
 - Analyser les SFD et détecter d'éventuelles anomalies
 - Identifier les composants à développer (classes, écrans, services métier, ...)
 - Formaliser la navigation entre les écrans
 - Formaliser sous forme de pseudo code les traitements

CONCEPTION TECHNIQUE : LIVRABLES

DOSSIER DE CONCEPTION TECHNIQUE

11

Les disciplines d'un projet

DÉVELOPPEMENT

TÂCHES

- Développement, dans le respect :
 - Des spécifications
 - De la conception applicative
- Tests usine
 - Tests Unitaires
 - Tests d'intégration
 - Tests fonctionnels, Tests IHM
 - Tests de montée en charge, de robustesse, de sécurité,

12

Les disciplines d'un projet

DÉVELOPPEMENT : LIVRABLES

- Code source
- Exécutables, dans un package de livraison complet
- Documentation du code (si intérêt)
- Documentation utilisateur (dérive spécifications fonctionnelles)
- Documentation des tests

13

Les disciplines d'un projet

RECETTE

- But : Passage des tests avant mise en production
- Plusieurs niveaux :
 - **VT** : Validation technique (non contractuelle) : porte sur des sous ensembles de l'application
 - **VA** : Vérification d'Aptitude (ou **VABF** : VA au Bon Fonctionnement) : opération contractuelle : vérifier que l'application finale respecte scrupuleusement les spécifications. Cette validation déclenche la mise en production
 - **VSP** : Vérification sur Site Pilote (optionnel) : correction de bugs éventuels en production sur une extraction des sites client
 - **VSР** : Vérification pour Service Régulier (contractuel) : correction de bugs en production sur tous les sites clients. 14
Déclenche le début de la garantie contractuelle

Les disciplines d'un projet

RECETTE: TESTS

Essayer le logiciel sur des données d'exemple pour s'assurer qu'il fonctionne correctement :

- **Tests unitaires** : faire tester les parties du logiciel par leurs développeurs
- **Tests d'intégration** : tester pendant l'intégration.
- **Tests de validation** : pour acceptation par l'acheteur.
- **Tests système** : tester dans un environnement proche de l'environnement de production.

15

Les disciplines d'un projet

RECETTE: TESTS

- **Tests Alpha** : faire tester par le client sur le site de développement.
- **Tests Bêta** : faire tester par le client sur le site de production.
- **Tests de régression** : enregistrer les résultats des tests et les comparer a ceux des anciennes versions pour vérifier si la nouvelle n'en a pas dégradé d'autres.

16

Les disciplines d'un projet

RECETTE : LIVRABLES

- Bordereaux de livraison (chaque étape)
- PV (procès verbal) de recette (chaque étape)
 - « sans réserves »
 - « avec réserves » : bugs à corriger (mineurs)

17

Les disciplines d'un projet

MAINTENANCE

- Mettre à jour et améliorer le logiciel pour assurer sa pérennité.
- Pour limiter le temps et les coûts de maintenance, il faut porter ses efforts sur les étapes antérieures

18



LES DOCUMENTS COUTANTS

19



Cahier des charges

Description **initiale** des **fonctionnalités désirées**,
généralement écrite par l'utilisateur.

20

Calendrier du projet

- Ordre des différentes tâches.
- Détails et ressources qu'elles demandent

21

Plan de tests des logiciels

- Décrit les procédures de tests appliquées au logiciel pour contrôler son bon fonctionnement.

⇒ Tests de validation : tests choisis par le client pour déterminer s'il peut accepter le logiciel.

22

Plan d'assurance qualité

Décrit les activités mises en œuvre pour garantir la qualité du logiciel.

23

Manuel utilisateur

Mode d'emploi pour le logiciel dans sa version finale.

24

Code source



Code complet du produit fini.

25

Rapport des tests



Décrit les tests effectués et les réactions du système.

26

Rapport des défauts

- Décrit les comportements du système qui n'ont pas satisfait le client.
- Il s'agit le plus souvent de défaillances du logiciel ou d'erreurs.

27

Documents produits / cycle de vie

DOCUMENTS

- Manuel utilisateur final.
- Conception architecturale.
- Plan d'assurance qualité.
- Code source.
- Cahier des charges.
- Plan de test.
- Manuel utilisateur préliminaire

PHASE DE PRODUCTION

- Implémentation.
- Conception.
- Planification.
- Implémentation.
- Faisabilité.
- Spécification.
- Spécification

28

Documents produits / cycle de vie

DOCUMENTS

- Conception détaillée.
- Estimation des coûts.
- Calendrier du projet.
- Rapport des tests.
- Documentation.

PHASE DE PRODUCTION

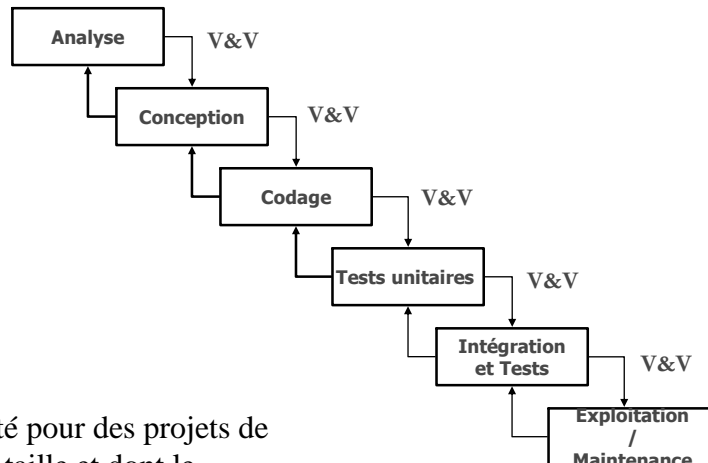
- Conception.
- Planification.
- Planification.
- Tests.
- Implémentation.

29

LES MÉTHODES «CLASSIQUES»

30

Modèle en cascade



Adapté pour des projets de petite taille et dont le domaine est bien maîtrisé.

31

Modèle en cascade

- Chaque étape du cycle est caractérisée par des ACTIVITES dont le but est d'élaborer un ou des produits intermédiaires
- Chaque fin d'étape est matérialisée par un événement, où s'exerce une activité de contrôle (VERIFICATION et VALIDATION) afin d'éliminer au plutôt les anomalies des produits réalisés. Le passage à l'étape suivante est conditionné par le résultat de contrôle (acceptation, rejet, ajournement)
- Autant que possible, les retours en arrière sur les étapes précédentes se limitent à un retour sur l'étape immédiatement antérieure

Vérification & Validation (V&V)

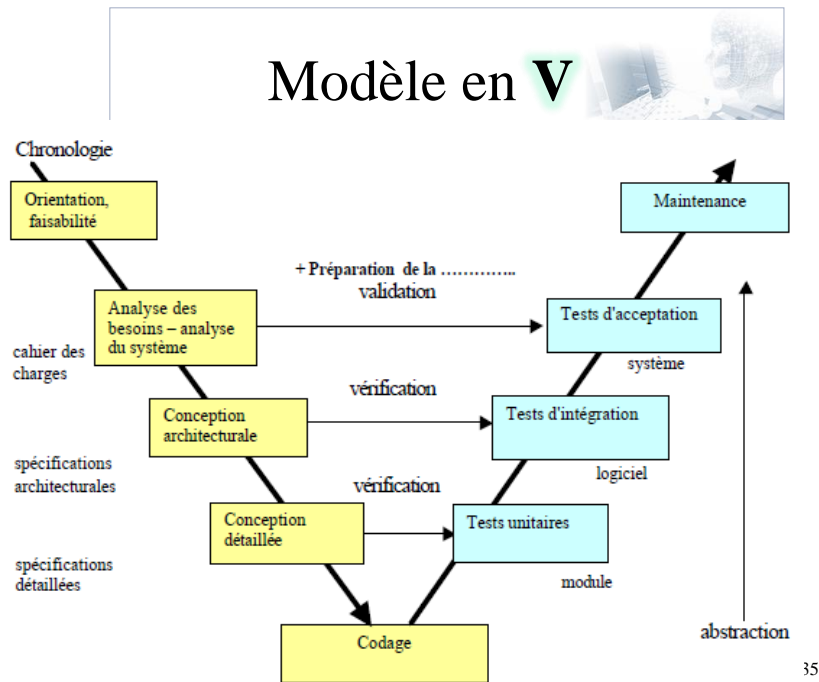
Vérification : « Est ce que nous construisons bien le produit? »

Validation : « Est ce que nous construisons le bon produit? »

- Validation des besoins
 - => Vérification du respect des spécifications du logiciel et des besoins
 - => Démonstration que les performances réelles sont conformes aux spécifications de performances
- Vérification de tous les éléments constituant la fourniture du logiciel : code, rapports, manuels, documentation, jeux de tests, etc.

Modèle en cascade

- **Avantages**
 - Facile à comprendre
 - Le plus utilisé dans l'industrie
 - **Inconvénients**
 - Approche purement séquentielle et « simpliste »
 - Il est rare que le client puisse fournir toutes les spécifications dès le début du projet.
 - Le client ne reçoit pas les résultats concrets pendant le développement du logiciel
 - Fort coût de correction
 - Etapes différentes: personnes différentes (risques d'incohérences)
- Cause de l'échec de nombreux projets



35



- Du côté gauche : flux de développement.
- Du côté droit : Flux de vérification et validation.

Adapté pour des projets dont le domaine est bien maîtrisé.

36

Modèle en V : Principe

- Validation systématique de chaque étape
- Décomposition du projet en parties
- Deux sortes de dépendances entre étapes :
 - **Traits continu** : correspondent à l'enchaînement et à l'itération éventuelle du modèle en cascade, les étapes se déroulent séquentiellement en suivant le V de gauche à droite
 - **Traits non continus** : Une partie des résultats de l'étape de départ est utilisée directement par l'étape d'arrivée.
Par exemple : à l'issue de la conception, le protocole d'intégration et les jeux de tests d'intégration doivent être complètement décrits.

Modèle en V

- **Avantages :**
 - Avec toute décomposition doit être décrite la recombinaison, toute description d'un composant est accompagnée de tests.
 - Principe qui évite un autre bien connu de la spécification: on énonce une propriété qu'il est impossible de vérifier objectivement une fois le logiciel réalisé.
 - L'obligation de concevoir les jeux de tests et leurs résultats oblige à une réflexion et à des retours sur la description en cours.
 - Les étapes de la branche droite du V peuvent être mieux préparées et planifiées.
- **Inconvénients :**
 - Pas de résultats intermédiaires dont on peut discuter avec le client

Avantages des méthodes classiques

- Permettent de connaître « dès le départ » l'ensemble des fonctionnalités d'une application → Rassurant
- Fournissent une documentation avancée du logiciel.
- Simple à appréhender → Rassurant pour beaucoup de décideurs.

39

Inconvénients des méthodes classiques

Difficultés à postériori :

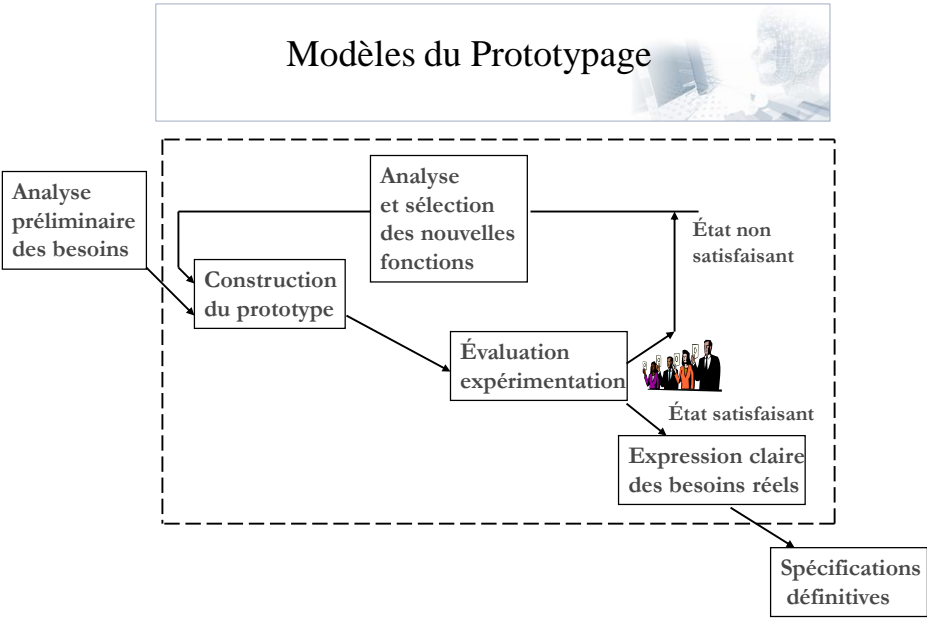
- Savez-vous aujourd'hui définir dans le détail et par avance la maison de vos rêves ??
 - Réponse oui en théorie, et **NON dans la pratique** : au mieux vous en aurez une idée plus ou moins précise (votre 3^{ème} maison sera la bonne!)
 - Il en va de même avec la conception d'une application.
 - Le besoin réel se construit au fur et à mesure de son utilisation.
- Cela implique la plus part du temps :
 - Une application mal pensée par rapport aux besoins réels des utilisateurs
 - Un dépassement des délais et des coûts établis au départ du projet

40



LE PROTOTYPAGE

41



Le prototypage

- **Prototype : version d'essai du logiciel**
 - Pour tester les différents concepts et exigences
 - Pour montrer aux clients les fonctions que l'on veut mettre en œuvre.
- Lorsque le client a donné son accord, le développement suit souvent un cycle de vie linéaire.
- Avantages : Les efforts consacrés au développement d'un prototype sont le plus souvent compensés par ceux gagnés à ne pas développer de fonctions inutiles

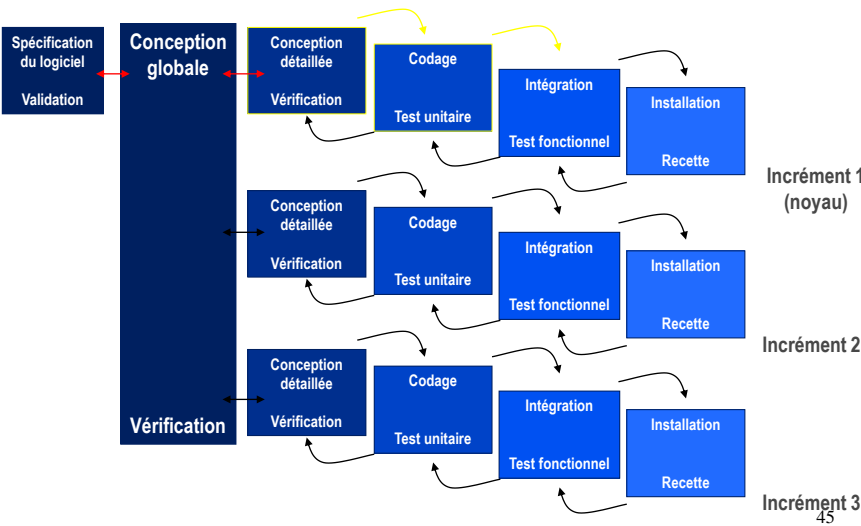
43

Modèle incrémental de Parnas

- **Il est adapté à un développement et à une mise en production par lots.**
- Les phases s'achèvent par une activité de contrôle ou de test

44

Modèle incrémental de Parnas



LES MÉTHODES « ITÉRATIVES »

Principes

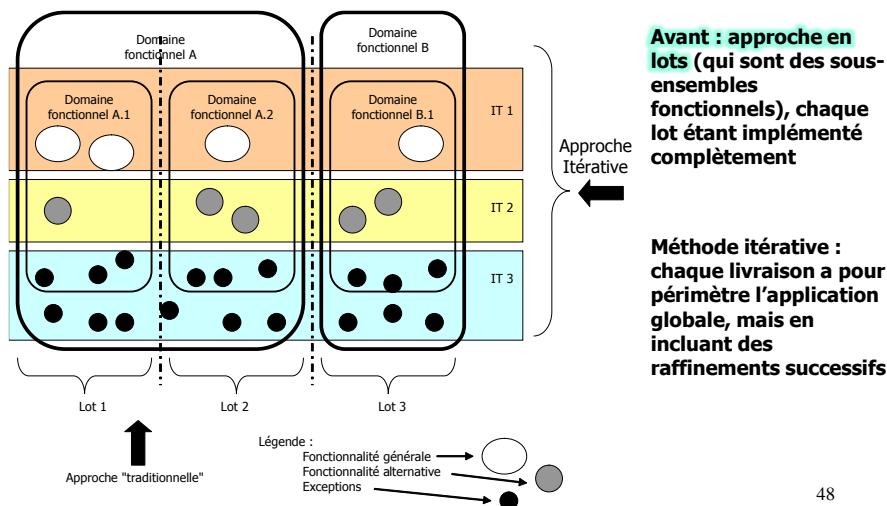
- L'idée est simple : pour modéliser (comprendre et représenter) un système complexe, il vaut mieux s'y prendre en plusieurs fois, en affinant son analyse par étapes.

→ Notion « itérative »

- Modèle souple, proche des réalités de terrain, non figé comme les modèles précédents évitant les cloisonnements stricts entre analyse et réalisation par exemple (dans les modèles précédents, on considère qu'il faut terminer 100% des spécifications avant de démarrer le développement => faux en pratique)
- Lié à une démarche de conception UML (scénario nominal, scénarii alternatifs, scénarii d'erreurs)

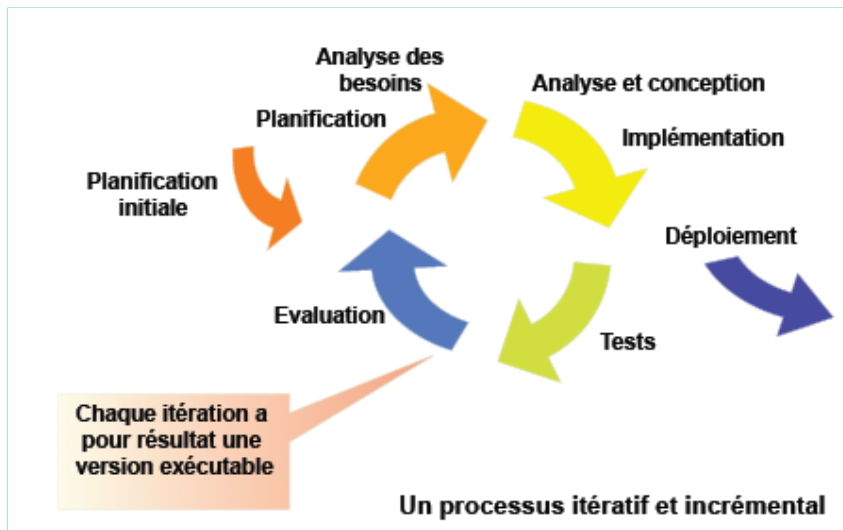
Comparaison avec les méthodes «classiques»

Ce qu'il y a de nouveau avec les méthodes itératives :



48

Développer le logiciel de façon itérative



Gérer les exigences


Problématiques

- Modifications des exigences au cours du développement
- Identification et intégration de l'ensemble des besoins au cours du processus

Propositions

- Organiser et décrire les fonctionnalités et les contraintes du système (*complétude et cohérence*)
- Evaluer les changements à apporter au cahier des charges et estimer leur impact (*modifiabilité*)
- Décrire les différentes décisions prises et en faire le suivi (*traçabilité*)

Utiliser des architectures à base de composants




L'architecture repose sur des décisions concernant :

- ☐ L'organisation du système
- ☐ Les choix des éléments structurels et de leurs interfaces
- ☐ Leur comportement
- ☐ Leur composition en sous systèmes

(Exemples : Composants Com, Corba, EJB, classes .Net, etc.)

51

Modéliser graphiquement le logiciel



- Utilisation des diagrammes UML, et plus globalement de la notion de *modèle*.
- ☐ « Tout » n'est pas modèle
- ☐ « Tout modèle » n'est pas UML

52

Vérifier la qualité du logiciel

Fonctionnalités, fiabilité et performances

- ☐ Tests (de « benchmark », de configuration, de fonctionnement, d'installation, d'intégrité, de charge, de performance, de stress, ...)
- ☐ Rapports de conception (ou revues)

53

Un cycle itératif

Les quatre phases :

- **Phase d'initialisation (Inception)** : mise en place et finalisation du SDP (Software Développment Plan, ou Plan de Développement Logiciel) en définissant les itérations nécessaires. Début de la modélisation métier et de la réalisation de prototypes ;
- **Phase d'élaboration (Elaboration)** : fixe le maximum de règles fonctionnelles (uses cases) et réaliser des prototypes permettant de lever les risques et les maquettes permettant de modéliser l'IHM générale de l'application ainsi que la cinématique d'accès,

54

Un cycle itératif

Les quatre phases :

- **phase de construction (Construction)** : développement et tests (unitaires, intégration, pré-qualification au gré de l'avancement des itérations de cette phase),
- **phase de transition (Transition)** : préparation du déploiement et conduite du changement.

55

Le modèle en spirale

- Créé par Boehm en 1988
- Il apporte une vue évolutive au modèle en cascade au travers d'une approche fondée sur l'analyse et la gestion des risques, puis la conception et la réalisation de prototypes évolutifs en fonction de l'avancement du projet.
- Avec une relation contractuelle entre le client et le fournisseur.
- Les engagements et validation présentent un caractère formalisé : chaque cycle donne lieu à une contractualisation préalable.

56

Le modèle en spirale

- Le développement présente quatre grandes phases qui se succèdent au fur et à mesure de la construction de la spirale,
 - Identification.
 - Évaluation.
 - Vérification.
 - Réalisation
- Au dernier tour de la spirale, le produit est totalement défini, les risques résolus et le produit développé

57

Le modèle en spirale

Phase Identification :

- Identification des besoins.
- Détermination des objectifs.
- Détermination des alternatives pour atteindre les objectifs.
- Détermination des contraintes

58

Le modèle en spirale

Phase évaluation:

- Analyse des risques.
- Evaluation des alternatives.
- Identification et résolution des risques

59

Le modèle en spirale

Phase de Réalisation : Développement et vérification de la solution retenue à l'issue de la phase précédente, la phase d'évaluation

60

Le modèle en spirale

Phase vérification :

- Vérification et validation du produit élaboré dans la phase de réalisation.
- Planification de la prochaine phase

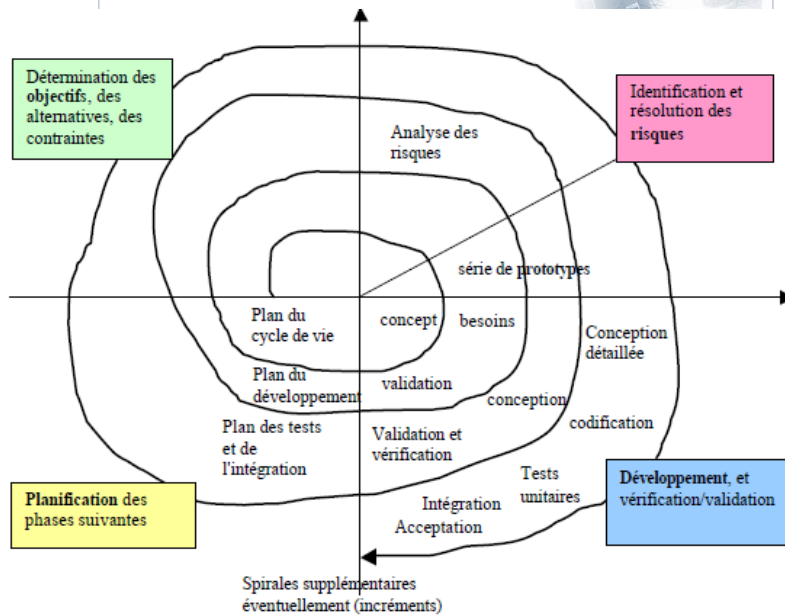
61

Le modèle en spirale

- Un cycle est composé des étapes suivantes :
 - Analyse du risque
 - Développement d'un prototype
 - Simulation et essais du prototype
 - Détermination des besoins (à des mailles différentes selon le cycle), à partir des résultats des essais.
 - Validation des besoins par un comité de pilotage
 - Planification du cycle suivant.
- Le dernier cycle permet de développer la version finale et d'implémenter le logiciel.

62

Le modèle en spirale



Synthèse : modèle en spirale

- Modèle itératif, donc chaque cycle produit une version opérationnelle du logiciel.
- Combine les meilleurs aspects du modèle classique et du modèle par prototypage.
- Introduit la notion d'analyse de risques

Synthèse : modèle en spirale

À chaque nouvelle itération :

- Plus de personnes sont impliquées.
- Le produit est plus complet.
- Le niveau de complexité augmente.
- Une nouvelle itération nécessite plus de temps.
- La décision d'arrêter ou de continuer le développement du produit est prise avant la phase d'évaluation des risques.

65

Modèle en spirale

AVANTAGES

- Modèle réaliste et naturel
- Conserve le caractère « étapiste » du modèle en cascade mais l'intègre dans une approche itérative
- Le risque est un facteur qui est tenu en compte explicitement dans ce modèle.

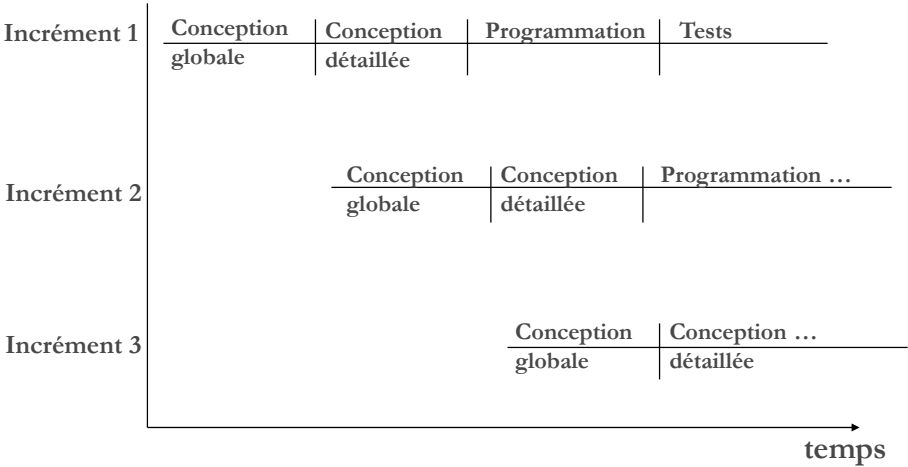
INCONVENIENTS

- Il est difficile de faire comprendre au client le mode d'opération de ce modèle
- L'évaluation des risques exige une expertise spécifique

Modèle Incrémental: Principe

- Développer des applications en étendant PROGRESSIVEMENT ses fonctionnalités.
- Les spécifications du logiciel sont figées et connues, l'étape de conception globale est terminée
- La stratégie consiste à développer le logiciel par extension successives à partir d'un produit « Noyau » du logiciel.
- Permet d'éviter de TOUT CONCEVOIR, de TOUT CODER et de TOUT TESTER comme l'approche en cascade
- Cette technique a des répercussions sur la répartition des efforts en fonction du temps, puisqu'il existe une possibilité de recouvrement des étapes.

Modèle Incrémental: Principe



Modèle Incrémental

- Portions toujours croissantes du logiciel
- Permet la correction
 - Erreurs de codage
 - Erreurs de conception
 - Erreurs de spécifications

Modèle Incrémental: Avantages et inconvénients

AVANTAGES

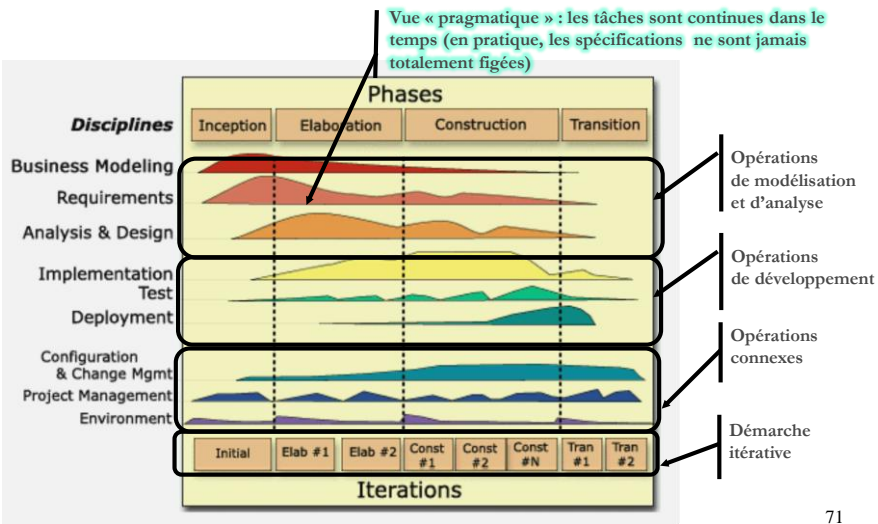
- Chaque développement est moins complexe
- Les intégrations sont progressives
- Il peut y avoir des livraisons et des mises en service après chaque intégration d'incrément
- Permet d'optimiser le temps et le partage de tâche (contrairement aux autres modèles)
- Diminution d'effort pendant la phase de tests d'intégration

INCONVENIENTS

- Le risque majeur de ce modèle est de voir remettre en cause le noyau ou les incréments précédents (définition globale des incréments et de leurs interactions dès le début du projet).
- Les incréments doivent être indépendants aussi bien fonctionnellement qu'au niveau des calendriers de développement.

Méthodes « itératives »

IMPLÉMENTATION RUP

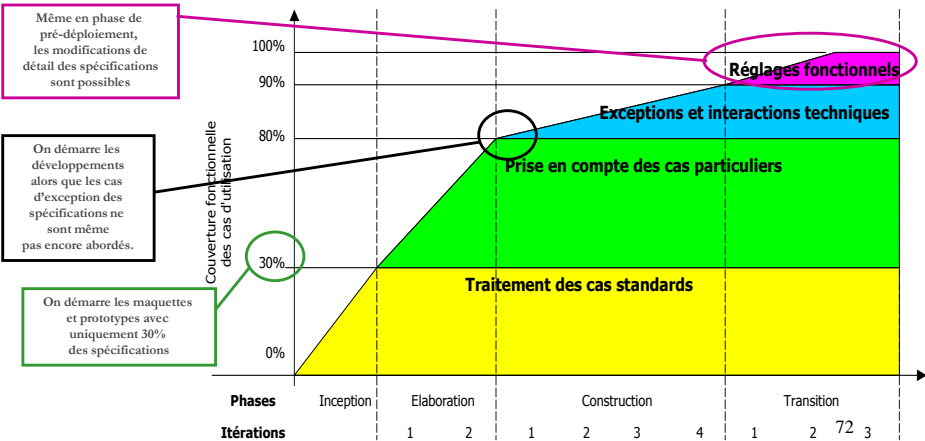


71

Méthodes « itératives »

ÉVOLUTION DE LA COUVERTURE APPLICATIVE

Illustration de la souplesse de la méthode : exemple type d'évolution de la couverture des spécifications sur le projet

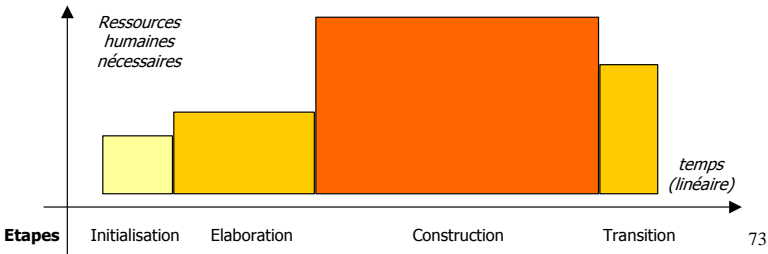


Méthodes « itératives »

RUP, RÉPARTITION DES CHARGES

La répartition des charges selon les phases du RUP :

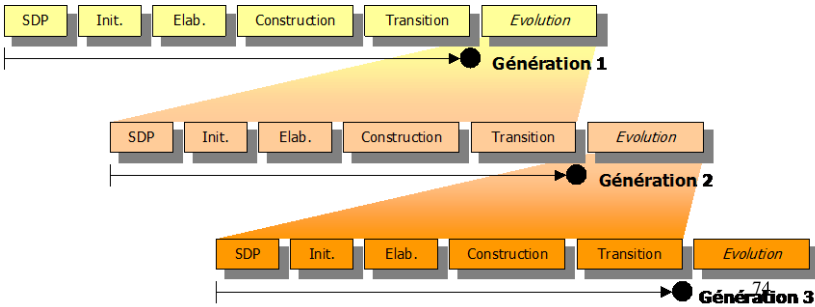
Etape	Initialisation	Elaboration	Construction	Transition
Charge de travail	~5 %	20 %	65 %	10%
Proportion de temps dans le projet	10 %	30 %	50 %	10%

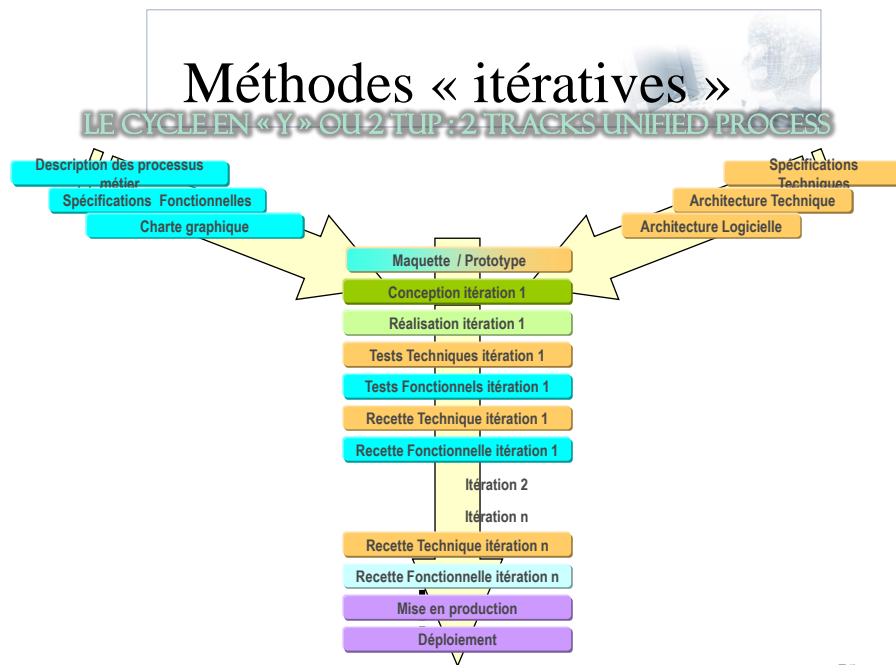


Méthodes « itératives »

PRISE EN COMPTE D'ÉVOLUTIONS DANS RUP

- Chaque évolution majeure du logiciel est traitée comme un projet à part entière, et consiste donc à redémarrer un cycle.
- A un instant donné, les différentes évolutions du logiciel peuvent être dans des phases différentes





75

Méthodes « itératives »

COMMENT CONTRACTUALISER ?

(en général)

- Le client envoie son cahier des charges plus ou moins flou
- Le prestataire doit effectuer une réponse chiffrée l'engageant sur le respect de besoin et de son estimation de charge
- Problème ! Cahier des charges flou => Nombreuses discussions contractuelles (= pertes de temps pour le projet) => Nombreux avenants => Non maîtrise du budget du projet
- Une fois le client satisfait de la qualité, il paie son prestataire
- De plus en plus les clients mettent des pénalités sur :
 - Les retards de livraison (1 jour de retard = 1 000 euros)
 - Sur la non qualité (1 anomalie bloquante = 2 000 euros)

76

Les méthodes « itératives » : Synthèse

• Intérêts :

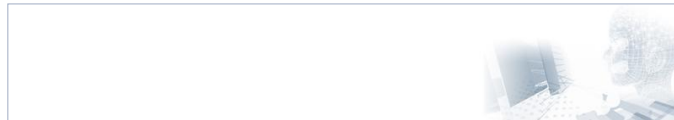
- Permettent de connaître rapidement les principales fonctionnalités d'une application → Rassurant
- Autorise de revoir une partie du besoin en cours de route
- Fournies une documentation avancée du logiciel
- Simple à appréhender et à contractualiser → Rassurant pour beaucoup de décideurs

• Difficultés à postériori :

- Méthode encore trop orientée sur une définition figée du besoin utilisateur
- Ne traite pas les problématiques de qualité de code, de non régression

• Ces méthodes s'approchent d'un processus de création incrémental

77



LES MÉTHODES « AGILE »

78

Les méthodes Agiles : Le manifeste

Extrait du manifeste Agile : <http://agilemanifesto.org/>

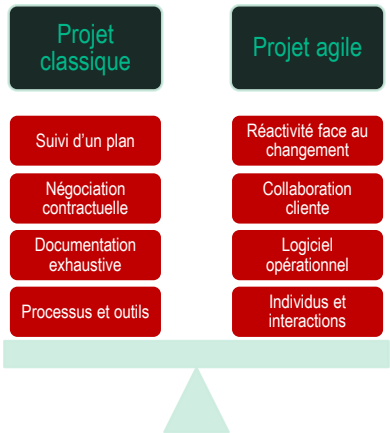
Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire. Ces expériences nous ont amenés à valoriser :

- Les individus et leurs interactions plus que les processus et les outils
- Des logiciels opérationnels plus qu'une documentation exhaustive
- La collaboration avec les clients plus que la négociation contractuelle
- L'adaptation au changement plus que le suivi d'un plan

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers.

79

Les méthodes Agiles : Le manifeste



80

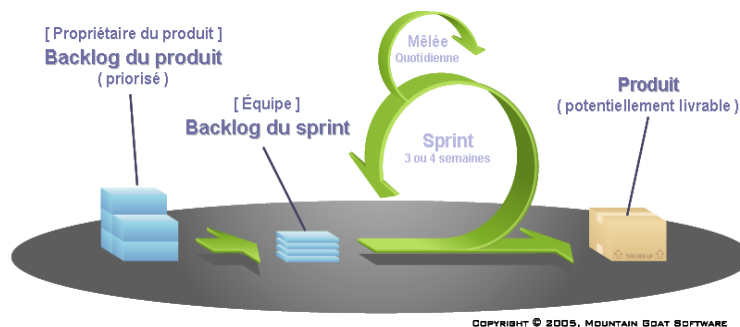
Les méthodes Agiles : plusieurs solutions

- Scrum
- eXtreme Programming
- Lean
- Devops
- Kanban
- Crystal clear
- ...

81

Agile : Focus sur SCRUM

La méthode Scrum est un processus itératif représenté par le diagramme suivant :



82

Agile : Focus sur SCRUM

Le « Backlog de produit » est constitué d’une liste de « User story », à savoir des cas d’utilisation simples et représentatifs. Il est modifié tout au long du projet par le directeur de produit. Chaque « User story » est orienté « fonctionnel » et est identifié de façon unique.

BACKLOG DE PRODUIT																				
Id	Relea.	Groupe	Nom	Priorité	Etat	Estim. (en pts)	Sprint	Ref. Vigilest	Notes	Estimation du reste à faire (en poir)										
										Date :	05/02	15/02	05/03							
<div><div></div><div></div></div>	<div><div></div><div></div></div>				<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>			Total :	85	77	81							
1	1.00	Ecran planning	Assemblage des panneaux	50	Termine	3	1	fichier.xls cas 52	cf documentation \$5.2.1		3	0								
2	1.00	Ecran planning	Affichage des cellules	49	Termine	1	1	fichier.xls cas 53			1	0								
3	1.00	Ecran planning	ScrollBar Verticale	48	Termine	1	1	fichier.xls cas 54			1	0								
4	1.00	Ecran planning	ScrollBar Horizontale	47	Termine	1	1	fichier.xls cas 55			1	0								
5	1.00	Ecran planning	Sliders	46	Termine	1	1	fichier.xls cas 56			1	0								

83

Agile : Focus sur SCRUM

Les acteurs

- Product Owner
- Scrum Master
- L’équipe

84

Agile : Focus sur SCRUM

Les acteurs – Product Owner

- Porteur de la vision globale du produit
- Gère le Backlog du Produit
- Définit les fonctionnalités du produit
- Choisit la date et le contenu de la release
- Responsable du retour sur investissement
- Définit les priorités dans le backlog en fonction de la valeur «métier»
- Ajuste les fonctionnalités et les priorités à chaque sprint si nécessaire
- Accepte ou rejette les résultat / livrables

85

Agile : Focus sur SCRUM

Les acteurs – Scrum Master

- Veille au bon fonctionnement de l'équipe (Enlève les obstacles)
- Gardien des pratiques de Scrum : responsable de faire appliquer par l'équipe les valeurs et les pratiques de Scrum
- Serviteur de l'équipe : Résout des problèmes
- N'est pas un chef de projet !
- Représente le management du projet
- S'assure que l'équipe est complètement fonctionnelle et productive
- Facilite une coopération poussée entre tous les rôles et fonctions
- Protège l'équipe des interférences extérieures

86

Agile : Focus sur SCRUM

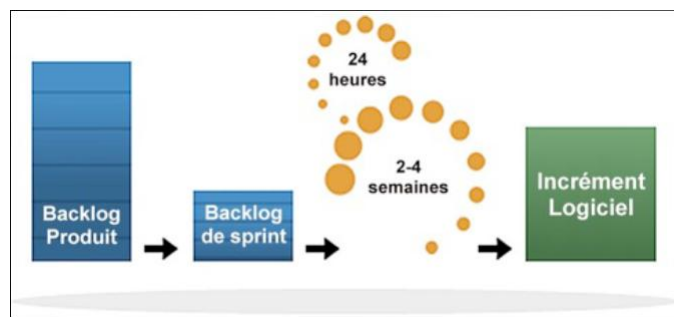
Les acteurs – L'équipe

- 5 à 9 personnes
- Regroupant tous les rôles et toutes les compétences nécessaires pour terminer le sprint : Architecte, concepteur, développeur, spécialiste IHM, testeur, etc.
- À plein temps sur le projet de préférence.
- Autogérée ; les décisions sont prises collectivement
- Ne change pas pendant un Sprint

87

Agile : Focus sur SCRUM

ORGANISATION



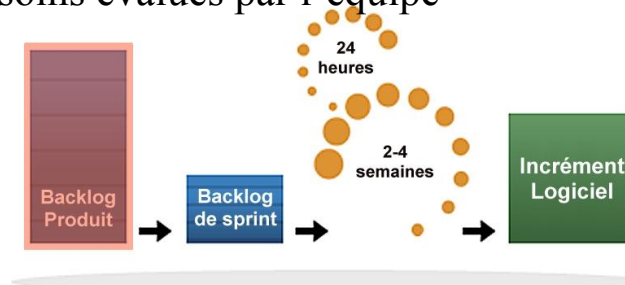
88

Agile : Focus sur SCRUM

ORGANISATION

Backlog produit (ou catalogue des besoins)

- Besoins priorisés par le product owner
- Besoins évalués par l'équipe



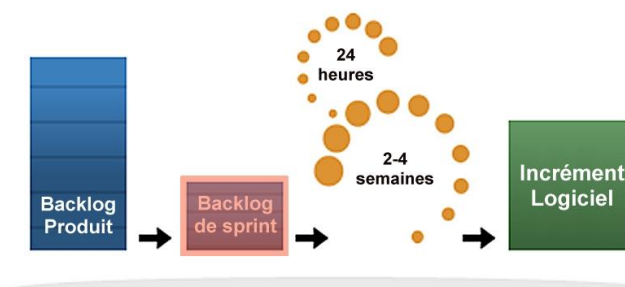
89

Agile : Focus sur SCRUM

ORGANISATION

Backlog de sprint

- Extrait du backlog produit
- Besoins éclatés en tâches



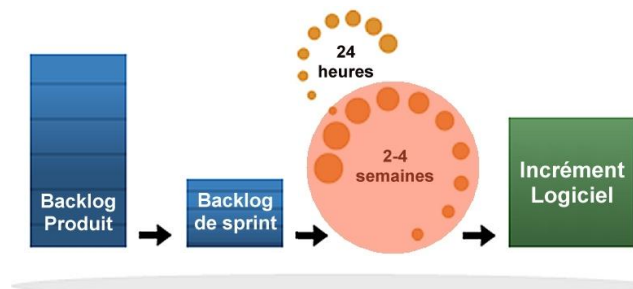
90

Agile : Focus sur SCRUM

ORGANISATION

Sprint

- Développement des fonctionnalités du backlog de sprint
- Aucune modification du backlog de sprint possible



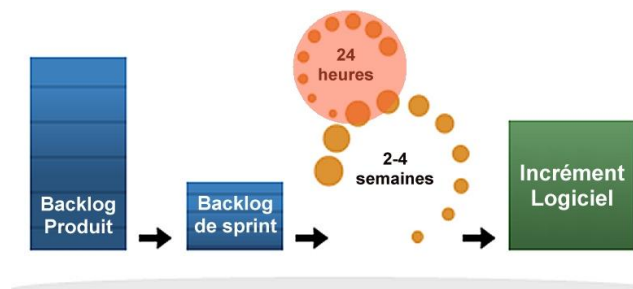
91

Agile : Focus sur SCRUM

ORGANISATION

Mêlée quotidienne

- Point de contrôle quotidien de l'équipe
- Interventions régulées – 2 min. par personne



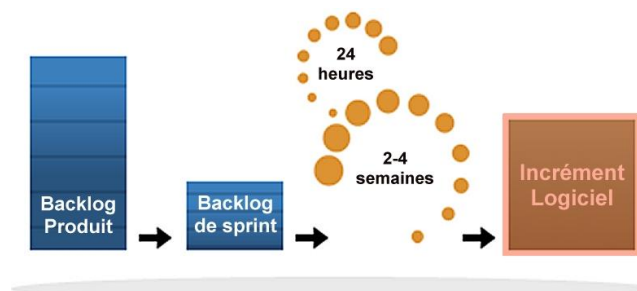
92

Agile : Focus sur SCRUM

ORGANISATION

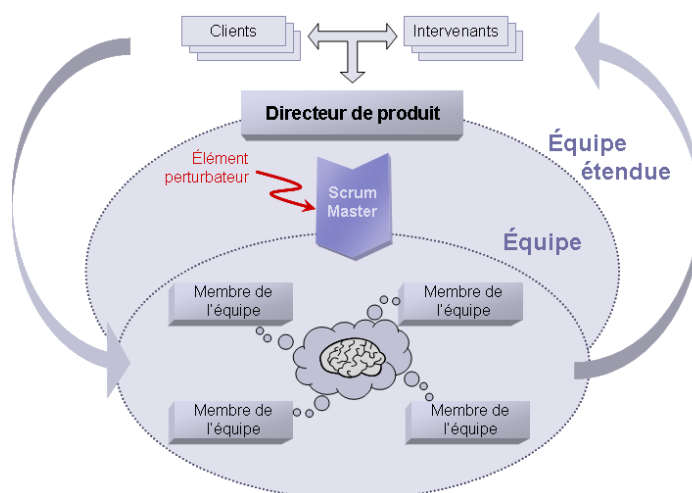
Incrément logiciel : livré au product owner à la fin du sprint

- Point de contrôle quotidien de l'équipe
- Interventions réglées – 2 min. par personne



93

Agile : Focus sur SCRUM

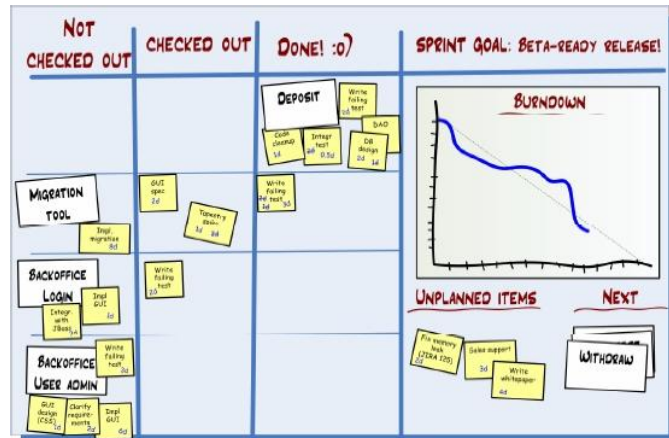


94

Agile : Focus sur SCRUM

LES INDICATEURS DE PROJET

Le tableau de tâches

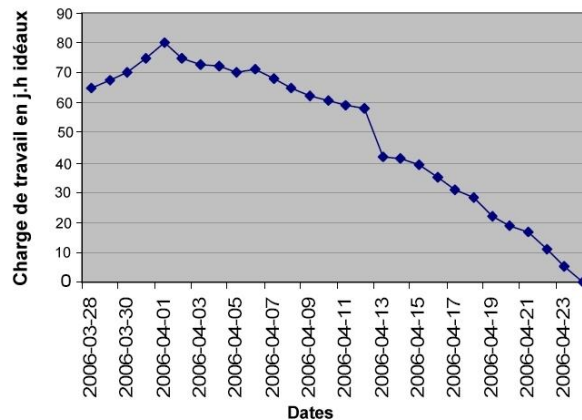


95

Agile : Focus sur SCRUM

LES INDICATEURS DE PROJET

Le burndown chart



96

Les méthodes « Agiles » : Synthèse

• Intérêts :

- Favorise clairement la productivité
- Permet aux utilisateurs de mieux appréhender leurs besoins

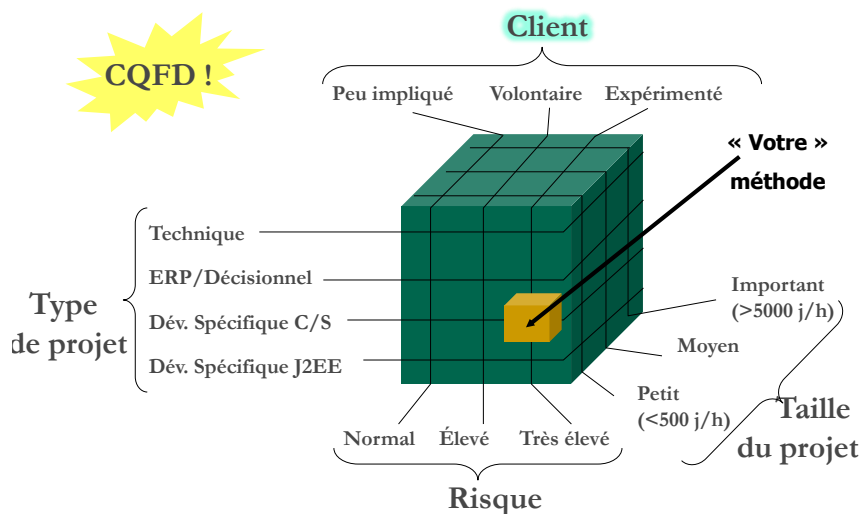
• Difficultés à posteriori :

- Nécessite une **forte implication** du client
- Conceptuellement plus complexe à appréhender (pour le client et pour les prestataires)
- Nécessite des collaborateurs **très compétents**, capables d'être performants aussi bien sur le **fonctionnel** que sur le **technique**
- Nécessité absolue de mettre en œuvre des outils de **contrôle** de qualité et de non régression
- Méthodes (très) difficiles à mettre en œuvre sur des gros projets (30, 40, ... ETP)
- La contractualisation est rendue plus difficile

97

Quelle méthode pour quel cas ?

Comment savoir quelle méthode appliquer ?



98