Chapitre 2 : Cycles de vie logiciel et méthodologies de développement

Plan Chapitre 2

- Modèles de cycles de vie
- Méthodologies de développement :
  - Méthodologie lourde
  - Méthodologie agile
- Exemple de méthode lourde : le processus unifié
- Exemple de méthode agile : SCRUM

Objectifs Chapitre 2

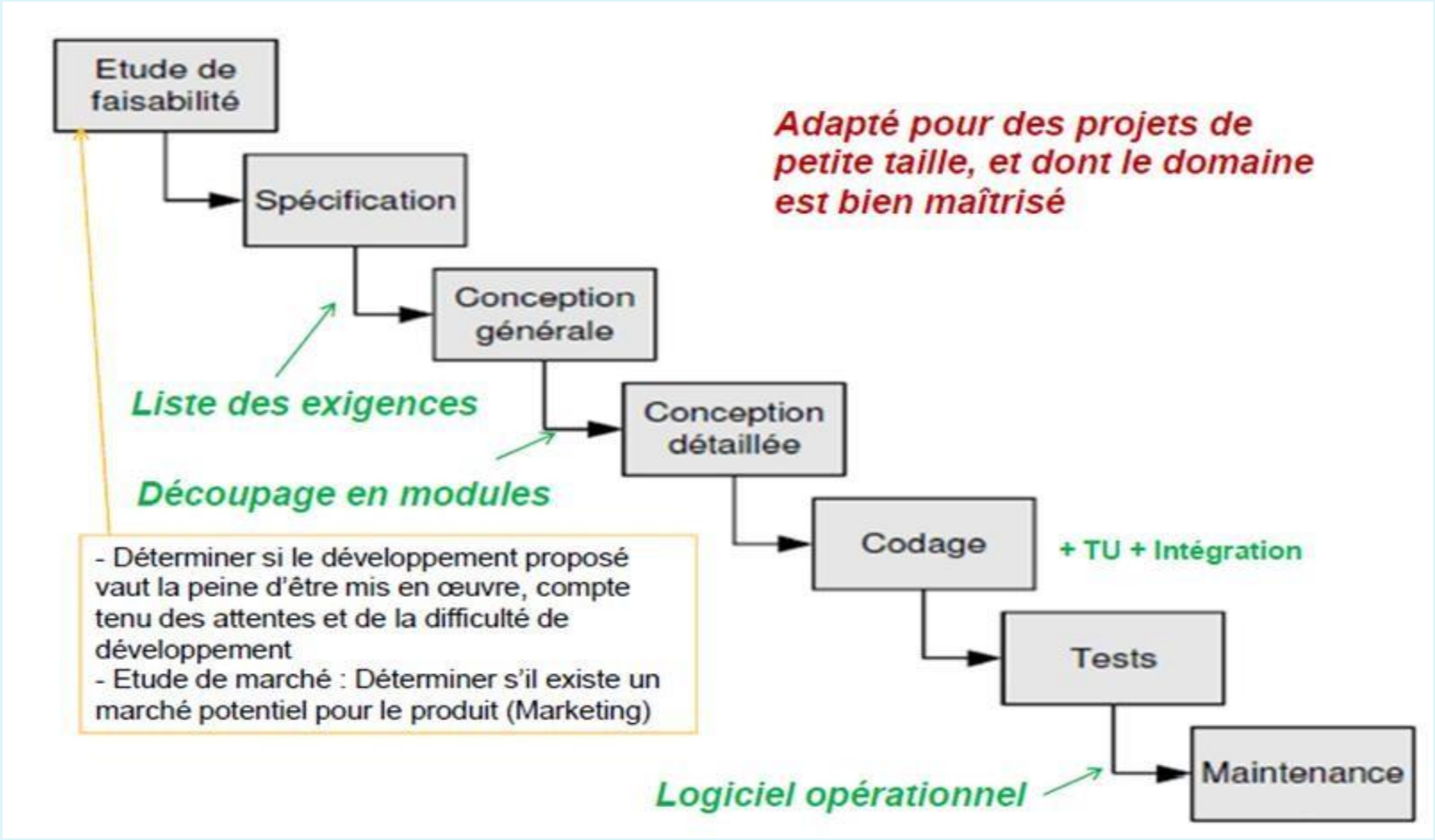
- Définition des étapes d'un cycle de vie logiciel.
- Distinction entre les méthodologies lourdes et les méthodologies agiles.
- Exploration de méthodes lourdes et de méthodes agiles.
- Comparaison entre les méthodes.

Modèles de cycle de vie

- Modèle linéaire en cascade
- Modèle linéaire en « V »
- Modèle par protoypage
- Modèle en spirale
- Modèle incrémental

Modèles de cycle de vie

Modèle linéaire en cascade

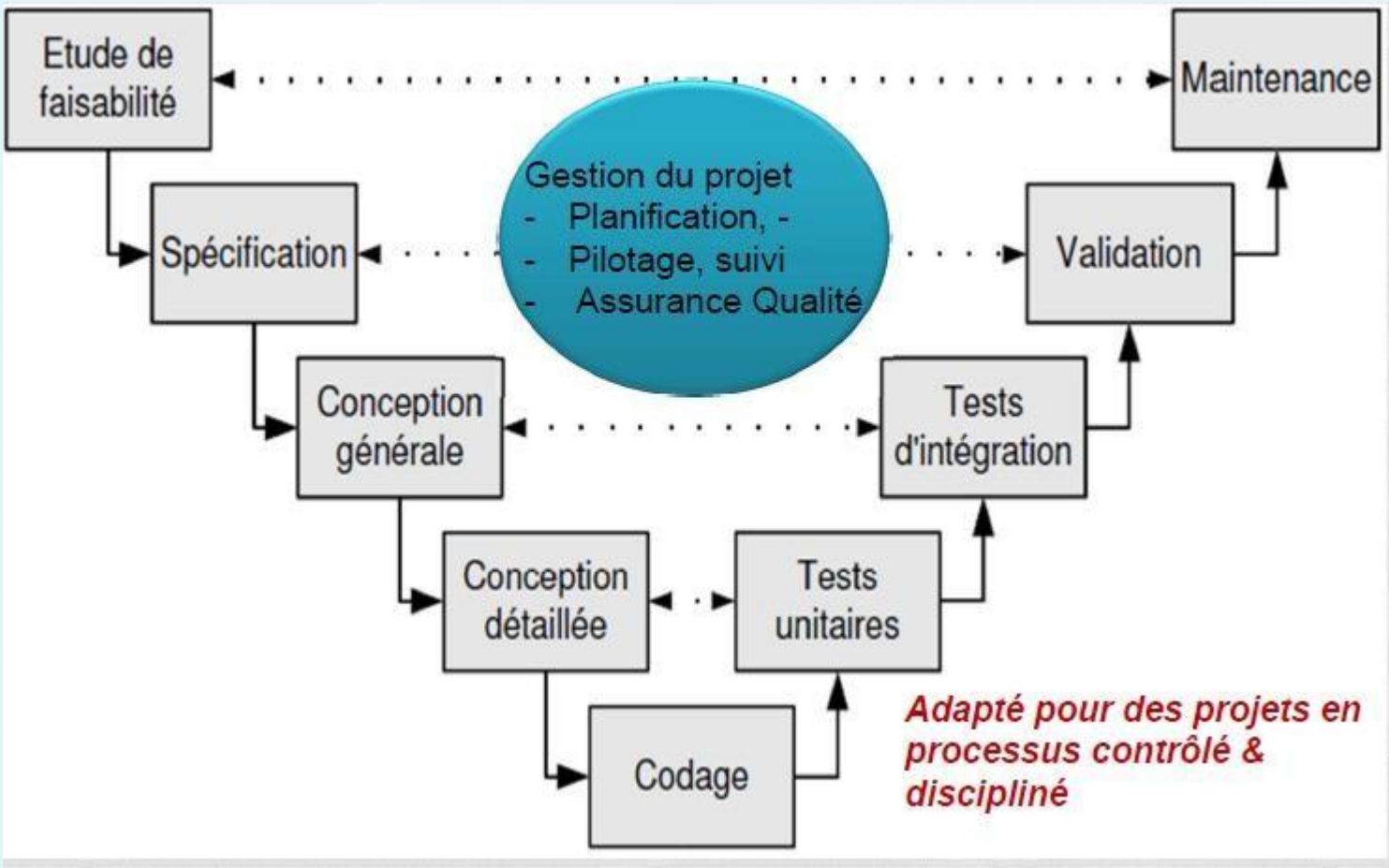


Modèles de cycle de vie

- Principales caractéristiques du modèle en cascade :
  - Chaque étape du cycle est caractérisé par des ACTIVITES dont le but est d'élaborer un ou des produits intermédiaires
  - Chaque fin d'étape est matérialisé par un évènement, où s'exerce une activité de contrôle (VERIFICATION et VALIDATION) afin d'éliminer au plus tôt 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
- Avantages :
  - Facile à mener,
  - Bien utilisé dans les petites industries
- Inconvénients :
  - Approche purement séquentielle et « simpliste », pas du tout adapté à un grand projet et complexe
  - Il est rare que le client puisse fournir toutes les spécifications dès le début du projet
  - Le client ne reçoit pas les résultats concret pendant le développement du logiciel

Modèles de cycle de vie

Modèle linéaire en « V » :



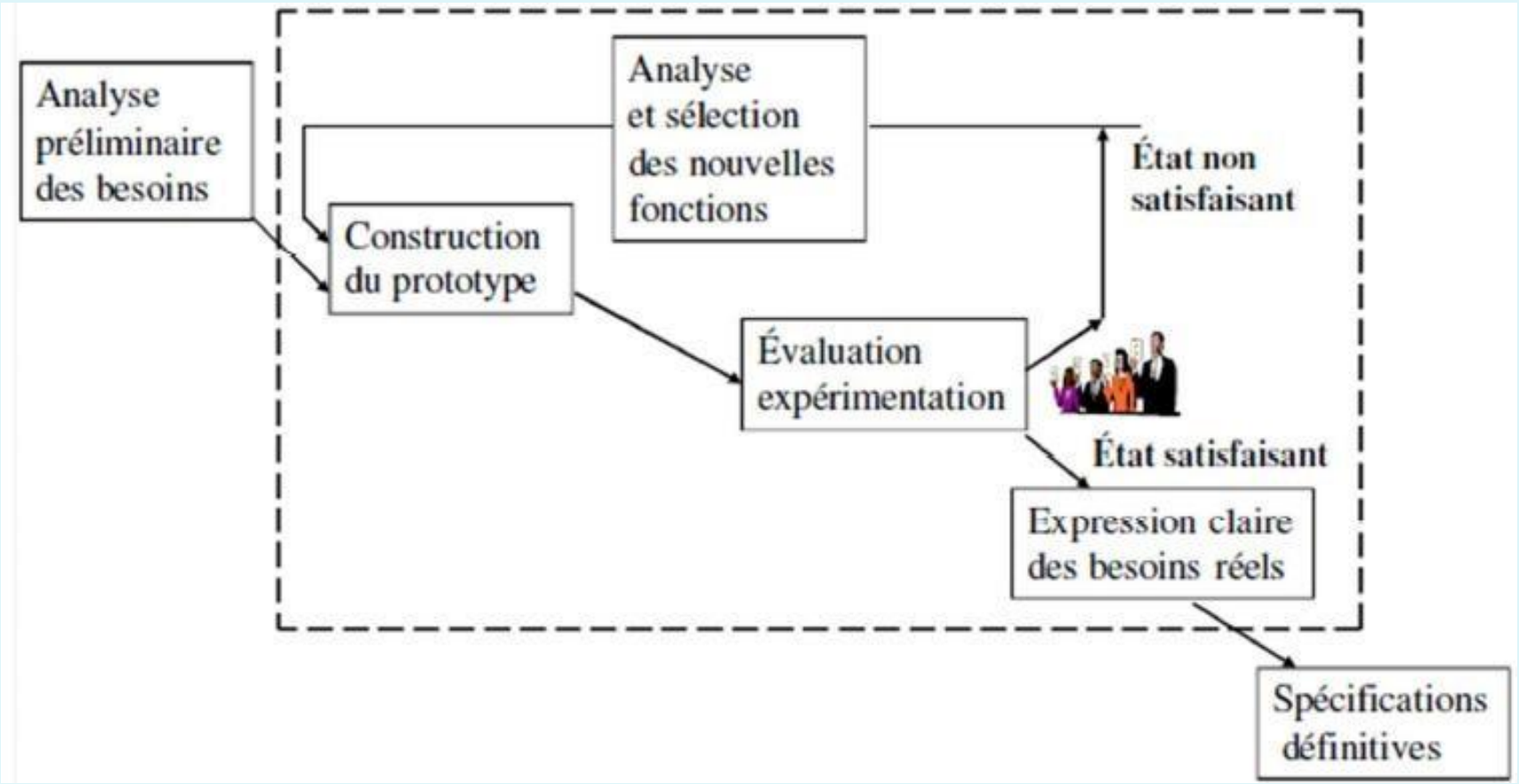
Modèles de cycle de vie

- Les première étapes du cycle doivent préparer les dernières étapes, essentiellement les activités de contrôle : vérification et de validation.
- Deux sortes de dépendances entre étapes :
  - Traits continu : correspondent à l'enchaînement et à l'itération éventuelle du modèle de la 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.
- A l'issue de la conception, le processus d'intégration et les jeux de tests d'intégration doivent être complètement décrits.
- En pratique, il est difficile voire impossible de totalement détacher la phase de conception d'un projet de sa phase de réalisation. C'est souvent au cours de l'implémentation qu'on se rend compte que les spécifications initiales étaient incomplètes, fausses, ou irréalisables, sans compter les ajouts de nouvelles fonctionnalités par les clients

C'est principalement pour cette raison que le Cycle en V n'est pas toujours adapté à un développement logiciel. La problématique des projets longues durée qui sont adaptés sur ce mode de gestion de projet est aussi souvent qu'ils risquent de ne plus "coller" aux besoins qui évoluent dans le temps.

Modèles de cycle de vie

Modèle par prototypage :



Modèles de cycle de vie

- Initialement, les spécifications données par le client sont d'ordre général et ne donne que des grandes lignes ou s'exprimant en fonction de similitude
- Raffinement des spécifications, des fonctionnalités et performances par des prototypes successifs.
- Les prototypes servent comme catalyseur pour mieux cerner les estimations et les cout de développement
- Invitation et implication du client à intervenir dans l'expression de son besoin en fonction de l'évolution du prototype
- L'analyse de chaque prototype conduit à un développement souvent rapide en but de converger rapidement
- Expérimentation de plusieurs techniques de réalisation :
  - Choix de la technologie de réalisation
  - Choix et benchmarking des outils de travail et des framework
  - Consultation des différents constructeurs autour des solutions proposés

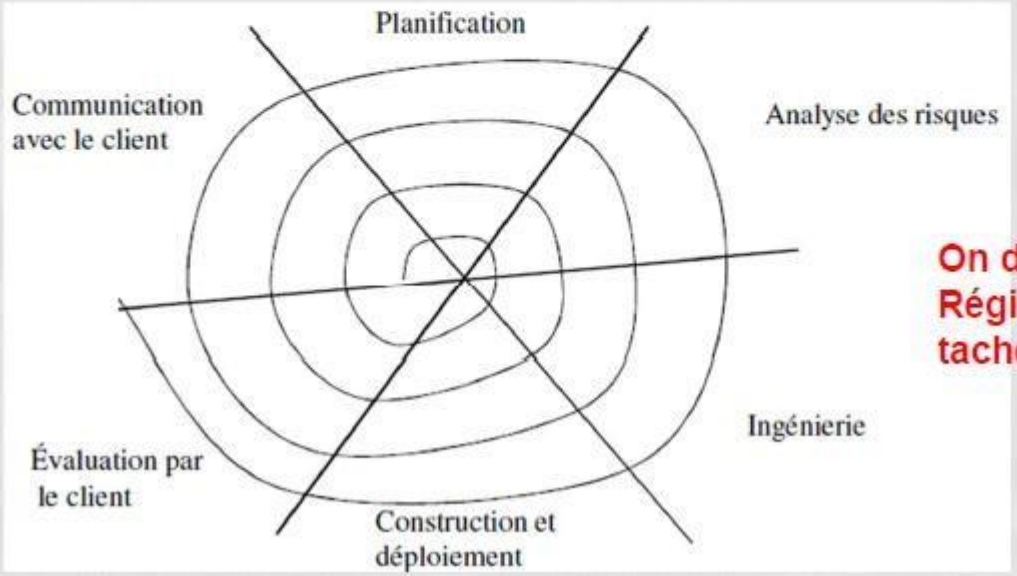
Modèles de cycle de vie

- Avantages :
  - Le client participe activement dans le développement du produit
  - Le client reçoit des résultats tangibles rapidement ; expérimentation rapide des fonctions voulue par les utilisateurs
  - Introduction d'un feedback immédiat de la part des utilisateurs
  - Amélioration de la communication entre d'une part le client et Le service commerciale et ce service et l' équipe de développement d'autre part ce qui peut garantir la satisfaction et la fidélisation du client
  - Le prototype peut servir d'apprentissage pour les futurs utilisateurs
- Inconvénients
  - Le prototype est un prétexte pour la non application des approches méthodologiques
  - Les cout et le budget de projet peut rapidement s 'éclater
  - Des aller retours avec un client gourmand et non connaisseur peuvent couter assez chère pour les deux parties

Modèles de cycle de vie

Modèle en spirale :

- Le **modèle en spirale** (*spiral model*) est un modèle de cycle de développement logiciel qui reprend les différentes étapes du cycle en V. Par l'implémentation de versions successives, le cycle recommence en proposant un produit de plus en plus complet et dur. Le cycle en spirale met cependant plus l'accent sur la gestion des risques que le cycle en V.
- Le modèle en spirale a été défini par Barry Boehm en 1988 dans son article "A Spiral Model of Software Development and Enhancement »



On distingue 6 Régions ou taches

Modèles de cycle de vie

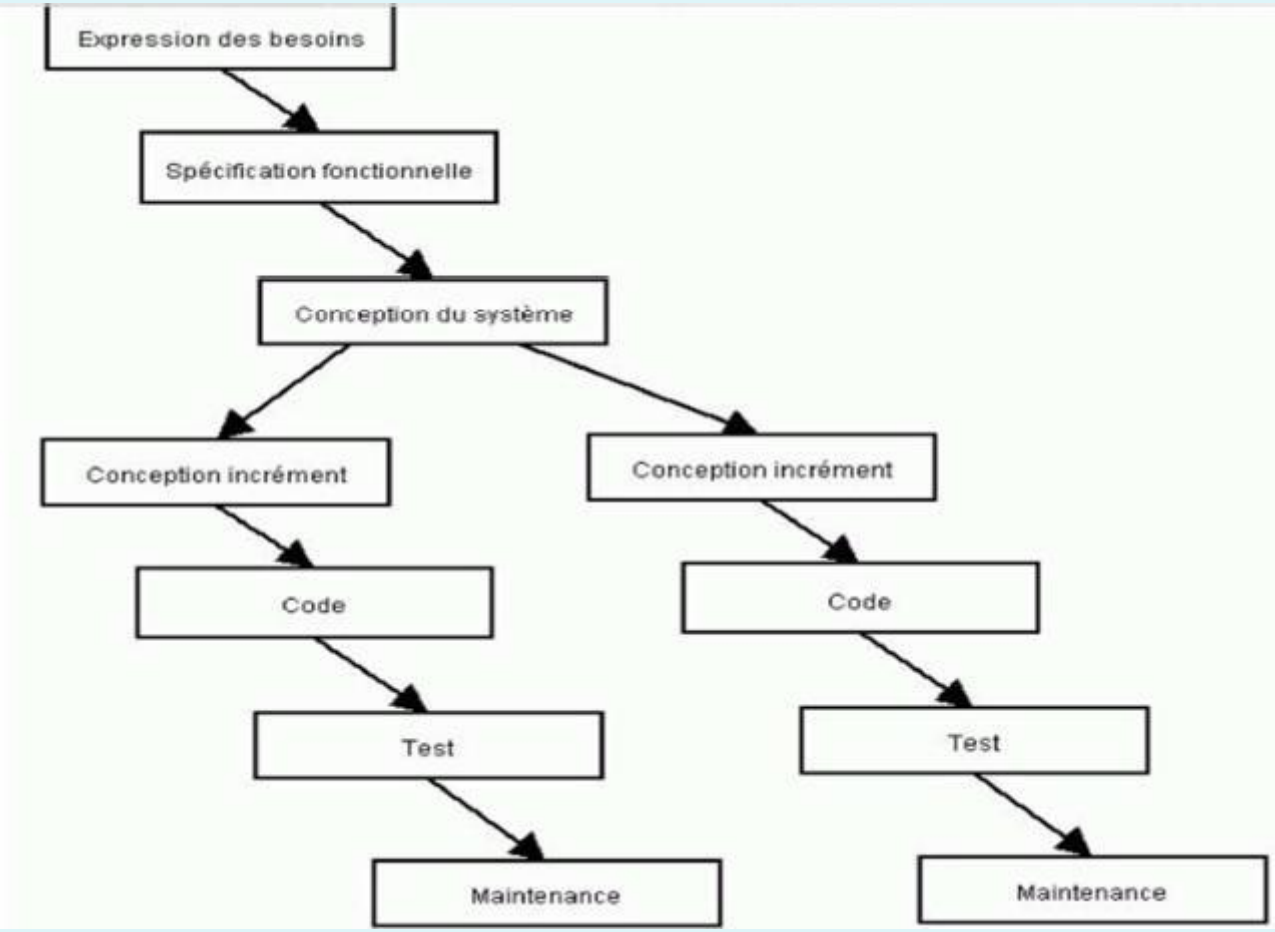
- On distingue 6 phases dans le déroulement du cycle en spirale :
  - Communication avec le client et détermination des objectifs du cycle, des alternatives pour les atteindre, les contraintes à partir du résultat précédent, analyse préliminaire des besoins…
  - Planification des activités du projet
  - Analyse des risques, évaluation des alternatives…
  - Ingénierie nécessaire à la réalisation des activités
  - Déploiement des activités
  - Vérification de la solution retenue par le client et planification du cycle suivant.
- Les activités du projet commencent par le spiral le plus profond :
  - Chaque tour passe par les régions tâches
  - L'accomplissement des phases du projet est le résultat de l'application des tâches prescrites par les régions
  - Un tour du modèle résulte en un prototype
  - On obtient un raffinement du produit en parcourant plusieurs tours du modèle

Modèles de cycle de vie

- 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 généralement pointu et repose sur une capitalisation de l'expérience vécue

Modèles de cycle de vie

Modèle incrémental :



Modèles de cycle de vie

- Développer des application 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.

Modèles de cycle de vie

- 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 précédent (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.

Méthodologies de développement

- Méthodologie lourde :
  - UP
    - RUP
    - 2TUP
- Méthodologie agile :
  - XP
  - SCRUM

Exemple de méthode lourde : Le Processus Unifié (UP)

- UP est une méthode générique de développement de logiciels.
  - Cette méthode nécessite donc d'être adaptée à chacun des projets pour lesquels elle sera employée.

Exemple de méthode lourde : Le Processus Unifié (UP)

- **Caractéristiques de UP :**
  - UP est piloté par les cas d'utilisation.
  - UP est centré sur l'architecture.
  - UP est itératif et incrémental.
  - UP est orienté risques.

Exemple de méthode lourde : Le Processus Unifié (UP)

- ***UP est piloté par les cas d'utilisation :***
  - Système analysé, conçu et développé pour des utilisateurs.
  - Tout doit donc être fait en adoptant le point de vue utilisateur.

- ***UP est centré sur l'architecture :***
  - L'architecture du système est décrite à l'aide de différentes vues.
  - L'architecte procède de manière incrémentale :
    - il commence par définir une architecture simplifiée qui répond aux besoins classés comme prioritaires
    - Puis définit à partir de l'architecture simplifiée les sous-systèmes de manière beaucoup plus précise.

Exemple de méthode lourde : Le Processus Unifié (UP)

- ***UP est itératif et incrémental :***
  - En procédant de manière itérative, il est possible de découvrir les erreurs et les incompréhensions plus tôt.
  - Le feedback de l'utilisateur est aussi encouragé et les tests effectués à chaque utilisation permettent d'avoir une vision plus objective de l'avancement du projet.
  - Le travail itératif permet à l'équipe de capitaliser à chaque cycle les enseignements du cycle précédent.

- ***UP est orienté risques :***
  - Identifier les risques.
  - Maintenir une liste de risques tout au long du projet.

Exemple de méthode lourde : Le Processus Unifié (UP)

- **Composantes de UP :**
  - **4 phases :**
    - Etude d'opportunité.
    - Elaboration.
    - Construction.
    - Transition.
  - **5 activités:**
    - Expression des besoins.
    - Analyse.
    - Conception.
    - Implémentation.
    - Test.
  - **Ensemble d'itérations :** circuit de développement aboutissant à un délivrable.

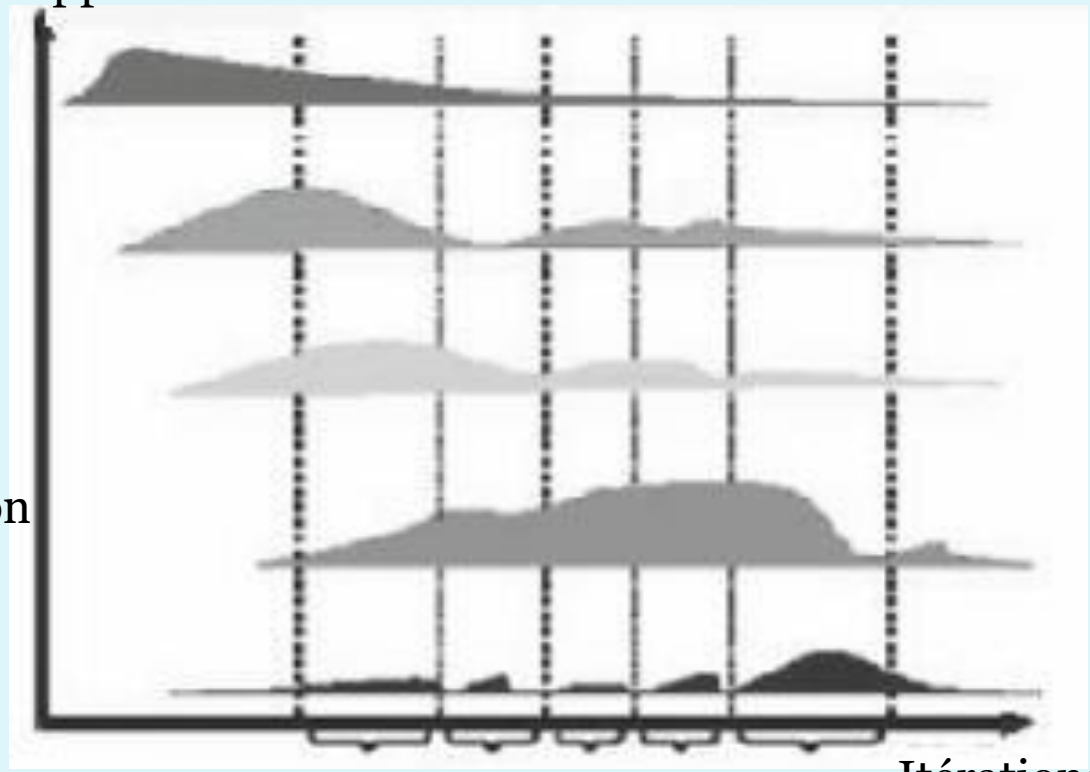
Exemple de méthode lourde : Le Processus Unifié (UP)

Organisation en fonction du temps : phases et itérations

Etude d'opportunité | Elaboration | Construction | Transition

Organisation en fonction du contenu : activités

- Expression des besoins
- Analyse
- Conception
- Implémentation
- Test



Itérations

Exemple de méthode lourde : Le Processus Unifié (UP)

- **Les activités dans UP :**
  - **Expression des besoins :**
    - Identification des besoins fonctionnels.
    - Identification des besoins non fonctionnels.
  - **Analyse :**
    - Formalisation du système à partir des besoins.
    - Modélisations de diagrammes statiques et dynamiques.
    - Vue logique du système.
  - **Conception :**
    - Définition de l'architecture du système.
    - Etendre les diagrammes d'analyse.
    - Prise en compte des contraintes de l'architecture technique.

Exemple de méthode lourde : Le Processus Unifié (UP)

- **Implémentation :**
  - Production du logiciel :
    - Composants.
    - Bibliothèques.
    - Fichiers.
    - Etc.

- **Test :**
  - Vérifier l'implémentation de tous les besoins (fonctionnels et non fonctionnels).
  - Vérifier l'interaction entre les objets.
  - Vérifier l'intégration de tous les composants.
  - Différents niveaux de tests (unitaires, d'intégration, de performance, etc.).

Exemple de méthode lourde : Le Processus Unifié (UP)

- **Les phases de UP :**
  - **Etude d'opportunité :**
    - Cette phase pose la question de la **faisabilité du projet**, des frontières du système, des **risques** majeurs qui pourraient mettre en péril le projet.
    - A la fin de cette phase, est établi un document donnant une **vision globale des principales exigences, des fonctionnalités clés et des contraintes majeures**.
      - Environ 10 % des cas d'utilisation sont connus à l'issue de cette phase.
    - Il convient aussi d'établir une estimation initiale des risques, un "Project Plan", un "Business Model".

Exemple de méthode lourde : Le Processus Unifié (UP)

- **Elaboration :**
  - Reprise des résultats de la phase d'incubation.
  - **Spécification détaillée des cas d'utilisation**.
  - Détermination de l'architecture de référence.

- **Construction :**
  - Construction d'une première version du produit (**version bêta du release** en cours de développement ainsi qu'une version du manuel utilisateur).
  - **Construction de tous les cas d'utilisation**.

Exemple de méthode lourde : Le Processus Unifié (UP)

- **Transition :**
  - **Test et correction** des anomalies.
  - **Déploiement** du produit.
  - Préparer la release suivante et boucler le cycle soit sur une nouvelle étude d'opportunité soit une élaboration ou construction.

Méthodes agiles

- **Inconvénients de UP :**
  - Fait tout, mais lourd.
  - Parfois difficile à mettre en œuvre de façon spécifique.
- UP pour les gros projets qui génèrent beaucoup de documentation.

- **Quelles activités pouvons nous abandonner tout en produisant des logiciels de qualité?**

- Comment mieux travailler avec le client pour nous focaliser sur ses besoins les plus prioritaires et être aussi réactifs que possible ?
  - ⟹ XP (eXtreme Programming) / SCRUM

Méthodes agiles

Priorités des méthodes agiles :

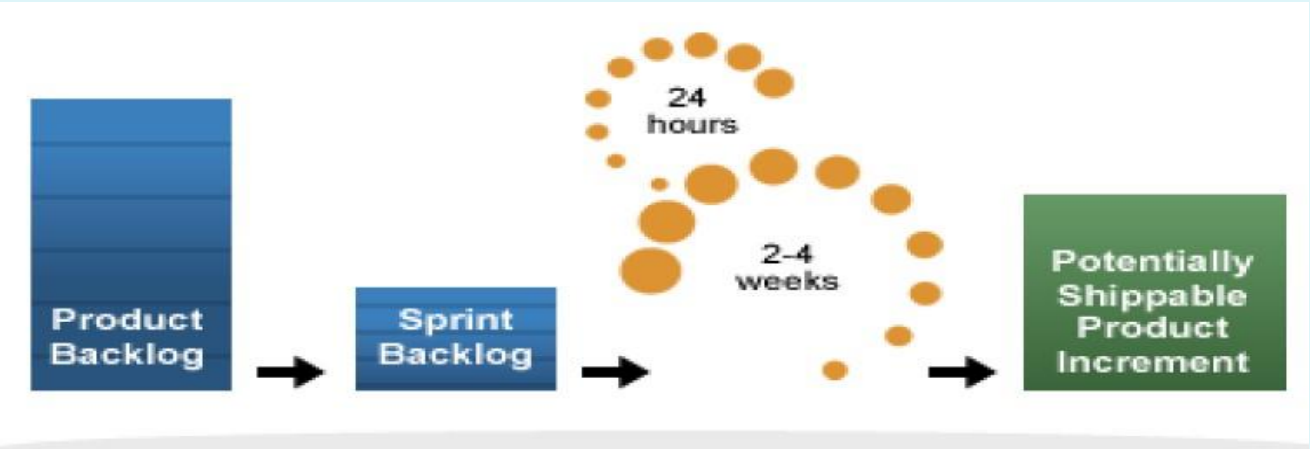
- Priorité aux **personnes** et aux **interactions** sur les *procédures et les outils*;
- Priorité aux **applications fonctionnelles** sur une *documentation pléthorique* ;
- Priorité à la **collaboration avec le client** sur la *négociation de contrat* ;
- Priorité à **l'acceptation du changement** sur la *planification*.

RUP 4 fois plus lent que Scrum : jusqu'à 27 rôles !

beaucoup plus de réunions
beaucoup plus de reportings
beaucoup plus d'efforts de communication

Exemple de méthode agile : Scrum

- Scrum est une méthode agile utilisée dans le développement de logiciels. Elle vise à **satisfaire au mieux les besoins du client** tout en **maximisant les probabilités de réussite du projet**.
- Scrum suppose que le développement d'un logiciel n'est pas **prévisible** et ne peut donc **pas** suivre de **processus défini**.
- Le résultat du projet dépend des **réorientations** que lui donne le **client** en cours de route.
- Un projet utilisant Scrum est composé d'une suite d'itérations courtes de l'ordre de 3 à 6 semaines appelées **sprints**.
- Le projet peut être réorienté par le client à la fin de chaque sprint



Exemple de méthode agile : Scrum

1. Le projet est organisé en **sprints**. C'est pendant cette phase que les **fonctionnalités** choisies sont développées et que le logiciel est créé.
2. Le **backlog du produit** constitue l'ensemble du **travail connu sur le projet à un instant t**. Il est régulièrement **remis à jour** et les besoins qui le constituent sont aussi régulièrement **priorisés**.
3. Le **travail à faire durant un sprint** est listé dans le **backlog du sprint**. Le contenu du sprint est un extrait du backlog produit. Les besoins sont priorisés de la même façon que dans le backlog du produit.
4. Durant le sprint, une réunion quotidienne appelée **daily scrum** (ou mêlée quotidienne) rassemble l'équipe afin de réorienter le sprint si besoin est. Il s'agit du point de contrôle de l'équipe.
5. A la fin d'un sprint, l'équipe livre au client un incrément de logiciel fini potentiellement livrable.
6. Ce cycle est répété jusqu'à ce que :
    1. La date de fin de projet soit atteinte.
    2. Le client ne peut plus financer le projet.
    3. Le client considère que le logiciel délivre suffisamment de valeur et décide d'arrêter le développement.

Exemple de méthode agile : Scrum

Les phases de SCRUM :

Phases

- Initiation / démarrage
  - Planning
    - définir le système : product Backlog = liste de fonctionnalités, ordonnées par ordre de priorité et d'effort
  - Architecture
    - conception de haut-niveau
- Développement
  - Cycles itératifs (sprints) : 30j
    - amélioration du prototype
- Clôture
  - Gestion de la fin du projet : livraison…

Exemple de méthode agile : Scrum

Scrum : Principes

Isolement de l'équipe de développement

- l'équipe est isolée de toute influence extérieure qui pourrait lui nuire. Seules l'information et les tâches reliées au projet lui parviennent : pas d'évolution des besoins dans chaque sprint.

Développement progressif

- afin de forcer l'équipe à progresser, elle doit livrer une solution tous les 30 jours. Durant cette période de développement l'équipe se doit de livrer une série de fonctionnalités qui devront être opérationnelles à la fin des 30 jours.

Pouvoir à l'équipe

- l'équipe reçoit les pleins pouvoirs pour réaliser les fonctionnalités. C'est elle qui détient la responsabilité de décider comment atteindre ses objectifs. Sa seule contrainte est de livrer une solution qui convienne au client dans un délai de 30 jours.

Contrôle du travail

- le travail est contrôlé quotidiennement pour savoir si tout va bien pour les membres de l'équipe et à la fin des 30 jours de développement pour savoir si la solution répond au besoin du client.

Exemple de méthode agile : Scrum

Scrum : rôles et pratiques

Scrum Master
- expert de l'application de Scrum

Product owner
- responsable officiel du projet

Scrum Team
- équipe projet.

Customer
- participe aux réunions liées aux fonctionnalités

Management
- prend les décisions

- **Product Backlog**
  - état courant des tâches à accomplir
- **Effort Estimation**
  - permanente, sur les entrées du backlog
- **Sprint**
  - itération de 30 jours
- **Sprint Planning Meeting**
  - réunion de décision des objectifs du prochain sprint et de la manière de les implémenter
- **Sprint Backlog**
  - Product Backlog limité au sprint en cours
- **Daily Scrum meeting**
  - ce qui a été fait, ce qui reste à faire, les problèmes
- **Sprint Review Meeting**
  - présentation des résultats du sprint

Exemple de méthode agile : Scrum

Scrum : Les outils

- Les outils dédiés au management de projets agiles exemple IceScrum, Xplanner, etc.
- Le tableau blanc et les post-its :
  - Il s'agit ici de gérer le backlog produit et le backlog de sprint grâce à un tableur et d'accrocher un grand tableau blanc, de le diviser en colonnes et de coller des post-its représentant les besoins et les tâches du backlog de sprint.

Exemple de méthode agile : Scrum

- **Avantages du tableau blanc et des post-its :**
  - **Outil participatif :** chacun peut venir déplacer son post-it et s'arrêter devant le tableau pour réfléchir ou juste voir ce qu'il reste à faire.
  - **Aide à la communication** dans l'équipe lors des réunions quotidiennes. C'est un support pour les questions quotidiennes dont les réponses sont étayées par les tâches affichées.
  - L'équipe se voit progresser. Les post-its passent de la gauche à la droite du tableau. C'est un **facteur de motivation**.

- **Inconvénients du tableau blanc et des post-its :**
  - **L'expérience n'est pas capitalisée** : les post-its du sprint précédent sont enlevés. Il faut donc consigner manuellement dans un document l'expérience acquise, en terme d'exactitude des évaluations par exemple.
  - **Les post-its se décollent**. Il est fortement conseillé de prendre régulièrement des photos du tableau, etc.
  - Les **courbes de progression** sont à faire à la main.
  - Le **reporting au client et à sa hiérarchie** est à faire manuellement. En cas de confiance totale, alors cela ne pose pas de souci : l'équipe s'engage au début du sprint et réalise une démonstration à la fin. Mais ce n'est souvent pas le cas.

Synthèse des méthodologies utilisées dans le cadre de développement objet et de nouvelles technologies.

	Description	Points forts	Points faibles
Cascade	-Propose de dérouler les phases projet de manière séquentielle -Cité pour des raisons historiques	Distingue clairement les phases Projet	- Non itératif -Ne propose pas de modèles de documents
RUP Rational Unified Process	-Promu par Rational. -Le RUP est à la fois une méthodologie et un outil prêt à l'emploi (documents types partagés dans un référentiel Web) -Cible des projets de plus de 10 personnes	-Itératif -Spécifie le dialogue entre les différents intervenants du projet : les livrables, les plannings, les prototypes… -Propose des modèles de documents, et des canevas pour des projets types	- Coûteux à personnaliser -Très axé processus, au détriment du développement : peu de place pour le code et la Technologie
XP eXtreme Programming	-Ensemble de « Bests Practices » de développement (travail en équipes, transfert de compétences…) - Cible des projets de moins de 10	-Itératif -Simple à mettre en œuvre -Fait une large place aux aspects techniques : prototypes, règles de développement, tests…	-Ne couvre pas les phases en amont et en aval au développement : capture des besoins, support, maintenance, tests d'intégration… -Elude la phase d'analyse, si bien qu'on peut dépenser son énergie à faire et défaire -Assez flou dans sa mise en œuvre: quels intervenants, quels livrables ?
2TUP Two Track Unified Process	-S'articule autour de l'architecture -Propose un cycle de développement en Y -Détaillé dans "UML en action" (voir références) - Cible des projets de toutes tailles	-Itératif -Fait une large place à la technologie et à la gestion du risque -Définit les profils des intervenants, les livrables, les plannings, les Prototypes	-Plutôt superficiel sur les phases situées en amont et en aval du développement : capture des besoins, support, maintenance, gestion du changement… - Ne propose pas de documents types

Source: Processus Unifié – Unified Process, Free Information Systems For Management Application , www.freewebs.com/fresma 1/4