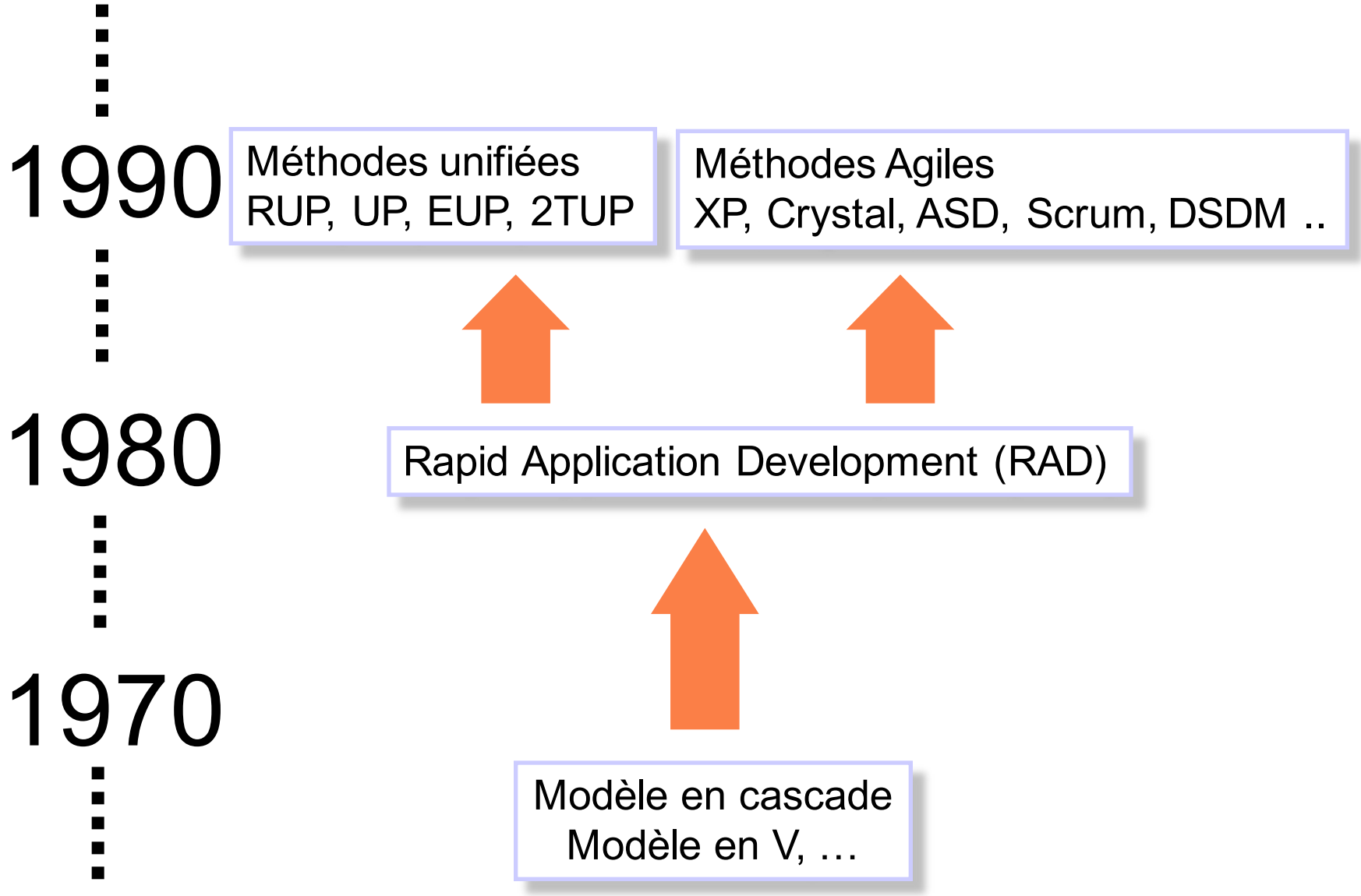
III
Démarche générale de modélisation avec UML

Les grandes approches de modélisation



Comment modéliser avec UML ?

UML est un langage qui permet de représenter des modèles, mais il ne définit pas le processus d'élaboration des modèles: UML n'est donc pas une méthode de modélisation.

➢Dans le cadre de la modélisation d'une application informatique, les auteurs d'UML préconisent d'utiliser une démarche :

- itérative et incrémentale,
- guidée par les besoins des utilisateurs du système,
- centrée sur l'architecture logicielle.

D'après les auteurs d'UML, un processus de développement qui possède ces qualités devrait favoriser la réussite d'un projet.

Démarche itératives et incrémentale

- Pour modéliser (comprendre et représenter) un système complexe, il vaut mieux s'y prendre en plusieurs fois, en affinant son analyse par étapes.

- Cette démarche doit aussi s'appliquer au cycle de développement dans son ensemble, en favorisant le prototypage.

- Le but est de mieux maîtriser la part d'inconnu et d'incertitudes qui caractérisent les systèmes complexes.

Démarche pilotée par les besoins des utilisateurs

- Avec UML, ce sont les utilisateurs qui guident la définition des modèles : Le périmètre du système à modéliser est défini par les besoins des utilisateurs (les utilisateurs définissent ce que doit être le système). Le but du système à modéliser est de répondre aux besoins de ses utilisateurs (les utilisateurs sont les clients du système).

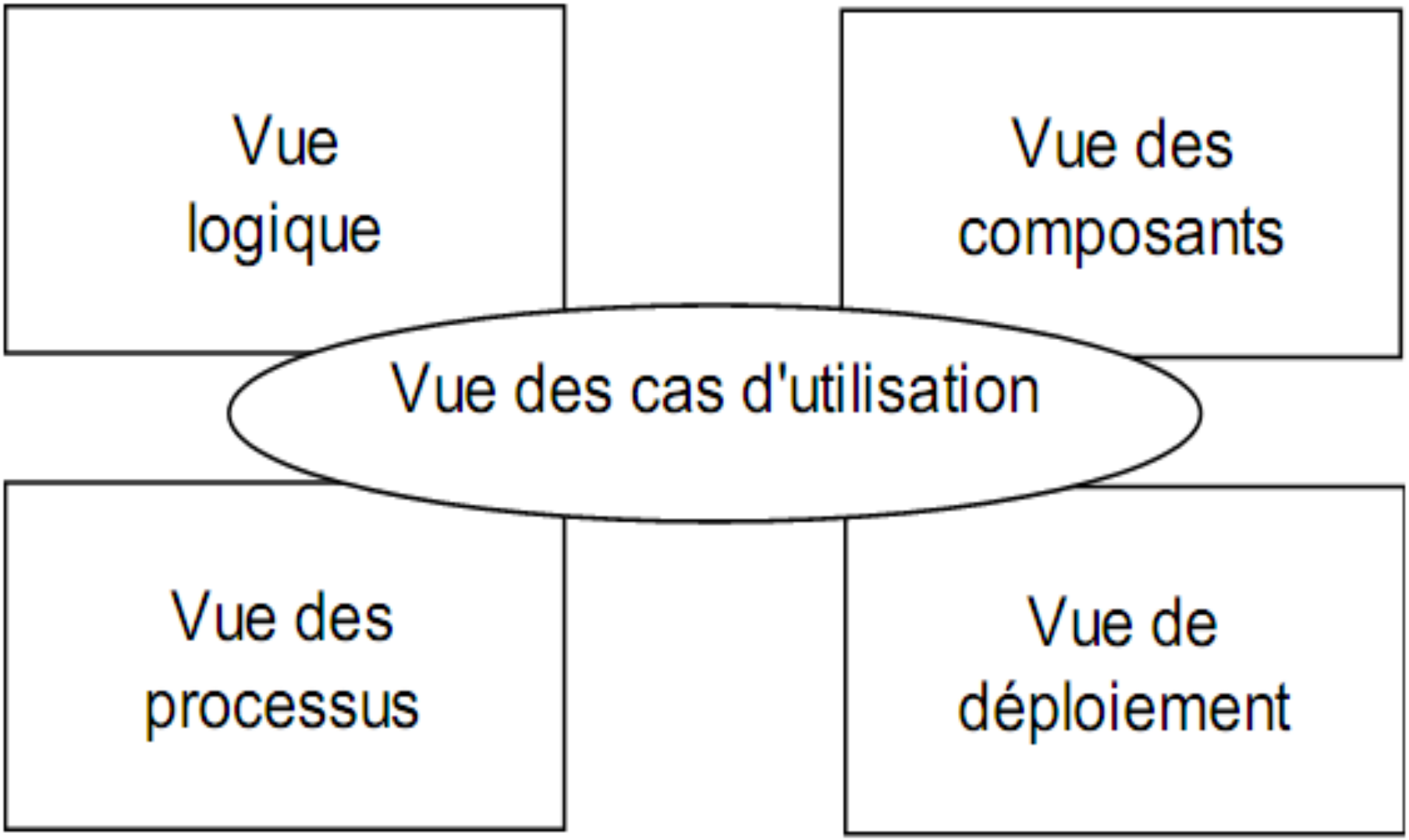
- Les besoins des utilisateurs servent aussi de fil rouge, tout au long du cycle de développement (itératif et incrémental) :
  - à chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs.
  - à chaque itération de la phase de conception et de réalisation, on veille à la prise en compte des besoins des utilisateurs.
  - à chaque itération de la phase de test, on vérifie que les besoins des utilisateurs sont satisfaits.

Démarche centrée sur l'architecture

- Une architecture adaptée est la clé de voûte du succès d'un développement. Elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...).
- L'architecture logicielle s'occupe d'abstraction, de décomposition et de composition, de style et d'esthétique. Pour décrire une architecture architecture logicielle, on utilisera un modèle composé de plusieurs vues ou perspectives.
- Le modèle de Kruchten (publication IEEE, 1995) dit « schéma 4+1 vues » ou simplement vue « 4+1 » est celui adopté dans le Processus Unifié.

Philippe B. Kruchten, The 4+1 View Model of Architecture [archive], IEEE Software, novembre 1995.

La vue « 4+1 » de ph. Kruchten



La vue des cas d'utilisation

Cette vue est particulière en ce sens qu'elle guide toutes les autres.

- Un cas d'utilisation est défini comme un ensemble de scénarios d'utilisation, chaque scénario représentant une séquence d'interaction des utilisateurs (acteurs) avec le système.
- L'intérêt des cas d'utilisation est de piloter l'analyse par les exigences des utilisateurs. Ceux-ci se sentent concernés car ils peuvent facilement comprendre les cas d'utilisation qui les concernent. Cette méthode permet donc d'aider à formaliser les véritables besoins et attentes des utilisateurs; leurs critiques et commentaires étant les briques de la spécification du système.
- L'ensemble des cas d'utilisation du logiciel en cours de spécification est représenté par un diagramme de cas d'utilisation, chacun des scénarios de celui-ci étant décrit par un ou plusieurs diagrammes dynamiques : diagrammes d'activités, de séquence, diagrammes de communication ou d'états-transitions.

La vue logique

- Cette vue concerne « l'intégrité de conception ». Cette vue de haut niveau se concentre sur l'abstraction et l'encapsulation, elle modélise les éléments et mécanismes principaux du système. Elle identifie les éléments du domaine, ainsi que les relations et interactions entre ces éléments « notions de classes et de relations » :
  - les éléments du domaine sont liés au(x) métier(s) de l'entreprise,
  - ils sont indispensables à la mission du système,
  - ils gagnent à être réutilisés (ils représentent un savoir-faire).
- La vue logique est représentée, principalement, par des diagrammes statiques de classes et d'objets enrichis de descriptions dynamiques : diagrammes d'activités, de séquence, diagrammes de communication ou d'états-transitions.

La vue des composants

- Cette vue concerne « l'intégrité de gestion ». Elle exprime la perspective physique de l'organisation du code en termes de modules, de composants et surtout des concepts du langage ou de l'environnement d'implémentation.
- Dans cette perspective, l'architecte est surtout concerné par les aspects de gestion du code, d'ordre de compilation, de réutilisation, d'intégration et d'autres contraintes de développement pur.
- Pour représenter cette perspective, UML fournit des concepts adaptés (tels que les modules, les composants, les relations de dépendance, l'interface …) qui permet ainsi de visualiser l'organisation des composants (bibliothèque dynamique et statique, code source, ...) dans l'environnement de développement.
- Cette vue permet également de gérer la configuration (auteurs, versions...).
- Les seuls diagrammes de cette vue sont les diagrammes de composants.

La vue des processus

- Cette vue concerne « l'intégrité d'exécution ». Elle décrit les interactions entre les différents processus, threads (fils d'exécution) ou tâches, elle permet également d'exprimer la synchronisation et l'allocation des objets.
- Cette vue est très importante dans les environnements multitâches. Elle vue permet avant tout de vérifier le respect des contraintes de fiabilité, d'efficacité et de performances des systèmes multitâches.
- Elle exprime la perspective sur les activités concurrentes et parallèles. Elle montre ainsi :
  - la décomposition du système en terme de processus (tâches).
  - les interactions entre les processus (leur communication).
  - la synchronisation et la communication des activités parallèles (threads).
- Les diagrammes utilisés dans la vue des processus sont exclusivement dynamiques : diagrammes d'activités, de séquence, diagrammes de communication ou d'états-transitions

La vue de déploiement

- Cette vue concerne « l'intégrité de performance ». Elle représente le système dans son environnement d'exécution.
- Elle exprime la répartition du système à travers un réseau de nœuds logiques de traitements.
- Elle montre ainsi :
  - la disposition et nature physique des matériels, ainsi que leurs performances;
  - l'implantation des modules principaux sur les nœuds du réseau;
  - les exigences en terme de performances (bandes passantes, temps de réponse, performances du système tolérance aux fautes et pannes...).
- Les diagrammes de cette vue sont les diagrammes de composants et de déploiement.
- Cette vue est fort utile pour l'installation et la maintenance régulière du système.

Quelques méthodes Unifiées

Processus Unifié UP:

Le processus unifié (Unified process en anglais) est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel. Caractéristiques essentielles du processus unifié :

- Le processus unifié est à base de composants,
- Le processus unifié utilise le langage UML (ensemble d'outils et de diagramme),
- Le processus unifié est piloté par les cas d'utilisation,
- Centré sur l'architecture,
- Itératif et incrémental.

Cycle de vie du processus UP

Capture des besoins : ***Modèle des cas d'utilisation*** - décrit les besoins de l'utilisateur.

Analyse : ***Modèle d'analyse*** - définit la structure statique et le comportement dynamique des objets.

Conception :

- ***Modèle de conception*** - définit la structure statique du système en termes de sous-systèmes, de classes et d'interfaces et les collaborations entre les sous-systèmes, les classes et les interfaces.
- ***Modèle de déploiement*** - définit la disposition physique des différents matériels.

Implémentation : ***Modèle de réalisation*** - définit les composants de réalisation et le passage des classes vers ces composants.

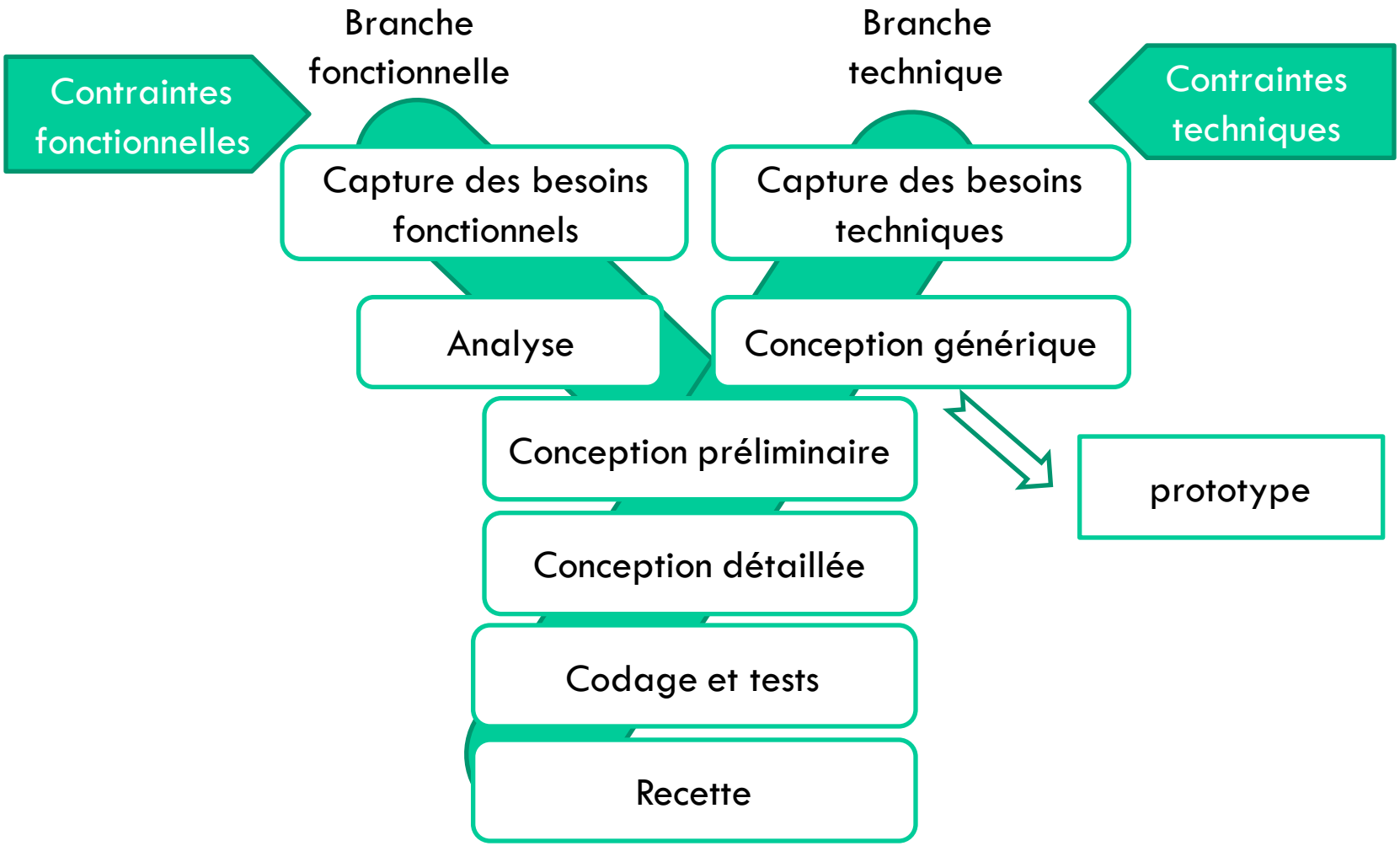
Test : ***Modèle de test*** - décrit les scénarios de test.

Processus 2TUP (« 2 Track Unified Process »)

Le Processus 2TUP (« 2 Track Unified Process ») est la méthode qui apporte une réponse aux contraintes de changement continuel imposées au système d'information de l'entreprise en suivant deux chemins. Il s'agit des chemins « **fonctionnels** » et « **d'architecture technique** » qui correspondent aux deux axes de changements imposés au système Informatique.

C'est un processus de développement en Y

Processus du développement en Y

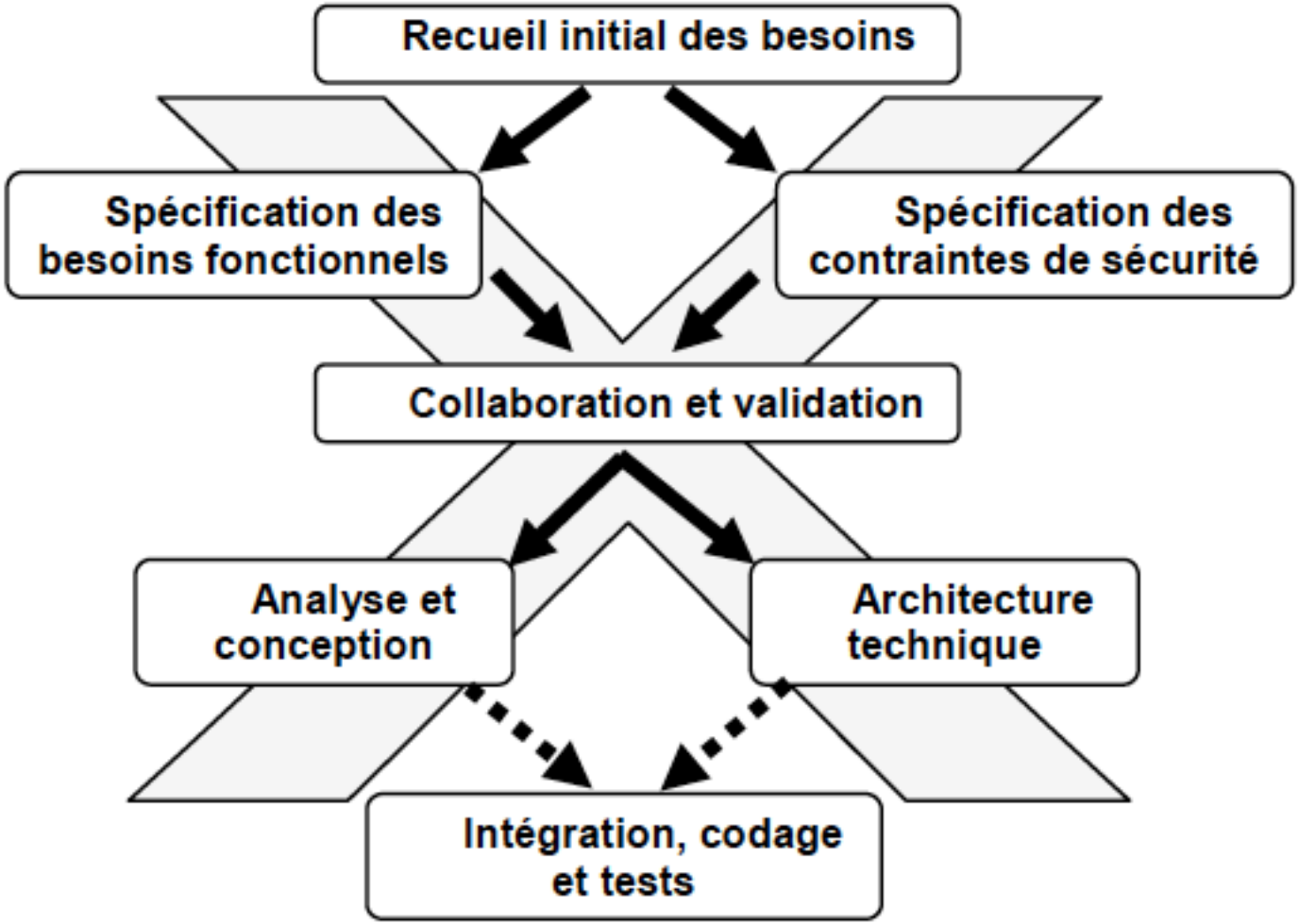


Processus en X ou XUP (X Unified Process)

- Le processus en X ou XUP (X Unified Process) est un processus de développement qui répond aux caractéristiques d'un processus UP et il est basé sur une approche qui permet de bien maîtriser, tout au long du cycle de développement du logiciel, l'assignation des tâches et la responsabilisation des différents acteurs participants.
- Le X signifie littéralement que le processus suit deux chemins en haut et deux chemins en bas ce que donne une forme en X.
- Les deux chemins du haut sont utilisés pour la spécification parallèle des besoins fonctionnels et des contraintes de sécurité, et les deux chemins du bas correspondent aux deux axes des changements imposés au système d'information : il s'agit de la vue logique qui décrit les aspects statiques et dynamiques du système en termes de classes et d'objets et la vue technique qui se préoccupe de la spécification de l'architecture technique du système.

Processus en X ou XUP (X Unified Process)

Les phases du processus de développement en X



Quelques méthodes agiles

https://ineumann.developpez.com/tutoriels/alm/agile_scrum/

Agilité : les 4 fondamentaux

Les méthodes agiles se reconnaissent toutes dans les valeurs suivantes :

L'équipe (« Personnes et interaction plutôt que processus et outils »)

L'application (« Logiciel fonctionnel plutôt que documentation complète »)

La collaboration (« Avec le client plutôt que négociation de contrat »)

L'acceptation du changement (« Réagir au changement plutôt que suivre un plan »)

Manifeste agile : Déclinés en 12 principes

Des pratiques communes liées :

1. aux ressources humaines
2. au pilotage du projet
3. à la qualité de la production

Satisfaire le client	Considérer comme naturel les changements d'exigences	Livrer fréquemment une application fonctionnelle	Fonctionnels et développeurs travaillent ensemble
Bâtissez le projet autour de personnes motivées	L'échange d'information le plus efficace est en face à face	Un logiciel fonctionnel est la meilleure façon de mesurer l'avancement du projet	Le rythme de développement doit être soutenable indéfiniment
Simplicité - l'art de maximiser la quantité de travail à ne pas faire - est essentielle	Architectures, spécifications et conceptions issues d'équipes auto-organisées	Vérifier en continue l'excellence des pratiques et techniques	Régulièrement, réflexion de l'équipe pour être plus efficace !

Processus XP (eXtreme Programming)

Extreme Programming est une méthode Agile basée sur un ensemble de pratiques destinées à organiser le travail d'une équipe de développement. Plus généralement, les pratiques XP sont sous-tendues par les quatre principes suivants (R.MEDINA, 2008) :

- **La Communication** (dans l'équipe, avec le client) : XP favorise le contact humain, la communication directe, plutôt que le cloisonnement des activités et les échanges de documents formels. Les développeurs travaillent directement avec le client et les testeurs sont intégrés à l'équipe de développement;
- **Le Feedback** (du client, de l'équipe) **:** Les pratiques XP sont conçues pour donner un maximum de feedback sur le déroulement du projet afin de corriger la trajectoire au plus tôt. En particulier, les points de début d'itération offrent à l'équipe le moyen de prendre du recul sur son fonctionnement et de l'améliorer sans cesse au fil des itérations;

R.MEDINA, « L'ExtremeProgramming », Cours Crossbow Labs, (2008).

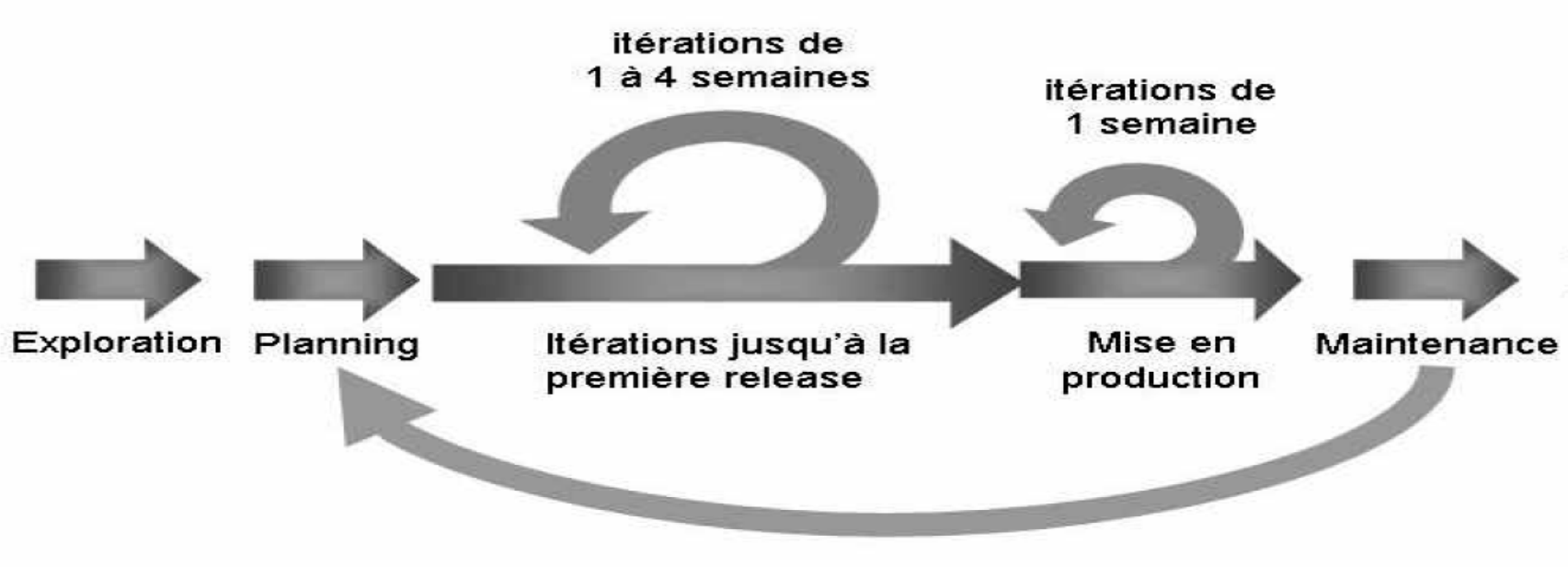
Processus XP (eXtreme Programming)

- **La Simplicité** (Economies, Précisions): XP relève le défi suivant : « que pouvons-nous arrêter de faire tout en continuant à créer efficacement un logiciel qui réponde aux besoins réels du client ? ». Cette recherche de simplification touche le processus lui-même, mais aussi l'outil fabriqué et la conception de l'application ;

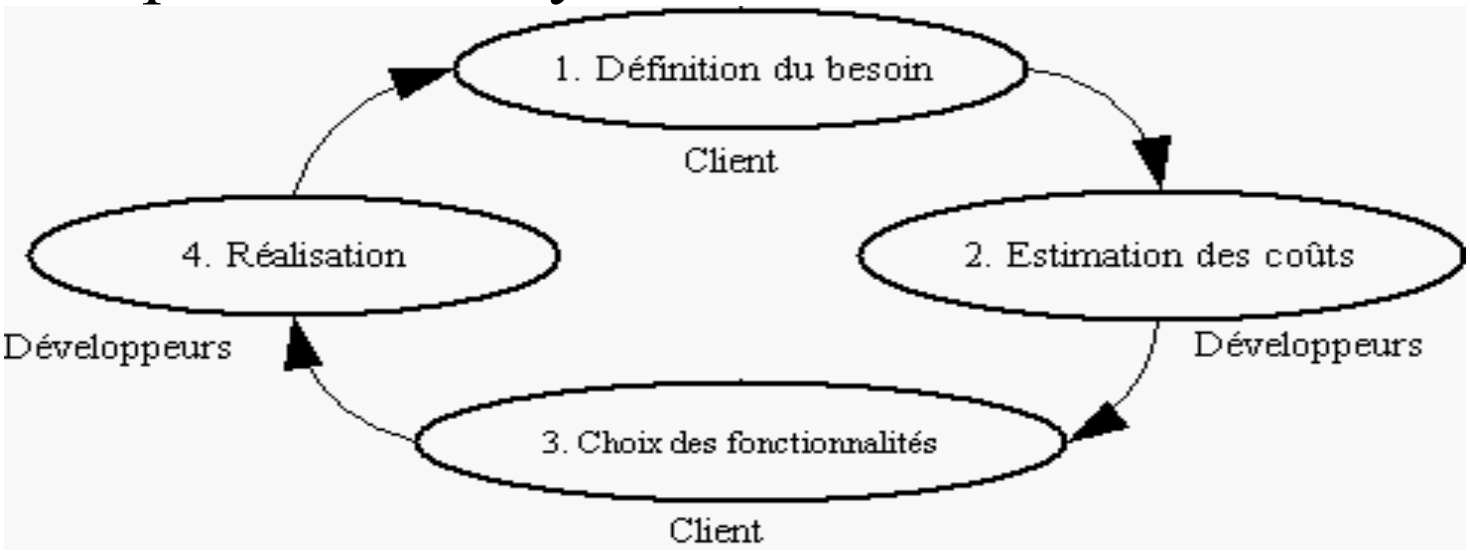
- **Le Courage** (Confiance en XP, Remise en cause des méthodes) : il s'agit principalement du courage d'honorer les autres valeurs – celui de maintenir une communication franche et ouverte, d'accepter et de traiter de front les mauvaises nouvelles, etc.

R.MEDINA, « L'ExtremeProgramming », Cours Crossbow Labs, (2008).

Cycle de vie d'un projet XP



Les 4 phases d'un cycle de livraison XP



Méthode SCRUM

La méthode Scrum est une méthode agile, dont le nom est un terme emprunté au rugby qui signifie « la mêlée ou encore Stand-up». Elle s'appuie sur le découpage des projets en itérations encore nommées « sprints ». Un sprint peut avoir une durée qui varie généralement entre deux semaines et un mois.

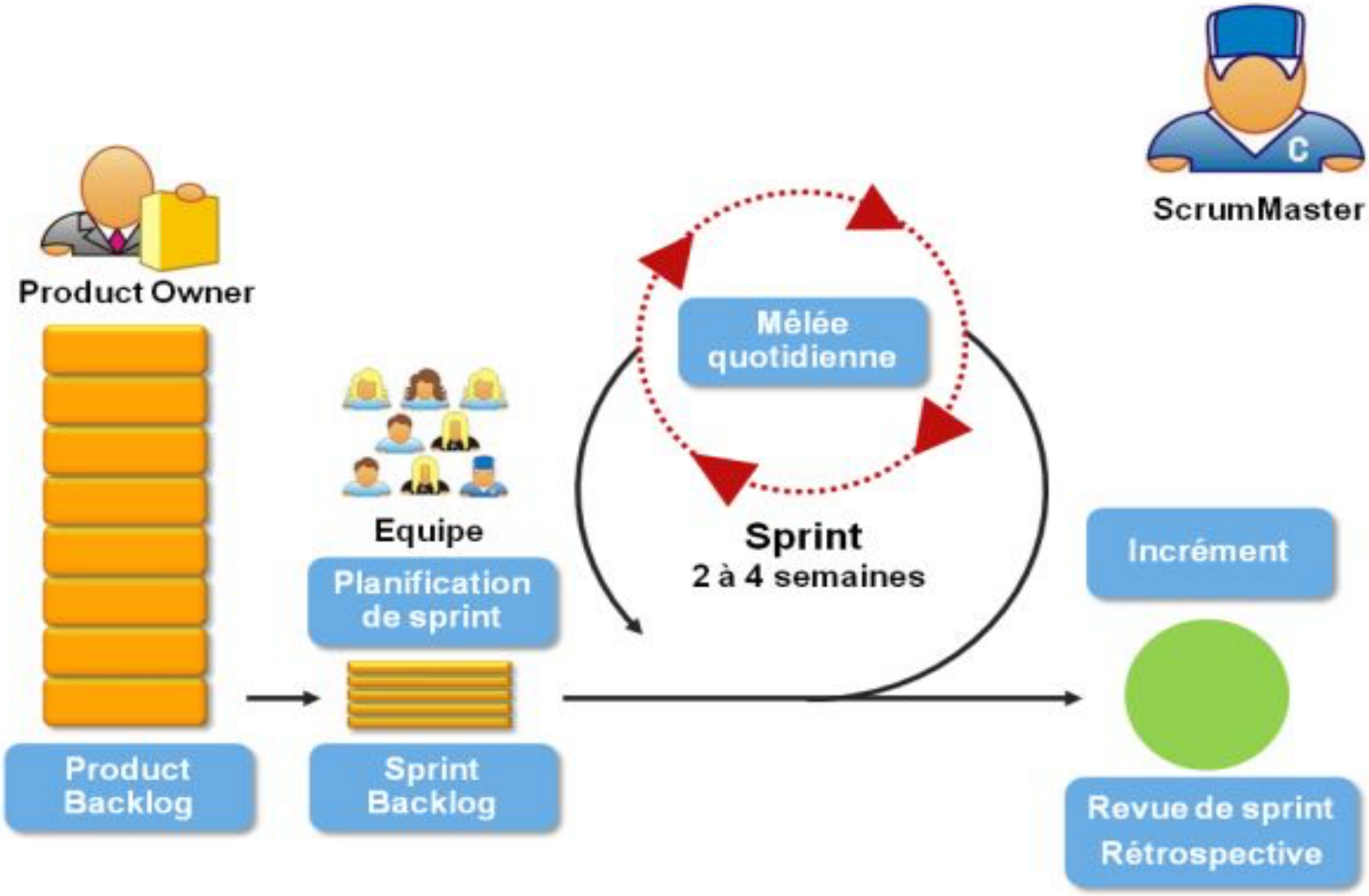
Principes :

- Plus orienté sur la gestion des ressources humaines
- Très complémentaire de la méthode XP
- Très peu formalisé: privilégie le management par rapport au développement

Pratique :

- Identification rapide des changements
- Donner/Faire confiance aux développeurs
- Réunion d'avancement tous les jours

Exemple de processus Scrum



Méthode SCRUM

Scrum définit seulement 3 rôles pour un projet :

- **Le Product Owner :** il s'agit du représentant officiel du client au sein d'un projet Scrum. Il est l'interlocuteur principal du Scrum Master et des membres de l'équipe. Il définit les besoins du produit et rédige les spécifications. Il s'agit généralement d'un expert du domaine métier du projet.
- **L'Equipe :** Ce sont les personnes chargées de transformer les besoins exprimés par le Product Owner en fonctionnalités utilisables. Il peut s'agir de développeurs, architectes, personnes chargées de faire des tests fonctionnels…
- **Le scrum Master :** il s'agit d'une personne chargée de veiller à la mise en application de la méthode et au respect de ses objectifs. Il a donc un rôle de coach à la fois auprès du Product Owner et auprès de l'équipe de développement. Il s'agit d'une personne chargée de lever les obstacles éventuels qui empêcherait l'avancement de l'équipe et du projet pendant les différents sprints.

Références

- Livre UML 2: UML 2 en action, de l'analyse des besoins à la conception J2EE, Pascal Roques et Franck Vallée.
- Livre *UML 2: Initiation, exemples et exercices Corrigés* de Laurrent Debrauwer et Fien Van der Heyde.
- Livre UML 2: *Entraînez-vous à la modélisation* de Laurrent Debrauwer et Naouel Karam.
- Livre MERISE et UML, pour la modélisation des systèmes d'information de Joseph Gabay.
- Livre Base de données objet & relationnel de George Gardarin
- Cours UML 2.0, Pr. Moussa LO, Université Gaston Berger de Saint-Louis.
- Cours Laurent Piechocki sur le site : de http://uml.free.fr