Partie I : Le processus UP

I. Définition du processus unifié UP

La maîtrise des processus de développement implique une organisation et un suivi des activités : c'est ce à quoi s'attachent les différentes méthodes qui s'appuient sur l'utilisation du langage UML pour modéliser un système d'information. UP (Unified Process) est une méthode générique de développement de logiciel. Générique signifie qu'il est nécessaire d'adapter UP au contexte du projet, de l'équipe, du domaine et/ou de l'organisation (exemple : R.UP, X.UP, 2 TUP…).

1. Les caractéristiques d'UP

Le processus unifié est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.

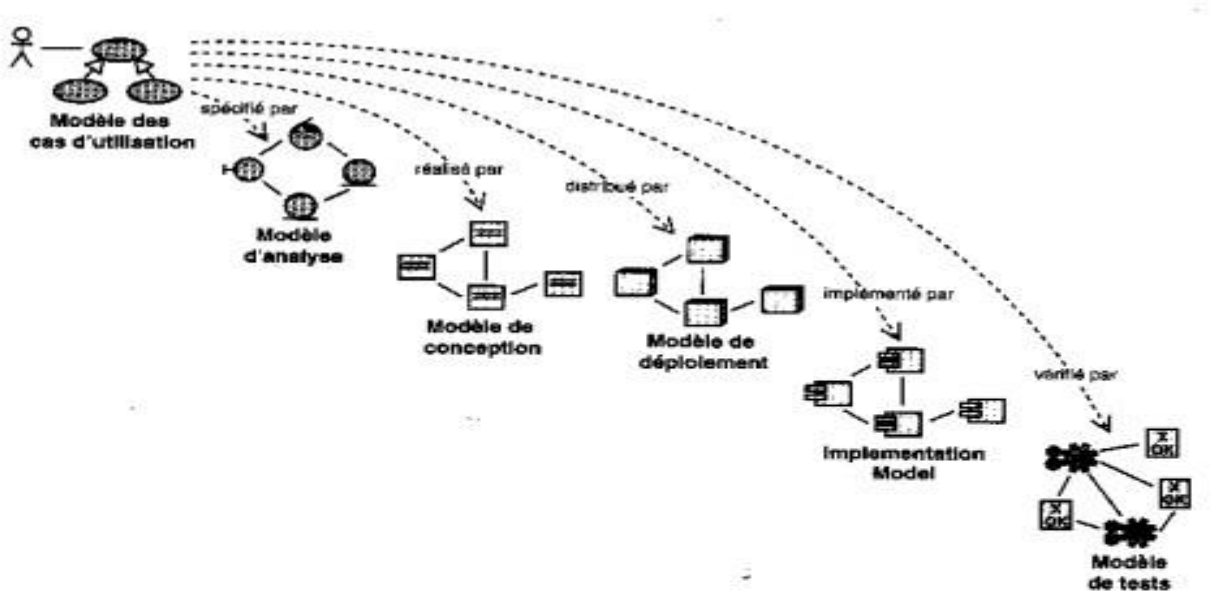
Les caractéristiques essentielles du processus unifié sont :

- Il est à base de composants,
- Il utilise le langage UML (ensemble d'outils et de diagramme),
- Il est piloté par les cas d'utilisation,
- Il est centré sur l'architecture,
- Il est itératif et incrémental.

a) **Le processus unifie est pilote par les cas d'utilisation**

L'objectif principal d'un système logiciel est de rendre service à ses utilisateurs ; il faut par conséquent bien comprendre les désirs et les besoins des futurs utilisateurs. Le processus de développement sera donc centré sur l'utilisateur. Le terme utilisateur ne désigne pas seulement les utilisateurs humains mais également les autres systèmes.

Les cas d'utilisation ne sont pas un simple outil de spécification des besoins du système. Ils vont complètement guider le processus de développement à travers l'utilisation de modèles basés sur l'utilisation du langage UML.



- A partir du modèle des cas d'utilisation, les développeurs créent une série de modèles de conception et d'implémentation réalisant les cas d'utilisation.
- Chacun des modèles successifs est ensuite révisé pour en contrôler la conformité par rapport au modèle des cas d'utilisation.
- Enfin, les testeurs testent l'implémentation pour s'assurer que les composants du modèle d'implémentation mettent correctement en œuvre les cas d'utilisation.

b) **Le processus unifie est centre sur l'architecture**

Dès le démarrage du processus, on aura une vue sur l'architecture à mettre en place. L'architecture d'un système logiciel peut être décrite comme les différentes vues du système qui doit être construit. L'architecture logicielle équivaut aux aspects statiques et dynamiques les plus significatifs du système. L'architecture émerge des besoins de l'entreprise, tels qu'ils sont exprimés par les utilisateurs et autres intervenants et tels qu'ils sont reflétés par les cas d'utilisation.

Elle subit également l'influence d'autres facteurs :

- la plate-forme sur laquelle devra s'exécuter le système ;
- les briques de bases réutilisables disponibles pour le développement ;
- les considérations de déploiement, les systèmes existants et les besoins non fonctionnels (performance, fiabilité..).

c) **Le processus unifie est itératif et incrémental**

Le développement d'un produit logiciel destiné à la commercialisation est une vaste entreprise qui peut s'étendre sur plusieurs mois. On ne va pas tout développer d'un coup. On peut découper le

travail en plusieurs parties qui sont autant de mini projets. Chacun d'entre eux représentant une itération qui donne lieu à un incrément.

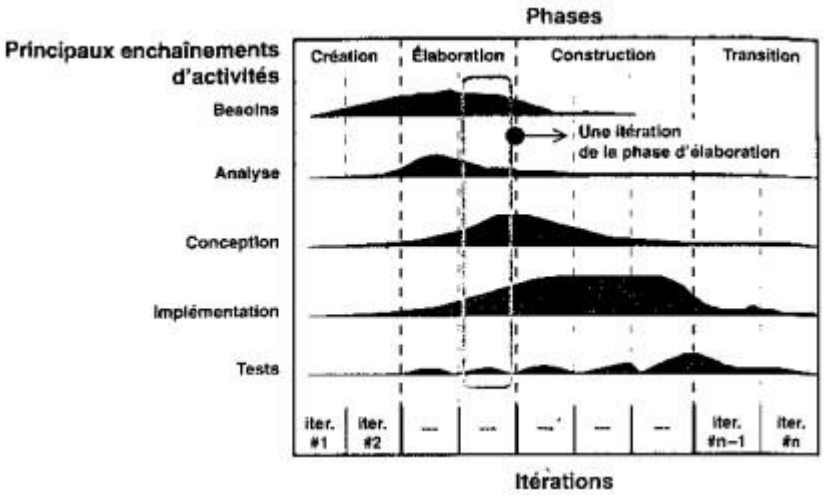
Une itération désigne la succession des étapes de l'enchaînement d'activités, tandis qu'un incrément correspond à une avancée dans les différents stades de développement. Le choix de ce qui doit être implémenté au cours d'une itération repose sur deux facteurs :

- Une itération prend en compte un certain nombre de cas d'utilisation qui ensemble, améliorent l'utilisabilité du produit à un certain stade de développement.
- L'itération traite en priorité les risques majeurs.

Un incrément constitue souvent un additif. A chaque itération, les développeurs identifient et spécifient les cas d'utilisations pertinents, créent une conception en se laissant guider par l'architecture choisie, implémentent cette conception sous forme de composants et vérifie que ceux-ci sont conformes aux cas d'utilisation. Dès qu'une itération répond aux objectifs fixés le développement passe à l'itération suivante. Pour rentabiliser le développement il faut sélectionner les itérations nécessaires pour atteindre les objectifs du projet. Ces itérations devront se succéder dans un ordre logique. Un projet réussi suivra un déroulement direct, établi dès le début par les développeurs et dont ils ne s'éloigneront que de façon très marginale. L'élimination des problèmes imprévus fait partie des objectifs de réduction des risques.

2. Le cycle de vie d'UP

Présentation



Phase	Description et Enchaînement d'activités
Phase de création	Traduit une idée en vision de produit fini et présente une étude de rentabilité pour ce produit - Que va faire le système pour les utilisateurs ? - A quoi peut ressembler l'architecture d'un tel système ? - Quels sont l'organisation et les coûts du développement de ce produit ? On fait apparaître les principaux cas d'utilisation. L'architecture est provisoire, identification des risques majeurs et planification de la phase d'élaboration.
Phase d'élaboration	Permet de préciser la plupart des cas d'utilisation et de concevoir l'architecture du système. L'architecture doit être exprimée sous forme de vue de chacun des modèles. Émergence d'une **architecture de référence**. A l'issue de cette phase, le chef de projet doit être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet.
Phase de construction	Moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet, elle est maintenant stable. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs ont décidé de mettre au point pour cette version. Celle ci doit encore avoir des anomalies qui peuvent être en partie résolue lors de la phase de transition.
Phase de transition	Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts. Cette phase suppose des activités comme la fabrication, la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.(où le report de leur correction à la version suivante)

Description

Le processus unifié répète un certain nombre de fois une série de cycles. Tout cycle se conclut par la livraison d'une version du produit aux clients et s'articule en 4 phases : création, élaboration, construction et transition, chacune d'entre elles se subdivisant à son tour en itérations. Pour mener efficacement le cycle, les développeurs ont besoin de construire toutes les représentations du produit logiciel :

Modèle des cas d'utilisation	Expose les cas d'utilisation et leurs relations avec les utilisateurs
Modèle d'analyse	Détaille les cas d'utilisation et procède à une première répartition du comportement du système entre divers objets
Modèle de conception	Définit la structure statique du système sous forme de sous système, classes et interfaces ; Définit les cas d'utilisation réalisés sous forme de collaborations entre les sous systèmes les classes et les interfaces
Modèle d'implémentation	Intègre les composants (code source) et la correspondance entre les classes et les composants
Modèle de déploiement	Définit les nœuds physiques des ordinateurs et l'affectation de ces composants sur ces nœuds.
Modèle de test	Décrit les cas de test vérifiant les cas d'utilisation
Représentation de l'architecture	Description de l'architecture

3. **Exemple de processus de développement logiciel qui implémente le Processus Unifié : 2TUP (Two tracks unified process)**

Dans le processus 2TUP, les activités de développement sont organisées suivant 5 workflows qui décrivent :

- La capture des besoins,
- L'analyse,
- La conception,
- ➢ L'implémentation
- ➢ Et le test.

Le processus 2TUP est une trame des meilleures pratiques de développement, il doit être utilisé comme un guide pour réaliser un projet et non comme l'arme ultime et universelle de développement.

A. Le cycle de développement avec 2TUP

Le 2TUP propose un cycle de développement en Y, qui dissocie les aspects techniques des aspects fonctionnels. Il commence par une étude préliminaire qui consiste essentiellement à identifier les acteurs qui vont interagir avec le système à construire, les messages qu'échangent les acteurs et le système. En suite à produire le cahier des charges et à modéliser le contexte. Le processus s'articule autour de 3 phases essentielles : une branche technique ; une branche fonctionnelle et une phase de réalisation.

i. La branche fonctionnelle

La branche fonctionnelle capitalise la connaissance du métier de l'entreprise. Cette branche comporte :

- La capture des besoins fonctionnels, qui produit un modèle des besoins focalisé sur le métier des utilisateurs. Elle qualifie au plus tôt le risque de produire un système inadapté aux utilisateurs. De son côté, la maîtrise d'œuvre consolide les spécifications et en vérifie la cohérence et l'exhaustivité.
- L'analyse, qui consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système en termes de métier. Les résultats de l'analyse ne dépendent d'aucune technologie particulière.

ii. La branche technique

La branche technique capitalise un savoir-faire technique et/ou des contraintes techniques. Les techniques développées pour le système le sont indépendamment des fonctions à réaliser. Cette branche droite comporte :

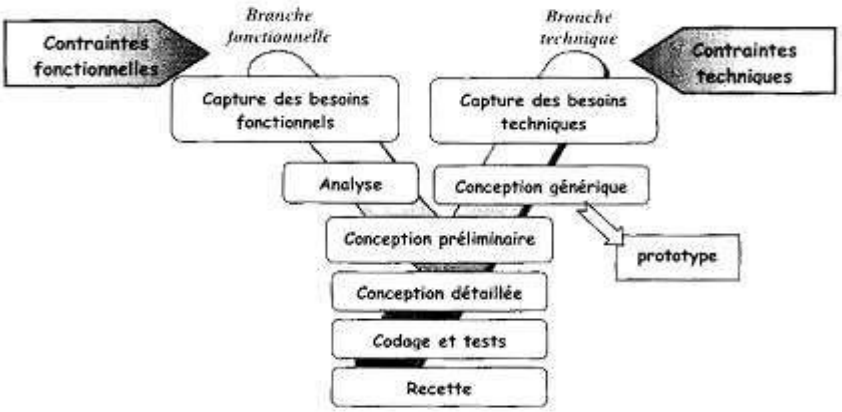
- La capture des besoins techniques, qui recense toutes les contraintes et les choix dimensionnant la conception du système. Les outils et les matériels sélectionnés ainsi que la prise en compte de contraintes d'intégration avec l'existant conditionnent généralement des pré-requis d'architecture technique ;
- La conception générique, qui définit ensuite les composants nécessaires à la construction de l'architecture technique. Celle conception est complètement indépendante des aspects fonctionnels. Elle a pour objectif d'uniformiser et de réutiliser les mêmes mécanismes pour tout un système. L'architecture technique construit le squelette du système informatique et écarte la plupart des risques de niveau technique. L'importance de sa réussite est telle qu'il est conseillé de réaliser un prototype pour assurer sa validité.

iii. La phase de réalisation

La phase de réalisation consiste à réunir les deux branches, permettant de mener une conception applicative et enfin la livraison d'une solution adaptée aux besoins. La branche du milieu comporte :

- La conception préliminaire, qui représente une étape délicate, car elle intègre le modèle d'analyse dans l'architecture technique de manière à tracer la cartographie des composants du système à développer,
- La conception détaillée, qui étudie ensuite comment réaliser chaque composant ;
- L'étape de codage, qui produit ces composants et teste au fur et à mesure les unités de code réalisées,
- L'étape de recette, qui consiste enfin à valider les fonctions du système développé.

iv. Représentation



4. Eléments de comparaisons entre MERISE et UML

MERISE : Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprises	UML : Unified Modeling Language
MERISE est une méthode systémique d'analyse et de conception de systèmes d'information. C'est-à-dire qu'elle utilise une approche systémique.	UML n'est cependant pas une méthode mais plutôt un langage de modélisation objet à qui il faut associer une démarche pour en faire une méthode. C'est le cas de la méthode 2TUP, RUT et XP.
MERISE propose de considérer le système réel selon deux points de vue : - Une vue statique (données) - Une vue dynamique (traitements). C'est-à-dire qu'avec la méthode MERISE, nous avons une étude séparée des données et des traitements.	UML propose une approche différente de celle de MERISE en ce sens qu'il associe les données et les traitements. Car avec UML, centraliser les données d'un type et les traitements associés permet de limiter les points de maintenance dans le code et facilite l'accès à l'information en cas d'évolution du logiciel. De plus, UML décrit la dynamique du système d'information comme un ensemble d'opérations attachées aux objets du système.