Chapitre 1

Processus de développement UP

I. Pourquoi UP ?
II. Définition
III. Activités et phases
IV. Modèles mis en place

1. Pourquoi UP ?

Les notions de base acquises dans le module ACOO1, notamment la notation UML, facilitent la compréhension et l'adoption d'une méthodologie orientée objet du développement logiciel qui s'appuie sur la modélisation des objets du monde réel, puis sur l'utilisation du modèle pour bâtir une conception indépendante des langages de programmation, organisée autour de ces objets.

Les bonnes méthodes d'analyse et de conception doivent fournir une méthodologie et des notations standard qui aident à concevoir des logiciels de qualité.

UML n'est pas une méthode (i.e. une description normative des étapes de la modélisation). C'est un langage graphique qui permet de représenter (modéliser) et de communiquer les divers aspects d'un système d'information.

Les concepteurs orientent leurs modélisations selon trois axes sur lesquels ils répartissent les diagrammes :

- L'axe fonctionnel qui est utilisé pour décrire ce que fait le système à réaliser,
  Diagramme de Use Cases
  (Diagramme d'activités)
  (Diagramme de séquences)

- L'axe structurel et statique qui est relatif à la structure du système,
  Diagramme de classes
  Diagramme de composants
  Diagramme de déploiement
  Diagramme d'objets

- L'axe dynamique qui est relatif à la construction des fonctionnalités du système.
  Diagramme de séquences
  Diagramme de collaboration
  Diagramme d'activités
  (Diagramme d'états/transitions)

Si UML permet de modéliser un système, il ne définit pas le processus d'élaboration des modèles.

- Dans quel ordre doit-on utiliser les neufs types de diagrammes?
- A quel moment de la conception d'un système doivent-ils intervenir?

Seul un processus de développement peut répondre à ces questions !
Le processus de développement régit les activités de production du logiciel selon deux aspects.

- L'aspect statique qui représente le processus en terme de tâches à réaliser.
  L'aspect statique définit:
  - Le « qui ». Les intervenants
  - Le « comment ». Les activités à réaliser
  - Le « quoi ». Les résultats d'une activité

- L'aspect dynamique qui représente la dimension temporelle du processus.
  L'aspect dynamique représente :
  - Le nom et le nombre de phases du cycle de vie du processus
  - L'organisation et l'ordre de réalisation des activités de développement

Un processus gère généralement les activités suivantes:

- **L'expression des besoins** consiste pour le client à élaborer un cahier des charges décrivant:
  - Les fonctionnalités du système à étudier.
  - La façon d'utiliser le système
- **La spécification des besoins** permet aux utilisateurs, experts et aux informaticiens de finaliser le cahier des charges en levant les ambiguïtés et en éliminant les redondances
- **L'analyse des besoins** vise à faire définir, par les experts et les utilisateurs les entités métier concernées par le système indépendamment de toutes considérations:
  - techniques
  - et informatiques
- **La conception** concerne les experts informatiques. On y détermine la manière de résoudre techniquement le problème posé. Comment réaliser les fonctionnalités attendues.
- **L'implémentation** consiste :
  - A construire les programmes dans un langage de programmation donné.
  - A organiser logiquement les programmes en fonction de l'architecture logique choisie.
  - A distribuer les programmes sur le système informatique selon l'architecture physique retenue.
- **Les tests** fonctionnels et techniques
  - Fonctionnels. Ils vérifient que le système implémente bien les fonctionnalités attendues.
  - Techniques. Ils vérifient que l'implémentation des fonctionnalités est techniquement correcte.
- **La maintenance**

UML et les activités de développement :

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.
Pour utiliser UML, ses auteurs insistent sur les caractéristiques suivantes qui sont essentielles pour un processus de développement :

1 Piloté par les cas d'utilisation
2 Centré sur l'architecture
3 Itératif et incrémental

Le Processus Unifié **UP (Unified Process)** répond bien à ces exigences. C'est un processus de développement moderne, itératif, efficace sur des projets informatiques de toutes tailles. Très complet, il couvre l'ensemble des activités, depuis la conception du projet jusqu'à la livraison de la solution. Intégrant une organisation de projet type, une méthodologie utilisant UML et un ensemble de bonnes pratiques cohérentes entre elles, il permet de circonvenir aux problèmes récurrents que rencontrent nombre de réalisations : dérive des coûts et des délais, qualité insuffisante, réponse incomplète aux attentes des utilisateurs. Un point d'excellence de cette démarche est son adaptabilité : UP peut se décliner en fonction de l'ampleur d'un projet, de l'expérience de l'équipe qui l'assume, de la nature de la solution à construire.

UP est une méthode générique de développement de logiciel.

2. Définitions

Processus de développement logiciel :
Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant.
L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles. Des techniques d'analyse et de conception permettant de bien prendre en charge les critères de qualité sont indispensables. En conséquence, le processus peut se décomposer suivant deux axes de contrôle sur le développement :

- L'axe de développement technique, qui se concentre principalement sur la qualité de la production ;
- L'axe de gestion du développement, qui permet la mesure et la prévision des coûts et des délais.

Processus Unifié : **(Unified Process)**
Le processus unifié est un processus de développement logiciel construit sur UML. Il est itératif, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques. Il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.

C'est un patron de processus pouvant être adaptée à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétences et à différentes tailles de l'entreprise.

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.

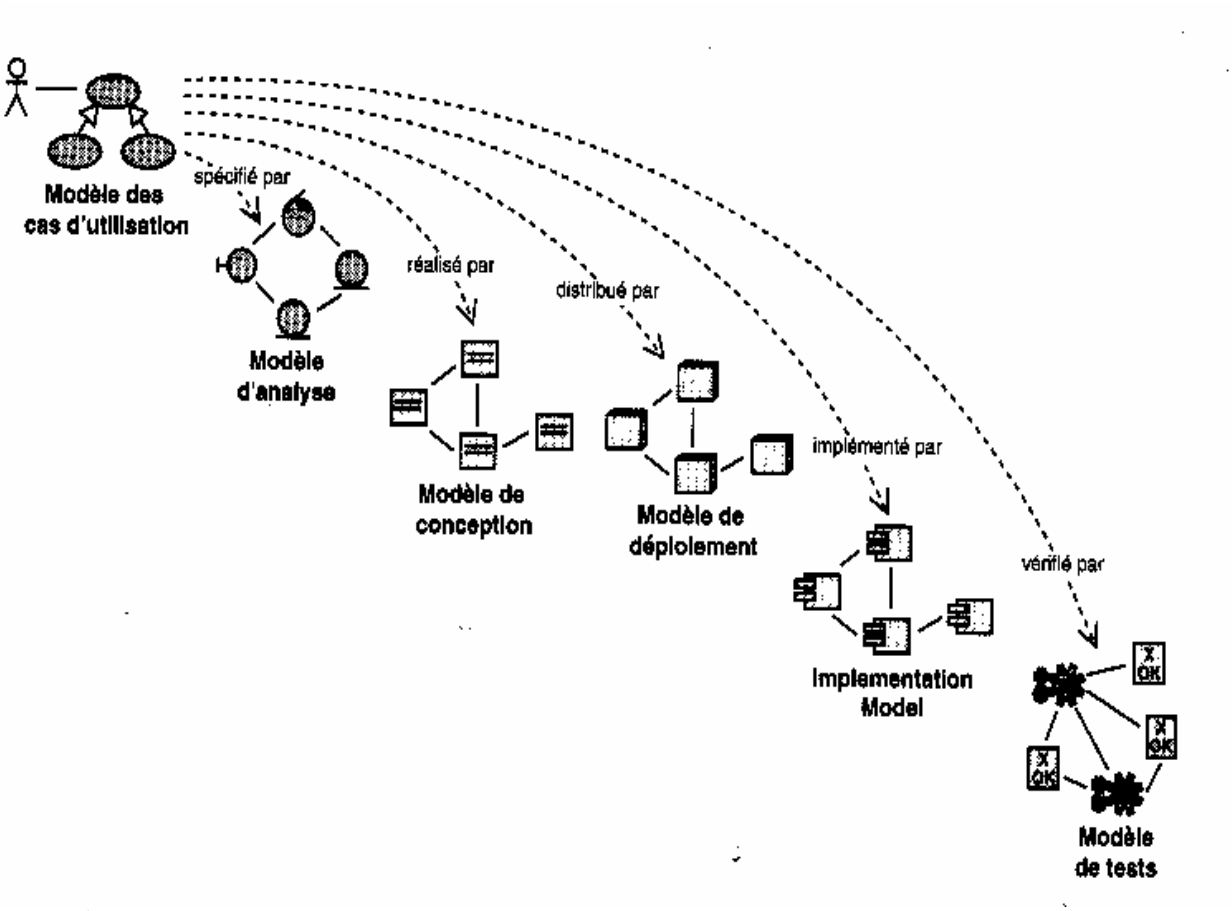
Les adaptations de UP les plus connues sont :

- RUP : Rational Unified Process
- XP : eXtreme Programming
- 2TUP : Two Tracks Unified Process

2. 1. UP est piloté 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. L'utilisateur représente donc une personne ou une chose dialoguant avec le système en cours de développement.
Les cas d'utilisation permettent d'illustrer les besoins. Ils détectent puis décrivent les besoins fonctionnels (du point de vue de l'utilisateur), et leur ensemble constitue le modèle de cas d'utilisation qui dicte les fonctionnalités complètes du système.

Stratégie des cas d'utilisation : *Que doit pouvoir faire le système pour chaque utilisateur ?*
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.

2. 2. UP est centré 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…).

Tout produit est à la fois forme et fonction. Les cas d'utilisation doivent une fois réalisés, trouver leur place dans l'architecture. L'architecture doit prévoir la réalisation de tous les cas d'utilisation. L'architecture et les cas d'utilisation doivent évoluer de façon concomitante.

- L'architecte crée une ébauche grossière de l'architecture, en partant de l'aspect qui n'est pas propre aux cas d'utilisation (plate forme..). Bien que cette partie de l'architecture soit indépendante des cas d'utilisation. L'architecte doit avoir une compréhension globale de ceux ci avant d'en esquisser l'architecture.

- Il travaille ensuite, sur un sous ensemble des cas d'utilisations identifiés, ceux qui représentent les fonctions essentielles du système en cours de développement.

- L'architecture se dévoile peu à peu, au rythme de la spécification et de la maturation des cas d'utilisation, qui favorisent, à leur tour, le développement d'un nombre croissant de cas d'utilisation.

Ce processus se poursuit jusqu'à ce que l'architecture soit jugée stable.

2. 3. UP est itératif et incrémental

L'itération est une répétition d'une séquence d'instructions ou d'une partie de programme un nombre de fois fixé à l'avance ou tant qu'une condition définie n'est pas remplie, dans le but de reprendre un traitement sur des données différentes. Elle qualifie un traitement ou une procédure qui exécute un groupe d'opérations de façon répétitive jusqu'à ce qu'une condition bien définie soit remplie.
Une itération prend en compte un certain nombre de cas d'utilisation et traite en priorité les risques majeurs.
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 vérifient que ceux ci sont conformes aux cas d'utilisation. Dés 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 dés 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.

Avantages d'un processus itératif contrôlé :

- Permet de limiter les coûts, en termes de risques, aux strictes dépenses liées à une itération.

- Permet de limiter les risques de retard de mise sur le marché du produit développé (identification des problèmes dès les premiers stades de développement et non en phase de test comme avec l'approche « classique »).

- Permet d'accélérer le rythme de développement grâce à des objectifs clairs et à court terme.

- Permet de prendre en compte le fait que les besoins des utilisateurs et les exigences correspondantes ne peuvent être intégralement définis à l'avance et se dégagent peu à peu des itérations successives.

3. Activités et Phases

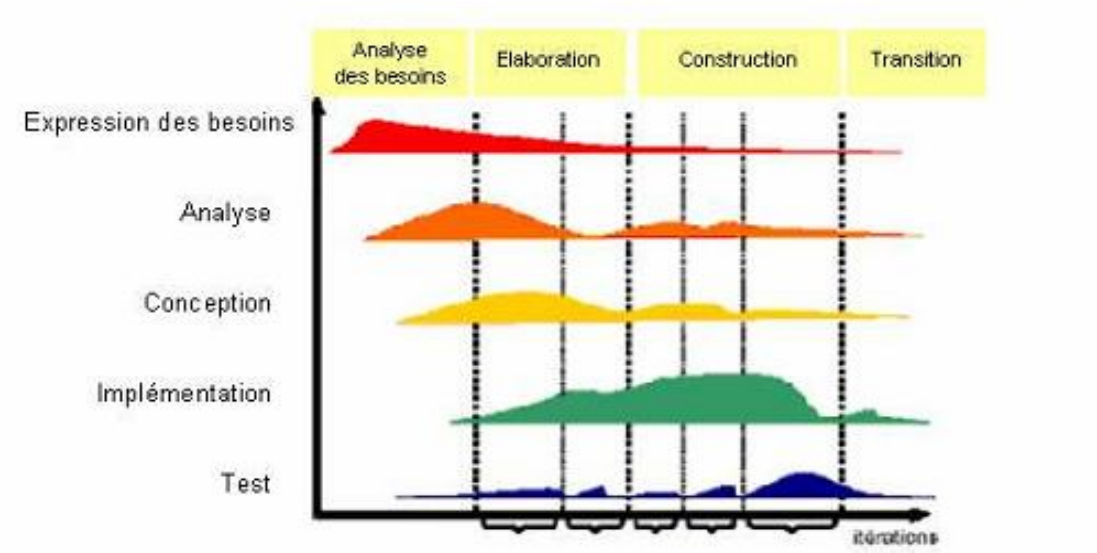
L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques.

UP est un ensemble de principes génériques adapté en fonctions des spécificités des projets. UP répond aux préoccupations suivantes :

- **QUI** participe au projet ?
- **QUOI**, qu'est-ce qui est produit durant le projet ?
- **COMMENT** doit-il être réalisé ?
- **QUAND** est réalisé chaque livrable ?

UP gère le développement selon deux axes.

- L'axe vertical représente les principaux enchaînements d'activités, qui regroupent les activités selon leur nature. Cette dimension rend compte l'aspect statique du processus qui s'exprime en terme de composants, de processus, d'activités, d'enchaînements, d'artefacts et de travailleurs.
- L'axe horizontal représente le temps et montre le déroulement du cycle de vie du processus; cette dimension rend compte de l'aspect dynamique du processus qui s'exprime en terme de cycles, de phases, d'itérations et de jalons.



UP répète un certain nombre de fois une série de cycle qui s'articule autours de 4 phases

- analyse des besoins
- élaboration
- construction
- transition

3. 1. Les Activités

Expression des besoins :

L'expression des besoins permet de:

- inventorier les besoins principaux et fournir une liste de leurs fonctions
- recenser les besoins fonctionnels (du point de vue de l'utilisateur) qui conduisent à l'élaboration des modèles de cas d'utilisation
- appréhender les besoins non fonctionnels (technique) et livrer une liste des exigences.

Le modèle de cas d'utilisation présente le système du point de vue de l'utilisateur et représente sous forme de cas d'utilisation et d'acteur, les besoins du client.

Analyse

L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client. Il s'agit de réaliser des spécifications permettant de concevoir la solution.
Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation et les Structures sous une forme qui facilite la compréhension (scénarios), la préparation (définition de l'architecture), la modification et la maintenance du futur système. Il peut être considéré comme une première ébauche du modèle de conception.

Conception
La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation. Elle détermine les principales interfaces et les transcrit à l'aide d'une notation commune.
Elle constitue un point de départ à l'implémentation :

- elle décompose le travail d'implémentation en sous-système
- elle créée une abstraction transparente de l'implémentation

Implémentation
L'implémentation est le résultat de la conception pour implémenter le système sous formes de composants, c'est-à-dire, de code source, de scripts, de binaires, d'exécutables et d'autres éléments du même type. Les objectifs principaux de l'implémentation sont de planifier les intégrations des composants pour chaque itération, et de produire les classes et les sous-systèmes sous formes de codes sources.

Test
Les tests permettent de vérifier des résultats de l'implémentation en testant la construction. Pour mener à bien ces tests, il faut les planifier pour chaque itération, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun.

3. 2. Les Phases

Analyse des besoins
L'analyse des besoins donne une vue du projet sous forme de produit fini.
Cette phase porte essentiellement sur les besoins principaux (du point de vue de l'utilisateur), l'architecture générale du système, les risques majeurs, les délais et les coûts (On met en place le projet).
Elle répond aux questions suivantes :

- que va faire le système ? Par rapport aux utilisateurs principaux, quels services va-t-il rendre?
- quelle va être l'architecture générale (cible) de ce système
- quels vont être : les délais, les coûts, les ressources, les moyens à déployer?

Elaboration
L'élaboration reprend les éléments de la phase d'analyse des besoins et les précise pour arriver à une spécification détaillée de la solution à mettre en œuvre.
L'élaboration permet de préciser la plupart des cas d'utilisation, de concevoir l'architecture du système et surtout de déterminer l'architecture de référence.
Au terme de cette phase, les chefs de projet doivent être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet.
Les taches à effectuer dans la phase élaboration sont les suivantes :

- créer une architecture de référence
- identifier les risques, ceux qui sont de nature à bouleverser le plan, le coût et le calendrier
- définir les niveaux de qualité à atteindre

- formuler les cas d'utilisation pour couvrir les besoins fonctionnels et planifier la phase de construction
- élaborer une offre abordant les questions de calendrier, de personnel et de budget

Construction

La construction est le moment où l'on construit le produit (architecture= produit complet). 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.

Transition

Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts.
Cette phase suppose des activités comme la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.

4. Modèles mis en place

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

Tous ces modèles sont liés. Ensemble, ils représentent le système comme un tout. Les éléments de chacun des modèles présentent des dépendances de traçabilité ; ce qui facilite la compréhension et les modifications ultérieures.