
UP : Unified Process

Table des matières

1	DÉFINITION.....	2
1.1	UP est itératif	2
1.2	UP est centré sur l'architecture.....	2
1.3	UP est piloté par les cas d'utilisation d'UML.....	2
2	VIE DU PROCESSUS UNIFIÉ.....	3
2.1	L'architecture bidirectionnelle.....	3
3	LES ACTIVITÉS.....	4
3.1	Expression des besoins.....	4
3.2	Analyse.....	4
3.3	Conception.....	5
3.4	Implémentation.....	5
3.5	Test.....	5
4	LES PHASES.....	5
4.1	Analyse des besoins.....	5
4.2	Elaboration.....	5
4.3	Construction.....	6
4.4	Transition.....	6
5	ANNEXE 1.....	7

1 Définition

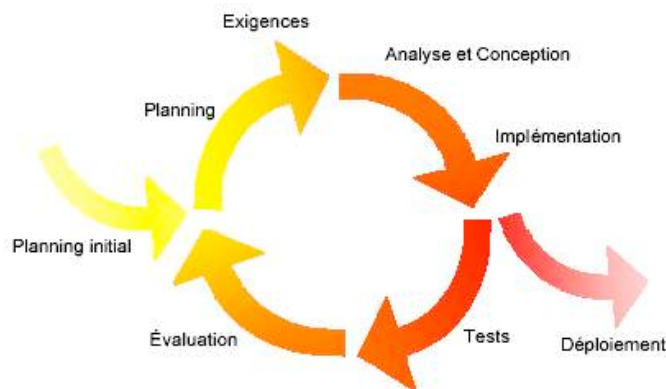
Le processus unifié est un processus de développement logiciel itératif, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques.

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.

1.1 UP est itératif

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.

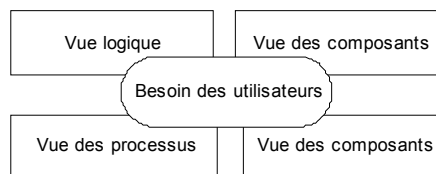


Une itération prend en compte un certain nombre de cas d'utilisation et traite en priorité les risques majeurs.

1.2 UP est centré sur l'architecture

Ph. Kruchten propose différentes perspectives, indépendantes et complémentaires, qui permettent de définir un modèle d'architecture (publication IEEE, 1995).

<http://uml.free.fr/cours/i-p7.html>

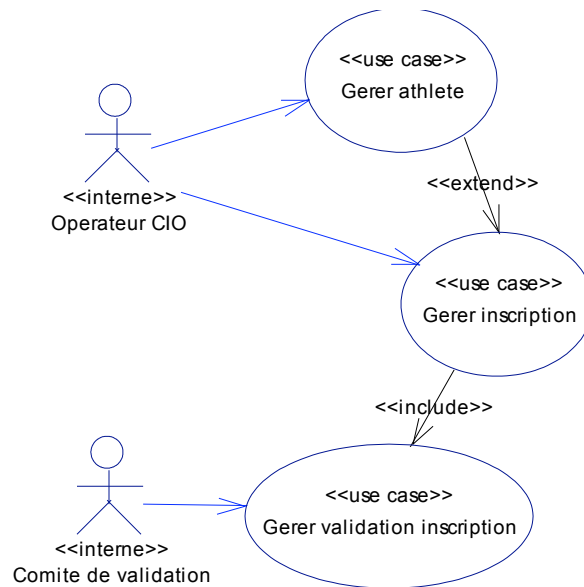


1.3 UP est piloté par les cas d'utilisation d'UML

Le but principal d'un système informatique est de satisfaire les besoins du client. Le processus de développement sera donc accès sur l'utilisateur.

Les cas d'utilisation permettent d'illustrer ces 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.



Exemple : gestion de l'inscription d'un athlète aux Jeux Olympique

2 Vie du processus unifié

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 :

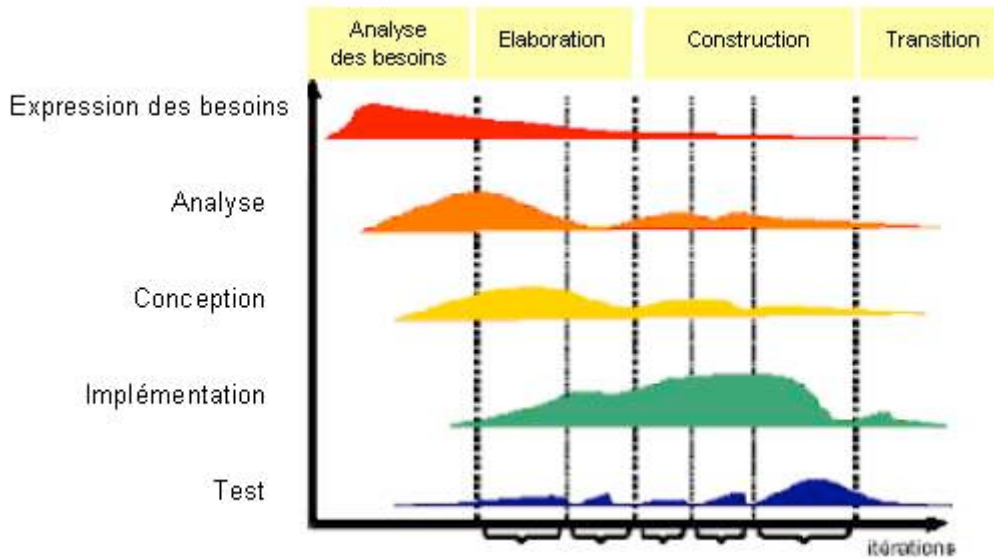
- **QUI** participe au projet ?
- **QUOI**, qu'est-ce qui est produit durant le projet ?
- **COMMENT** doit-il être réalisé ?
- **QUAND** est réalisé chaque livrable ?

2.1 L'architecture bidirectionnelle

UP gère le processus de développement par 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 autour de 4 phases

- analyse des besoins
- élaboration
- construction
- transition

Pour mener efficacement un tel cycle, les développeurs ont besoins de toutes les représentations du produit logiciel

- un modèle de cas d'utilisation
- un modèle d'analyse : détailler les cas d'utilisation et procéder à une première répartition du comportement
- un modèle de conception : finissant la structure statique du système sous forme de sous-systèmes, de classes et interfaces.
- un modèle d'implémentation : intégrant les composants
- un modèle de déploiement : définissant les nœuds physiques des ordinateurs
- un modèle de test : décrivant les cas de test vérifiant les cas d'utilisation
- une représentation de l'architecture

3 Les activités

3.1 Expression des besoins

L'expression des besoins comme son nom l'indique, permet de définir les différents besoins :

- 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.

3.2 Analyse

L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client. Il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution.

Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation et les structure sous une forme qui facilite la compréhension (scénarios), la préparation (définition de l'architecture), la modification et la maintenance du futur système.

Il s'écrit dans le langage des développeurs et peut être considéré comme une première ébauche du modèle de conception.

3.3 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 crée une abstraction transparente de l'implémentation

3.4 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.

3.5 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.

4 Les phases

4.1 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?

4.2 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 tâches 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

4.3 Construction

La construction est le moment où l'on construit le produit. L'architecture de référence se métamorphose en 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.

4.4 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 formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.

5 Annexe 1

