

70. Concepts de l'approche orientée objet de développement de systèmes d'information informatisés (SII)

1 Préambule

L'approche orientée objet de développement de système d'information informatisé (SII) de gestion est actuellement influencée par UP ou Unified Processus ; en français, nous nommons cette méthode « Processus unifié ». Le terme de processus choisi par les auteurs doit être compris comme « chemin raisonné » de notre définition initiale du concept de méthode.

UP est le fruit du travail de UP de I. Jacobson, G. Booch et J. Rumbaugh . Les auteurs ont réalisé UP à partir de la réunion de leurs travaux antérieurs et de multiples sources telles que les autres méthodes et les retours d'expériences pratiques ; le terme « unifié » qualifie ce travail de réunion ou d'unification qui a sous-tendu UP.

Les 3 auteurs ont publié leur méthode sous le titre : « The Unified Software Development Process » en 1999 ; cette publication a été traduite en français en 2000 sous le titre « Le Processus unifié de développement logiciel [JBR-00].

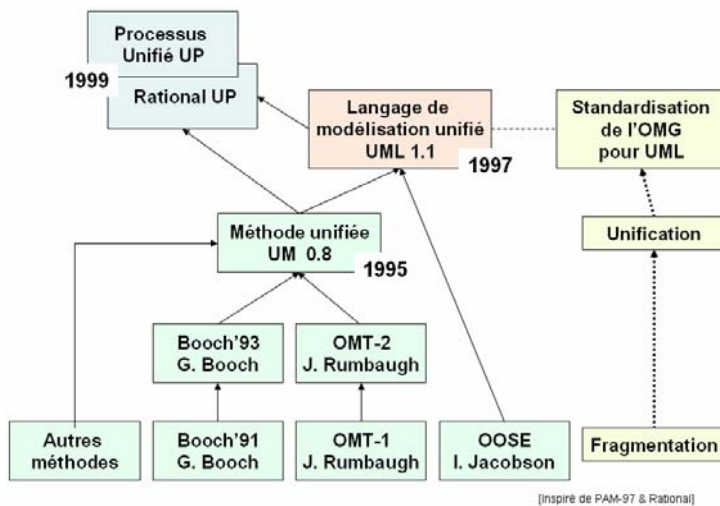


Figure 1 - Genèse du Processus unifié

Dans un premier temps cette méthode, ou processus selon la terminologie des auteurs, s'appelait Unified Method ; une première version de cette méthode a été présentée en 1995 ; mais, par la suite, les auteurs se sont concentrés sur la problématique de l'unification des modèles qui sont à la base du processus ; pour ce faire, ils ont spécifié un langage « unifié » de modélisation de représentation du système

d'information et Unified Method est devenue UML ou Unified Modeling Language en 1997.

Ensuite I. Jacobson, G. Booch et J. Rumbaugh ont repris les éléments du processus UM et du langage UML pour affiner le processus qui est devenu UP en 1999 comme indiqué précédemment.

Parallèlement, aux travaux de Jacobson, Booch & Rumbaugh, la société Rational¹, met au point une méthode commerciale, RUP ou Rational Unified Process, basée sur les mêmes principes que ceux qui ont prévalu à l'élaboration d'UP ; ces travaux ont été menés par divers collaborateurs dont P. Kruchten qui a publié un ouvrage d'introduction au processus sous le titre « The Rational Unified Process : An Introduction » en 1999 ; cette publication a été traduite en français sous le titre « Introduction au Rational Unified Process » [PK-00].

En fait, UP fournit librement, sous forme d'un ouvrage de référence, les principes que chacun peut utiliser et adapter à ses besoins. De son côté, RUP est un produit commercial ; il fournit la méthode et toute une série de guides pour la mettre en œuvre efficacement. Naturellement, d'autres méthodes existent, inspirées ou pas du Processus unifié, UP.

Trois concepts centraux supportent le Processus unifié (UP) :

- **Il est piloté par les cas d'utilisation** ; les cas d'utilisation permettent de **décrire les services attendus** par chacun des futurs utilisateurs du système.
- **Il est centré sur l'architecture** ; l'architecture est **l'ensemble des choix significatifs** de construction réalisés pendant la modélisation. Par analogie avec la construction d'une maison la complétude des modèles ne requiert pas un architecte, il s'agit du travail d'un dessinateur.
- **Il est itératif et incrémental** ; de par la complexité des organisations, des contraintes de temps ou encore de nouveautés technologiques constantes de nombreux risques mettre en péril un développement logiciel ; une démarche itérative permet de réduire dans les premières itérations les risques les plus élevés.

Comme nous venons de le voir, la gestion des risques est un élément majeur du Processus unifié ; un autre en est la modélisation visuelle avec UML.

¹ Les auteurs de UP & UML I. Jacobson, G. Booch et J. Rumbaugh sont parties prenantes de la société Rational. Depuis 2003, Rational est passé dans le giron d'IBM et le processus s'appelle RUP ou IBM Rational Unified Process.

2 UP est piloté par les cas d'utilisation

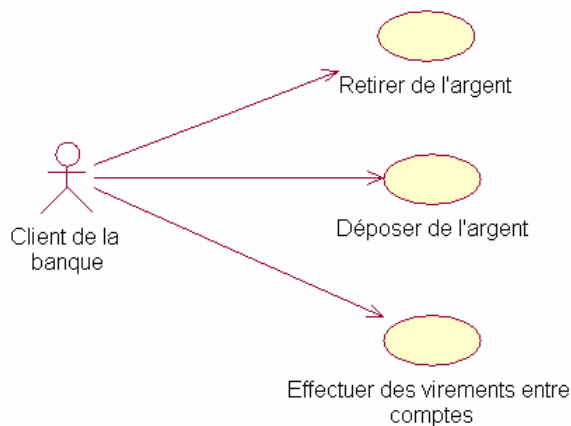


Figure 2 - Extrait de modèle de cas d'utilisation

[JBR-00 p17] *L'objectif d'un système logiciel est de rendre service à ses utilisateurs. Pour réussir la mise au point d'un système, il importe, par conséquent, de bien **comprendre les désirs et les besoins de ses futurs utilisateurs.***

*Un cas d'utilisation est une **fonctionnalité du système produisant un résultat satisfaisant pour l'utilisateur.***

*Les cas d'utilisation saisissent les besoins fonctionnels et **leur ensemble forme le modèle des cas d'utilisation qui décrit les fonctionnalités complètes du système.***

[JBR-00 p47] *L'objectif du Processus unifié est de guider les développeurs vers l'implémentation et le déploiement efficaces de systèmes répondant aux **besoins des clients.** Cette efficacité se mesure en termes de coûts, de qualité et de délai de fabrication. Le passage de l'estimation des besoins du client à leur implémentation est loin d'être naturel. D'abord parce que **les besoins des clients ne se laissent pas si facilement appréhender.** Il faut disposer d'un moyen de les communiquer de façon claire à toute personne impliquée dans le projet. Il faut, ensuite, être en mesure de concevoir une implémentation opérationnelle répondant à ces besoins. Il faut, enfin, vérifier la pleine satisfaction de ces besoins en testant le système.*

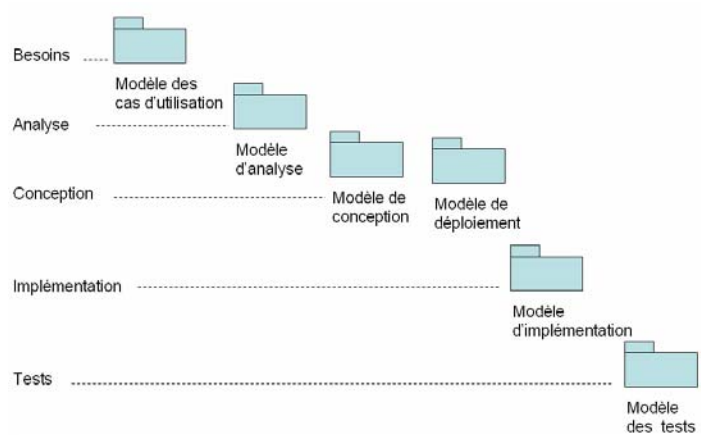


Figure 3 - Enchaînement d'activités et de modèles depuis l'expression des besoins par les cas d'utilisation aux tests du logiciel en passant par l'analyse et la conception

3 UP est centré sur l'architecture

[JBR-00 p18] *Le rôle de l'architecture logicielle est comparable à celle que joue l'architecte dans la construction d'un bâtiment. Le bâtiment est envisagé de différents points de vue: structure, services, conduite de chauffage, plomberie, etc. Ce regard multiple dessine une image complète du bâtiment avant le début de la construction. De la même façon, l'architecture d'un système logiciel peut être décrite comme les différentes vues du système qui doit être construit.*

[JBR-00 p71] *Les cas d'utilisation ne suffisent pas. Pour échafauder un système opérationnel, il faut quelque chose de plus. Ce "plus" c'est l'architecture. On peut se représenter l'architecture d'un système comme la vision commune sur laquelle doivent s'accorder tous les travailleurs (c'est-à-dire les développeurs et les autres intervenants), ou qu'ils doivent au moins accepter. L'architecture offre une perspective claire de tout le système, indispensable pour en maîtriser le développement.*

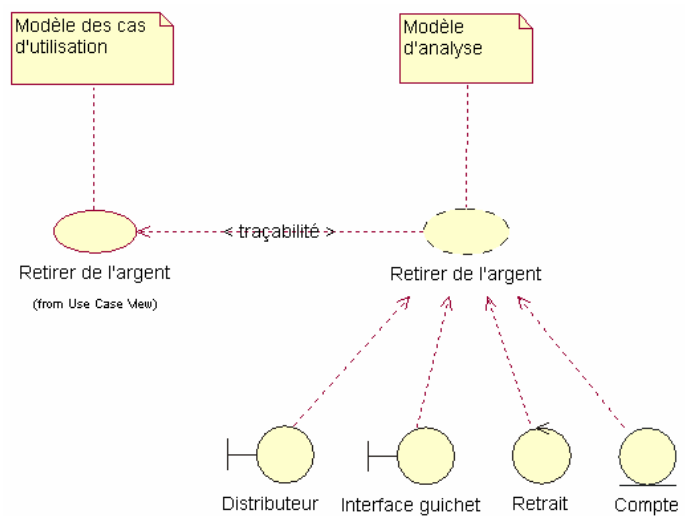


Figure 4 - Vue architecturale des classes d'analyse du cas d'utilisation "Retirer de l'argent"



[JBR-00 p81] *Les cas d'utilisation orientent le développement de l'architecture, tandis que l'architecture guide la réalisation des cas d'utilisation*

4 UP est itératif et incrémental

[JBR-00 p19] *Le développement d'un produit logiciel destiné à la commercialisation est une vaste opération qui peut s'étendre sur plusieurs mois, voire sur une année ou plus. Il n'est pas inutile de découper le travail en plusieurs parties qui sont autant de mini-projets (Concept systémique de système et sous-systèmes). Chacun d'eux représente une itération qui donne lieu à un incrément. Les itérations désignent des étapes de l'enchaînement d'activités, tandis que les incréments correspondent à des stades de développement du produit.*

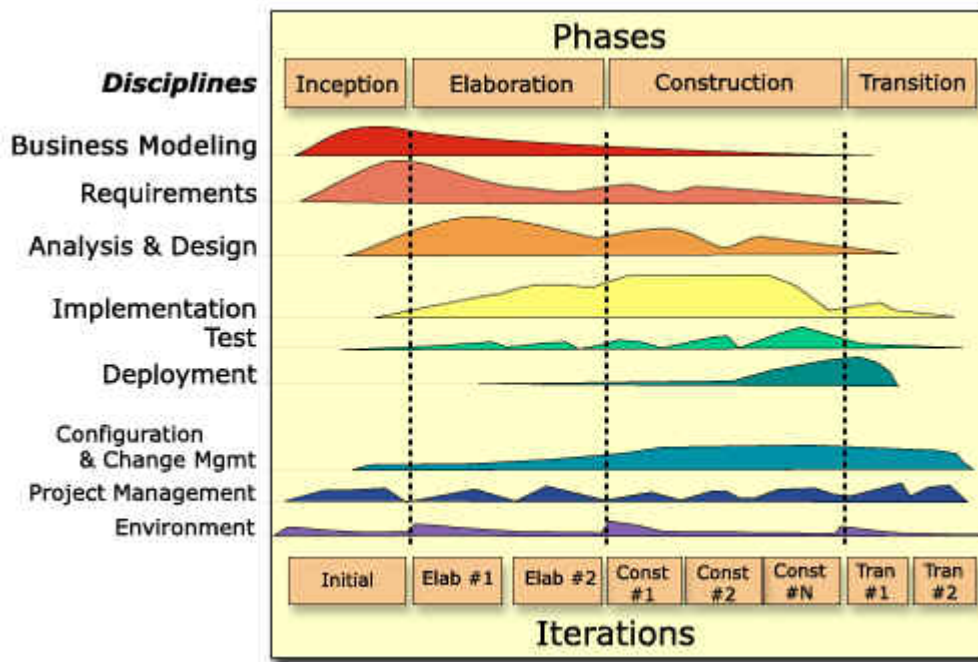


Figure 5 - Les itérations du RUP

[JBR-00 p101] *Parvenir à un juste équilibre entre cas d'utilisation et architecture revient, grosso modo, à harmoniser forme et fond dans le développement d'un produit... Savoir ce qui vient en premier nous ramène au problème de l'œuf et de la poule... Ce sont d'interminables itérations, survenues tout au cours du long processus d'évolution qui ont donné naissance à la poule et à l'œuf. De la même façon, au fil du processus de développement logiciel, les développeurs recherchent consciencieusement cet équilibre (entre cas d'utilisation et architecture) à travers une série d'itérations. L'approche itérative et incrémentale du développement constitue bien, par conséquent, le troisième pivot du Processus unifié.*

5 Modélisation

5.1 Démarche

Les modèles sont au cœur du processus ; comme nous l'avons indiqué précédemment, les choix essentiels de modélisation définissent l'architecture du futur logiciel et donc le résultat final fournit aux futurs utilisateurs.

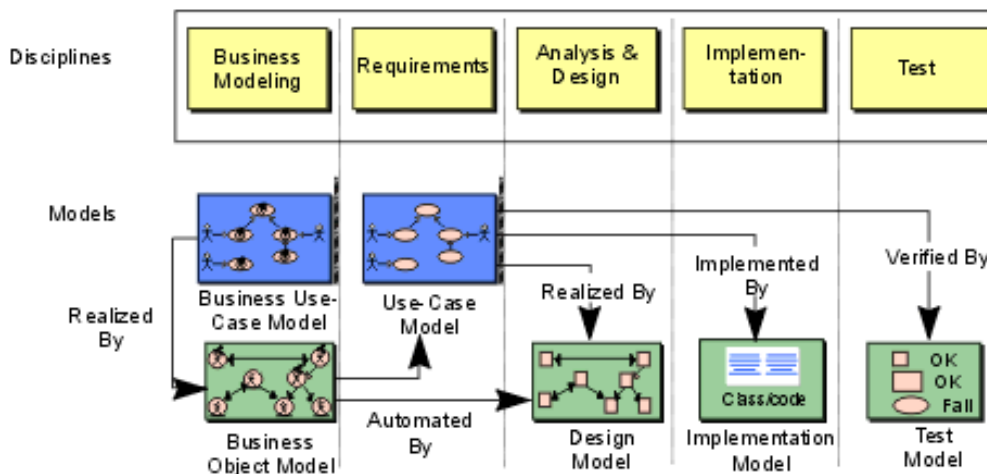


Figure 6 - Les enchaînements et dépendances de modèles du RUP

Dans un premier temps, la modélisation du métier « Business Modeling » permet de représenter les activités de l'organisme dans son ensemble sous forme de modèles des cas d'utilisation métier, de modèles d'objets métier et de modèles du domaine ; les concepts du modèle du domaine s'apparentent à ceux du modèle entités-associations de l'approche classique.

Ensuite ces modèles servent de données d'entrée pour la modélisation des exigences « Requirements » sous forme de modèle de cas d'utilisation système ; il s'agit cette fois de cas d'utilisation du sous-système d'information de l'organisme.

Le modèle de cas d'utilisation système, le modèle d'objets métier et le modèle du domaine sont utilisés comme données d'entrée pour l'analyse ; la modélisation en analyse fournira le modèle d'analyse et ainsi de suite jusqu'aux tests finaux.

5.2 Langage de modélisation unifié UML



Un apport qui nous semble essentiel de la démarche d'unification des auteurs du Processus unifié, c'est d'avoir spécifié UML, un langage de modélisation tout aussi unifié² ; comme nous pouvons le voir dans la figure 1, la version 1.1 d'UML a été standardisée par l'OMG³ en fin 1997. Après une version 1.5, qui a été largement utilisée ; fin 2004, l'OMG a libéré les spécifications d'une version 2.0 qui fait apparemment la part belle à la modélisation des activités de l'entreprise et plus particulièrement au concept d'entités de données.

Grâce à une large utilisation d'UML, à la limite quelle que soit la méthode sous-jacente, les erreurs dues à une interprétation erronée d'un modèle ou d'un diagramme par méconnaissance d'un langage particulier se trouvent grandement réduite ; dans l'approche classique, pratiquement chaque méthode ou école de pensée a défini son langage de représentation, ce qui rend le passage d'une méthode ou d'un outil à un autre relativement périlleux ou, en tout les cas, rébarbatif.

De plus, l'unification et la standardisation d'UML permet l'échange⁴ de modèles entre outils logiciels différents ; en conséquence le changement d'outils logiciel en est facilité.

Le même langage s'applique aux multiples modèles nécessaires à représenter les divers aspects des systèmes d'information.

5.3 Exemple

L'activité de modélisation étant essentielle à la pratique efficace des méthodes de développement de systèmes d'information informatisés, nous produisons ci-après, à titre d'illustration, deux diagrammes issus du modèle d'analyse pour le premier et du modèle de conception pour le deuxième. Les deux diagrammes, respectivement les deux modèles, affinent le cas d'utilisation « Retirer de l'argent » de la Figure 2.

Ces deux modèles ont été réalisés avec l'atelier de génie logiciel Rose du constructeur IBM⁵ à partir d'un cas pratique issu de [JBR-00].

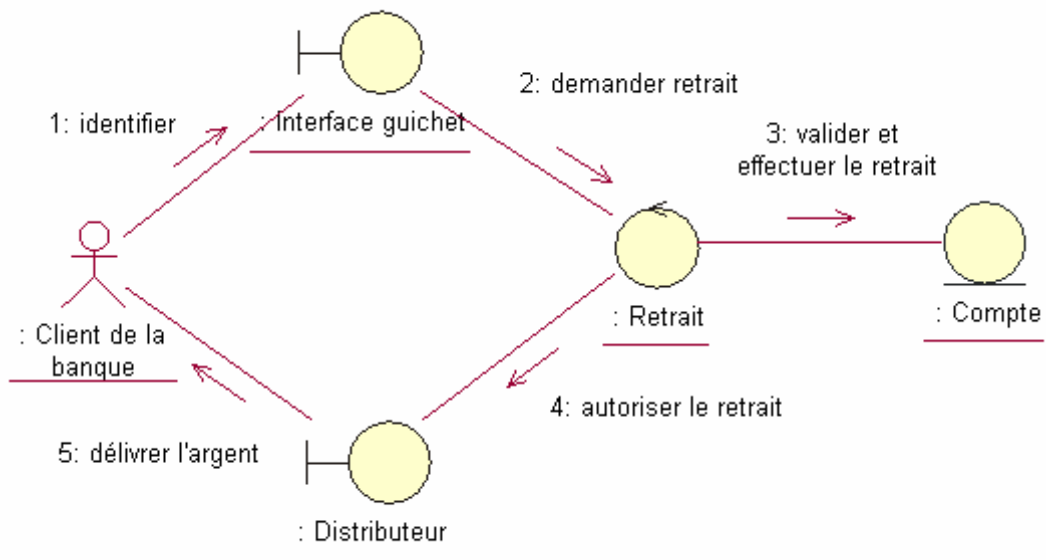
² UML – Unified Modeling Language

³ Site UML de l'OMG-Object Management Group : <http://www.uml.org>

⁴ L'échange de contenu de référentiel se fait grâce à XMI, un schéma XML défini par l'OMG ; <http://www.omg.org/technology/xml/index.htm>

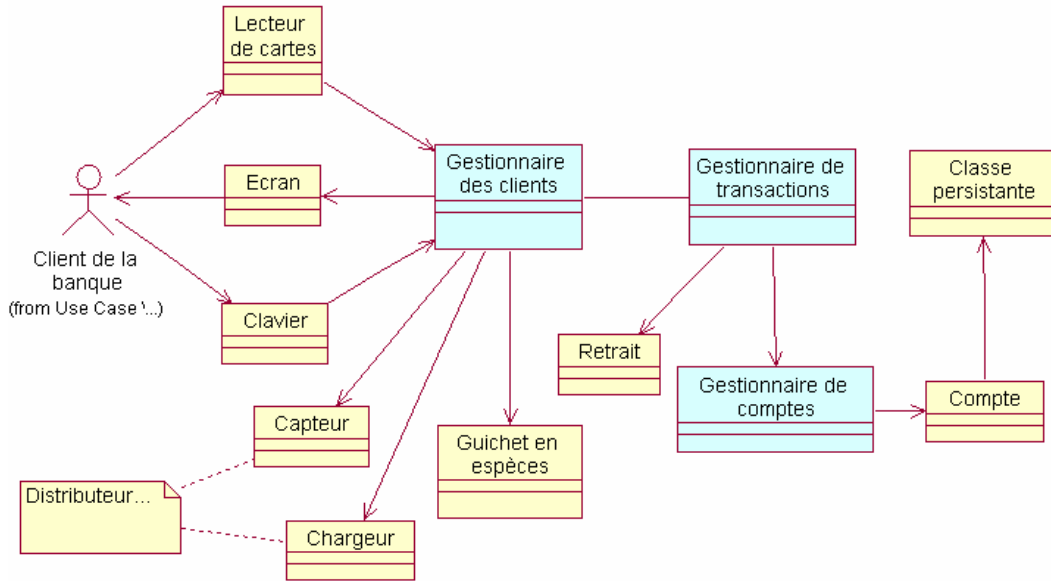
⁵ Anciennement Rational

5.3.1 Diagramme de collaboration en analyse



Nous voyons sur le diagramme ci-dessus, les classes d'analyse qui réalisent le cas d'utilisation « Retirer de l'argent ». Nous pouvons observer l'acteur « :Client de la banque » qui interagit avec les classes d'interfaçage du système ; à leur tour ces classes interagissent avec les classes de contrôle et de données (entités) pour réaliser le service attendu.

5.3.2 Diagramme de classes de conception



Nous voyons sur le diagramme ci-dessus, les classes de conception qui réalisent le cas d'utilisation « Retirer de l'argent ». Comme précédemment, nous pouvons observer l'acteur « :Client de la banque » qui interagit avec le système ; les classes de conception réalisent, en détail, les spécifications abstraites des classes d'analyse comme illustré par le schéma de dépendance ci-dessous.

