Le Rational Unified Process®

Philippe Kruchten,
Rational Software Canada
Janvier 1999

Note : Ce texte est extrait d u livre
Philippe Kruchten, *Introduction au Rational Unified Process,* Editions Eyrolles, 1999.

Ce document présente le Rational Unified Process®. Il décrit la structure du processus, ses principales caractéristiques et le produit commercial disponible qui permet de le mettre en œuvre.

Qu'est-ce que le Rational Unified Process?

Le Rational Unified Process (RUP®) est un *processus de développement de logiciel*. Il permet d'affecter selon une approche disciplinée à l'affectation des tâches et des responsabilités au sein d'une organisation de développement. Son but est d'assurer la production d'un logiciel de grande qualité satisfaisant les besoins de ses utilisateurs finaux dans des délais et avec un budget prévisibles.
Le Rational Unified Process est un *processus commercial*. C'est un produit développé et maintenu par Rational Software qui est parfaitement intégré à l'ensemble des outils de développement logiciel proposés par la société. Il est disponible sur CD-R ou via l'Internet. Le livre d'introduction fait partie intégrante du Rational Unified Process, mais n'en présente qu'une petite fraction, soit environ 8 %. La structure physique du produit est décrite plus loin dans ce chapitre.
Le Rational Unified Process est aussi un *framework* de processus que l'on peut adapter et étendre pour tenir compte des besoins de l'organisation qui l'adopte. Nous reviendrons plus loin dans ce chapitre sur la structure logique de ce framework
Le Rational Unified Process intègre un grand nombre des meilleures pratiques de développement du logiciel moderne sous une forme adaptée à un large éventail de projets et d'organisations.

La structure du produit commercial RUP®

Un grand nombre d'organisations ont peu à peu réalisé l'importance d'un processus de développement de logiciel bien défini et bien documenté pour assurer le succès de leurs projets logiciels. Au cours des années, elles ont rassemblé des connaissances et les ont partagées avec leurs développeurs. Ce savoir-faire collectif prend ses sources dans des méthodes, des manuels publiés,

des cours de formation et des petites notes techniques accumulés au cours des projets. Malheureusement, ces informations se retrouvent souvent dans de beaux classeurs sur les étagères des développeurs, rarement mis à jour, elles sont rapidement périmées et presque jamais utilisées.
En revanche, le Rational Unified Process est un produit commercial qui est maintenu comme n'importe quel produit ou outil logiciel. Voici quelques-unes des atouts clés du Rational Unified Process, en tant que produit :

- Des mises à jour régulières sont publiées par Rational Software.
- Il utilise la technologie du web, de sorte qu'il est littéralement à portée de souris des développeurs.
- On peut aisément l'adapter pour accommoder les besoins particuliers d'une organisation.
- Il est intégré à un grand nombre d'outils de développement logiciel de Rational, de sorte que les développeurs peuvent accéder aux recommandations données par le processus à partir de l'outil qu'ils utilisent.

Le produit RUP en ligne a des avantages que ne présente pas un processus disponible uniquement sur papier, sous la forme d'un livre ou d'un classeur :

- tous les membres du projet ont accès à la toute dernière version sur un intranet ;
- un accès instantané à toutes les informations-clés du processus grâce à plusieurs méthodes et outils d'accès : moteur de recherche, index, table des matières dynamique, etc. ;
- la possibilité de naviguer d'une partie du processus à une autre, ou vers une référence externe, comme UML, ou même d'activer directement l'outil correspondant à une tâche ;
- l'intégration aisée des procédures et recommendations propres à un projet, et des améliorations qu'on veut apporter au processus ;
- le contrôle de version du processus et la gestion de variantes propres à un projet ou un département.

Le produit Rational Unified Process comprend :

1. Un CD-ROM qui contient une version en ligne de la description de tous les éléments du Rational Unified Process : toutes les tâches, activités, recommendations, procédures et exemples, sous forme d'un site web que l'usager installe sur sa machine, ou sur un serveur de l'entreprise.
2. Un manuel d'introduction[1], et que l'on retrouve aussi dans le produit en ligne.

La figure 1 montre la structure logique en plusieurs manuels du RUP.

[1] Philippe Kruchten, *Introduction au Rational Unified Process*, Editions Eyrolles, 1999.

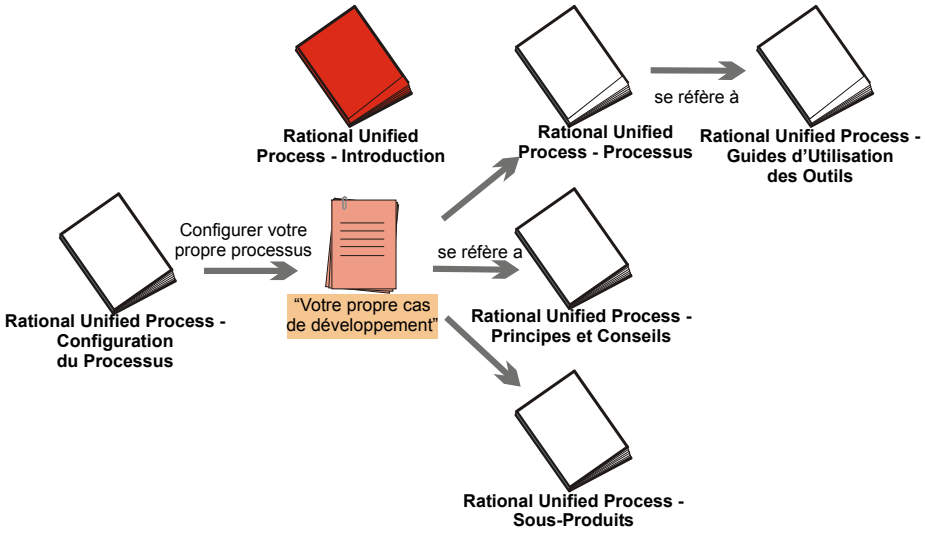

Figure 1 Structure logique du processus

Le produit en ligne contient en outre :

- Des guides d'utilisation d'outils qui établissent des ponts avec les outils de développement de logiciel, et donnent des conseils supplémentaires sur leur usage lorsqu'on suit le Rational Unified Process. Il y a par exmaple des guides pour les outils de de Rational, tel que Rational Rose pour la modélisation graphique ou ClearCase pour la gestion de configuration.
- Des canevas (templates) pour tous les artefacts importants du processus, à savoir canevas pour Rational SoDA™, qui aide à automatiser la documentation du logiciel ; pour RequisitePro™, qui aide à gérer les besoins ; pour Adobe FrameMaker™ and Microsoft Word™, pour la plupart des documents classiques ; pour Microsoft Project™ pour la planification du processus ; et enfin pour Microsoft FrontPage™, pour étendre le RUP lui-même.

Le Rational Unified Process en ligne s'utilise avec navigateur Web, tel que Netscape Navigator™ ou Microsoft Internet Explorer™. Il contient des milliers de liens hypertextes, en particulier à partir des centaines d'images (GIF ou JPEG) interactives qui permettent un accès facile à l'information. On peut également accéder à une information spécifique grâce à des applets en Java qui réalisent une exploration de l'arborescence du contenu, ou un moteur de recherche intégré.

Les deux dimensions du processus

La figure 2 représente l'architecture générale du Rational Unified process, caractérisé par ses deux dimensions :

- l'axe horizontal représente le temps et montre le déroulement du cycle de vie du processus ;
- l'axe vertical représente les principaux enchaînements d'activités, qui regroupent les activités selon leur nature.

La première dimension représente l'aspect dynamique du processus tel qu'il est mis en œuvre et s'exprime en termes de cycles, de phases, d'itérations et de jalons. La seconde dimension représente l'aspect statique du processus ; elle rend compte de la façon dont il est décrit en termes de composants, de processus, d'activités, d'enchaînements, de artefacts et de travailleurs.

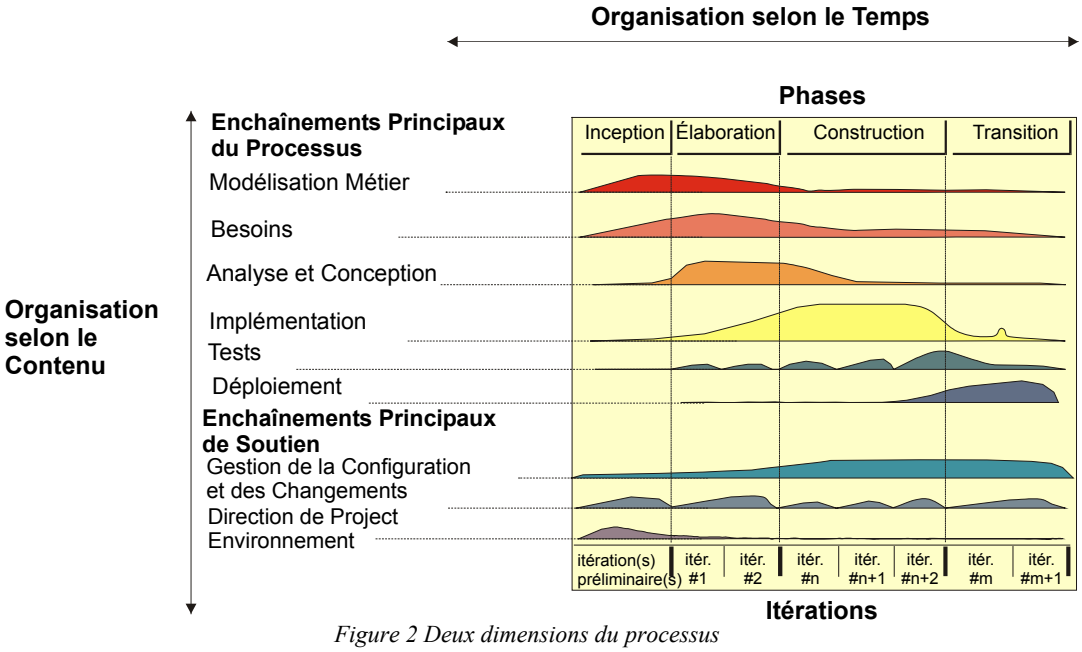

Figure 2 Deux dimensions du processus

Les meilleures pratiques du développement de logiciel

Ces 6 pratiques sont prise en compte dans le Rational Unified Process, parmi des dizaines d'autres.

Développement itératif

L'approche itérative recommandée par le Rational Unified Process est, de façon générale, supérieure à l'approche linéaire en cascade sur plusieurs plans :

- Elle permet de prendre en considération les changements de besoins. Il est rare que le cahier des charges ne subisse aucun changement. Mais un flot constant de demandes de modifications ont toujours engendré des problèmes de livraisons tardives, de délais non-respectés, de clients mécontents et de développeurs frustrés.
- Le Rational Unified process ne préconise pas une intégration en "big bang" à la fin du cycle de développement, mais au contraire d'intégrer les éléments au fur et à mesure. Cette approche itérative est presque un processus d'intégration continue. Dans un cycle de vie en cascade, l'intégration est une période assez pénible et pleine d'aléas, représentant jusqu'à 40% de l'effort total à la fin du projet ; dans le RUP, on décompose cette opération en six à neuf plus petites intégrations qui démarrent chacune avec un nombre réduit d'éléments à intégrer.

- L'approche itérative permet de contrôler les risques plus tôt parce que c'est généralement au moment de l'intégration que l'on découvre les risques et que l'on s'en occupe. Lorsque l'on effectue les premières itérations, on passe par toutes les activités du processus, mettant en œuvre beaucoup d'aspects du projet, tels que les outils, les logiciels acquis, les compétences du personnel, et ainsi de suite. Ce que l'on pensait être des risques s'avéreront ne pas en être et l'on va en révéler de nouveaux dont on ne soupçonnait pas l'existence.
- L'approche itérative donne aux responsables du développement la possibilité d'opérer des changements tactiques à la définition du projet, par exemple, pour rester compétitif avec d'autres produits. On peut plus aisément décider de livrer une version précoce réduite en fonctionnalités pour contrecarrer l'action d'un concurrent, ou changer de fournisseur pour une technologie donnée.
- L'approche itérative facilite la réutilisation parce qu'il est plus aisé d'identifier les parties communes alors qu'elles sont partiellement conçues et implémentées, plutôt que de les identifier toutes ex nihilo. Il est difficile d'identifier et de développer des composants réutilisables, mais les revues de conception lors des premières itérations permettent aux architectes de découvrir des composants potentiellement réutilisables auxquels personne n'avait songés, puis de les développer et d'affiner le code commun dans les itérations suivantes.
- Avec une approache itérative, on obtient une architecture plus robuste parce qu'elle permet de corriger les erreurs au cours de plusieurs itérations. Les anomalies sont détectées pendant les premières itérations lors de l'inception à l'élaboration, lorsqu'on a encore suffisamment de temps pour les cooriger, au lieu de l'être au cours d'une massive campagne de tests à la fin du projet. On découvre les problèmes de performance alors que l'on peut encore s'en occuper, ce qui évite la panique à la veille d'une livraison.
- Les développeurs peuvent apprendre tout au long du projet, et leurs diverses capacités et spécialités sont mieux employées pendant toute la durée du cycle de vie. Les testeurs commencent à tester plus tôt, les rédacteurs techniques commencent à rédiger plus tôt, et ainsi de suite. Dans un développement en cascade, ces mêmes personnes doivent attendre la fin des phases initiales pour pouvoir commencer leur travail, rédigeant plan après plan, mais ne faisant aucun progrès réel. Qu'est-ce qu'un testeur peut tester, lorsque le produit consiste en une pile de documentation de conception longue d'un mètre sur une étagère ? Les besoins en formation et en personnel supplémentaire sont repérés plus tôt, lors de l'évaluation des premières itéerations.
- Le processus de développement lui-même peut être amélioré et affiné en cours de route. L'évaluation à la fin d'une itération s'intéresse à l'état du projet du point de vue du produit et du planning, mais analyse aussi ce que l'on doit modifier dans l'organisation et dans le processus pour le rendre plus performant pendant l'itération suivante.

Les chefs de projet s'opposent souvent à l'idée de l'approche itérative, qu'ils considèrent comme un bricolage sans fin et incontrôlé. Dans le Rational Unified Process, cette approche est tout à fait contrôlée ; les itérations sont planifiées quant à leur nombre, leur durée et leurs objectifs. Les tâches et les

responsabilités des participants sont bien définies. La progression des mesures objectives est suivie. On fait certaines reprises de code d'une itération à l'autre, qui sont aussi contrôlées avec soin.

Gestion des besoins

La gestion des besoins est une approche systématique visant à déterminer, organiser, communiquer et gérer un cahier des charges toujours changeant d'un système informatique ou d'une application logicielle.
Les bénéfices d'une gestion efficace des besoins adéquate sont multiples :

- *Un meilleur contrôle des projets complexes*
  Le manque de compréhension du comportement souhaité du système, et la porte ouverte à des changements d'exigences constants sont deux des facteurs principaux des échecs de projets informatique.
- *Une qualité de logiciel améliorée et la satisfaction du client*
  Pour mesurer la qualité d'un système, en d'autres termes pour déterminer s'il fait ce qu'il est supposé faire, il faut que tous les intervenants[2] partagent la même idée de ce que l'on doit construire et tester.
- *Des coûts et des délais réduits pour le projet*
  Corriger les erreurs de cahier des charges coûte très cher. En conséquence, en éliminant ces erreurs tôt dans le cycle de développement, on réduit les coûts du projet et on évite de prendre du retard.
- *Une meilleure communication à l'intérieur de l'équipe*
  La gestion des exigences permet d'impliquer des utilisateurs tôt dans le développement, pour s'assurer que l'application va satisfaire leurs vrais besoins. Un cahier des charges bien géré est la clé d'un consensus entre différents intervenants : utilisateurs, clients, gestionnaires, concepteurs et testeurs, sur ce que le système doit faire et sur leurs engagements respectifs.

Architecture et utilisation de composants

Les cas d'utilisation sont un fil conducteur du Rational Unified Process pendant tout le cycle de vie, mais les activités de conception sont centrées sur la notion d'*architectur*e, qu'il s'agisse de l'architecture du système, ou, pour les systèmes à forte composante logicielle, de l'architecture du logiciel. Le but principal des premières itérations du processus est de produire et de valider une architecture de logiciel qui, dans le cycle de développement initial, prend la forme d'un prototype architectural exécutable qui évoluera graduellement pour devenir la version finale du système dans les dernières itérations.
Avec le Rational Unified process, on dispose d'un moyen méthodique et systématique pour concevoir, développer et valider une architecture. Il préconise de décrire une architecture au moyen de vues architecturales multiples. Il suggère d'établir un style architectural, avec des règles de conception et des contraintes. Dans enchaînement des activités de conception figurent des activités spécifiques pour identifier les contraintes et les éléments architecturaux significatifs, ainsi que les principes qui vont guider leur choix. La planification

[2] Un intervenant est une personne ou un représentant d'une organisation qui a un intérêt dans l'issue du projet, à savoir un utilisateur final, un acheteur, un sous-traitant, un développeur, un responsable de projet, ou toute autre personne qui s'intéresse suffisamment au projet ou dont les besoins doivent être satisfaits par ledit projet.

des premières itérations prend en compte la conception d'une architecture et la résolution des principaux risques techniques associés.
Un *composant* logiciel est un morceau significatif de logiciel, un module, un paquetage, ou un sous-système qui remplit une fonction bien précise, dont les frontières sont bien délimitées et que l'on peut intégrer dans une architecture bien définie. C'est la réalisation physique d'une abstraction de la conception. Le développement à base de composants peut prendre divers aspects : développement de composants, réutilisation de composants, et famille de composants standardisés.

- Définir une architecture modulaire suppose d'identifier, d'isoler, de concevoir, de développer et de tester individuellement des composants bien construits. On les intègre ensuite progressivement pour former le système dans son ensemble.
- De plus, certains de ces composants peuvent être développés de façon à être réutilisable, en particulier s'ils fournissent des solutions à un grand éventail de problèmes. Ces composants réutilisables sont typiquement plus ambitieux que de simples collections d'utilitaires ou des bibliothèques de classes. Il constituent une base réutilisable, augmentant la qualité du logiciel et la productivité de l'organisation de développement.
- Plus récemment, l'avènement et le succès d'infrastructures commerciales qui soutiennent le concept de composant logiciel—tel que CORBA (Common Object Request Broker Architecture), l'Internet, ActiveX et les JavaBeans—a mis sur orbite toute une industrie de composants disponibles, tout prêts, dans des domaines variés, permettant aux développeurs d'acheter et d'intégrer des composants plutôt que de les développer en interne.

Le premier aspect exploite les vieux concepts classiques de modularité et d'encapsulation, sur lesquels repose la technologie orientée objet, mais à une plus grande échelle que celle de la classe. Les deux derniers aspects montrent que le développement est en train d'évoluer de la programmation du logiciel (une ligne de code à la fois), vers la composition du logiciel (assemblage de composants).

Le Rational Unified Process encourage le développement à base de composants de plusieurs manières.

- L'approche itérative permet aux développeurs d'identifier progressivement les composants et de décider lesquels seront développés, en interne ou en externe, réutilisés ou achetés.
- L'architecture du logiciel permet d'articuler la structure. Elle énumère les composants, décrit la façon de les intégrer, ainsi que les mécanismes et les patterns fondamentaux selon lesquels ils interagissent.
- On utilise les concepts de paquetages, de sous-systèmes et de couches pendant l'analyse et la conception pour organiser les composants et spécifier les interfaces.
- L'activité de test porte d'abord les composants simples, puis progressivement sur de plus grands ensembles de composants intégrés.

La modélisation et UML

Une grande partie du RUP traite du développement et de la gestion de *modèles* du système en cours de dévelopement. Les modèles aident à comprendre la

réalité et à donner une représentation à la fois au problème et à sa solution. Un modèle est une simplification de la réalité que l'on construit pour maîtriser un grand système complexe difficile à appréhender dans sa totalité. UML (Unified Modeling Language) est un langage graphique qui permet de visualiser, de spécifier, de construire et de documenter les artefacts d'un système logiciel. UML fournit un moyen normalisé pour décrire les plans d'un système, à travers des éléments conceptuels tels que les processus métier et les fonctions du système, et d'éléments concrets tels que les classes écrites dans un langage de programmation particulier, les schémas de base de données et les composants logiciels réutilisables.[3]
UML est le langage commun pour exprimer les différents modèles, mais il n'explique pas comment développer du logiciel. Il fournit le vocabulaire, mais n'explique pas comment écrire le livre. C'est pourquoi Rational a développé le Rational Unified Process et UML en parallèle, l'un complétant l'autre. Le Rational Unified Process est un guide pour une utilisation efficace de UML : il précise de quel modèle on a besoin, pour quelles raisons on doit l'utiliser et comment le construire.

Qualité du processus et du produit

Pourquoi n'y a-t-il pas de travailleur chargé de la qualité dans le Rational Unified process ? La réponse à cette question courante est la suivante : la qualité n'est pas ajoutée au produit par l'intervention d'un petit nombre de personnes. Au contraire, la qualité est la responsabilité de tous les membres de l'organisation de développement. Lorsque l'on développe du logiciel, il y a deux aspects interdépendants de la qualité auxquels on s'intéresse : la qualité du produit final et la qualité du processus utilisé pour le construire.

- La qualité du produit principal en cours de production (le logiciel ou le système) et de tous les éléments qu'il comprend (par exemple, les composants, les sous-systèmes, l'architecture, etc.) .
- La qualité du processus, c'est-à-dire le degré avec lequel un processus de développement acceptable est mis en place et suivi pendant la fabrication du produit, y compris les mesures et les critères de qualité. De plus, la qualité du processus inclut la qualité de artefacts intermédiaires, tels que les plans d'itération, les plans de test, les réalisations de cas d'utilisation, les modèle de conception, etc. qui sont fabriqués pour soutenir le produit principal.

Le Rational Unified Process cherche à vérifier et à évaluer objectivement si le produit satisfait ou non le niveau attendu de qualité.

Gestion de la configuration et des changements

Tous les artefacts d'un développement de logiciel sont modifiés souvent, et encore plus souvent durant un développement itératif. En permettant une certaine souplesse dans la planification et l'exécution du développement, et en permettant au cahier des charges d'évoluer, le développement itératif met l'accent sur le problème crucial de la gestions des demandes de changements, du suivi des modifications et de la synchronisation de toutes les parties concernées. La gestion des changements est une façon systématique pour l'équipe de

[3] Grady Booch et al., *UML Users Guide*. Reading, MA: Addison Wesley Longman, 1998.

développement de gérer les modifications dans exigences, de la conception ou d'implémentation. Cela implique de garder une trace explicite de tous les défauts, les malentendus et les engagements du projet, et de déterminer les artefacts et versions du produit affectés par les modifications. Cette gestion des changements est par conséquent étroitement liée à la gestion de configuration et on peut en dériver des mesures quant à l'avancement du projet et la qualité du produit et du processus.

Autres caractéristiques importantes du Rational Unified Process

Le Rational Unified Process se distingue aussi par les trois caractéristiques suivantes :

- le rôle qu'y jouent les cas d'utilisation (" use cases ") qui sont le point de départ de bien des aspects du développement ;
- son utilisation comme processus générique ou «framework» de processus que l'on peut peut adapter et étendre ;
- le soutien d'outils de développement du logiciel pour effectuer certains aspects du processus.

Un développement dirigé par les cas d'utilisation

Il est souvent difficile de dire comment fonctionne un système orienté objet simplement à l'examen d'un modèle objet. Cette difficulté provient de l'absence d'un fil conducteur visible et cohérent qui passe à travers le tous les éléments du système lorsqu'on effectue certaines tâches. Dans le Rational Unified Process, les *cas d'utilisation* fournissent ce fil conducteur en définissant le comportement attendu du système.

Les cas d'utilisation ne sont pas indispensables à une conception objet, mais ils fournissent un lien important entre le cahier des charges du système et d'autres artefacts de développement, tels que la conception et les tests. D'autres méthodes orientées objet suggèrent des approches semblables aux cas d'utilisation, mais avec des noms différents, tels que scénarios ou "threads".

Le Rational Unified Process est une approche *pilotée par les cas d'utilisation*, ce qui signifie que les cas d'utilisation constituent un point de départ pour le reste du processus de développement. Ils jouent un rôle prépondérant dans plusieurs des enchaînements d'activités du processus, en particulier pour les exigences, la conception, les tests et la gestion de projet. Les cas d'utilisation sont aussi la clé de la modélisation métier.

La configuration du processus

Le Rational Unified Process est général et suffisamment détaillé pour être utilisé tel que livré, par des organisations de développement de logiciel de petite ou de moyenne taille, en particulier par celles dont la culture de processus n'est pas très établie. Mais le Rational Unified Process est aussi un framework de processus : l'organisation qui l'adopte peut le modifier, l'adapter et l'étendre pour l'harmoniser à ses besoins particuliers, à ses caractéristiques, ses contraintes propres, son histoire, sa culture d'entreprise et son domaine d'action.

Il ne faut pas suivre un processus les yeux fermés, au rique de faire du travail inutile et de générer des artefacts de peu de valeur ajoutée. Le processus doit être le plus sobre possible tout en remplissant quand même sa mission, qui est de produire rapidement et de façon prévisible du logiciel de grande qualité.
L'organisation qui l'adopte le complète en y ajoutant ses propres «meilleures pratiques», ainsi que ses règles et procédures de travail spécifique.
Les éléments du processus qui sont les plus à même d'être modifiés, adaptés, ajoutés ou supprimés sont les artefacts, les activités, les travailleurs et les enchaînements d'activités, mais surtout les principes et conseils, et les canevas d'artefacts.

Outils de support

Pour être efficace, un processus doit être soutenu par des outils adéquats. Le Rational Unified Process est associé à une large palette d'outils qui automatisent les étapes de plusieurs d'activités. Ces outils permettent de créer et mettre à jour les différents artefacts du processus, et supportent les activités de modélisation graphique, de programmation et de tests. Les outils sont très précieux pour effectuer les tâches fastidieuses associées à la gestion des changements et la gestion de configuration pour chaque itération

Un bref historique du Rational Unified Process

Le Rational Unified Process a mûri au cours des années et est le fruit de l'expérience collective des nombreuses personnes et entreprises qui ont participé à l'aventure de Rational Software. La figure 4 dresse l'arbre généalogique du processus du RUP.
Le Rational Unified Process est le successeur direct du Rational Objectory Process (version 4.1). Le Rational Unified Process contient plus d'informations dans les domaines de l'ingénierie des données, de la modélisation du métier, de la gestion de projet et de la gestion de configuration, résultat de la fusion avec Pure-Atria. Il intègre également de façon plus étroite l'ensemble des outils de Rational Software.
Le Rational Objectory Process était le résultat d'une intégration entre "L'approche Rational" et l'Objectory Process (version 3.8) après l'acquisition, en 1995, de la société suédoise Objectory AB par Rational Software. D'Objectory, le RUP a hérité du modèle de processus et du concept central des *cas d'utilisation*. De son passé Rational, il a acquis la stratégie de développement itératif et le souci de l'architecture. La version 4.1 intégrait également des éléments concernant la gestion des besoins hérité de Requisite, Inc., et un processus de test détaillé hérité de SQA, Inc., compagnies qui ont aussi fusionné avec Rational Software. Finalement, cette version du processus était la première à utiliser le langage UML nouvellement créé (UML 0.8).
Objectory, créé en Suède en 1987 par Ivar Jacobson, est le résultat d'une longue expérience acquise chez Ericsson. Ce processus devint un produit de sa compagnie, Objectory AB. Centré autour du cas d'utilisation et d'une méthode de développement orientée objet, il a rapidement acquis une grande réputation au sein de l'industrie du logiciel et a été adopté et intégré par de nombreuses

compagnies à travers le monde. Une version simplifiée d'Objectory a été publiée dans un manuel en 1992.[4]
Le Rational Unified Process est une instance spécifique et très détaillée d'un processus plus générique décrit par Ivar Jacobson, Grady Booch et James Rumbaugh dans le manuel pédagogique *The Unified Software Development Process*.[5]

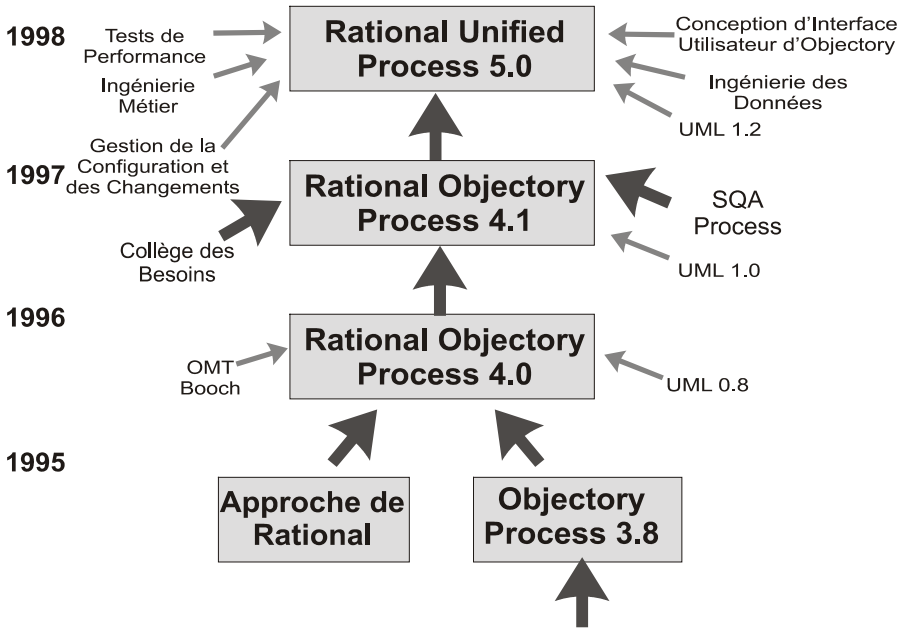

Figure 4 Généalogie du Rational Unified Process

Structure statique du RUP

Cette section décrit comment le Rational Unified Process est représenté. Il décrit les concepts clés de travailleur, d'activité, de artefact et d'enchaînement de travaux, ainsi que d'autres éléments utilisés dans la description du processus.

Un processus décrit *qui* fait *quoi*, *comment* et *quand*. Le Rational Unified Process est représenté en utilisant quatre éléments primaires de modélisation.

- Les travailleurs: le *qui*
- Les activités: le *comment*
- Les artefacts: le *quoi*
- Les enchaînements: le *quand*

Ces éléments sont représentés à la figure 5.

[4] Ivar Jacobson et al., *Object-Oriented Software Engineering-A use Case-Driven Approach*. Reading, MA: Addison-Wesley, 1992.
[5] Ivar Jacobson, Grady Booch et James Rumbaugh, *The Unified Software Development Process*. Reading, MA: Addison Wesley Longman, 1998.

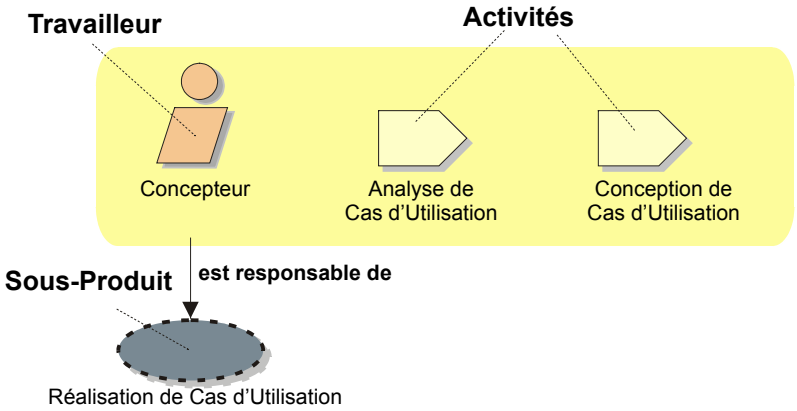

Figure 5 Travailleur, activités et artefacts

Le terme *travailleur* définit le comportement et les responsabilités d'un individu ou d'un groupe d'individus travaillant en équipe. Le comportement s'exprime en termes d'*activités* que le travailleur accomplit, et chaque travailleur est associé à un ensemble d'activités cohésives. Dans ce contexte, « cohésives » signifie qu'il vaut mieux que ces activités soient accomplies par une même personne. Les responsabilités de chaque travailleur s'expriment, en général, en relation avec certains *artefacts* que le travailleur crée, modifie ou contrôle.

Travailleurs

Pour fixer les idées, on peut penser à un travailleur comme étant une « casquette » qu'un individu peut porter au cours d'un projet. La même personne peut porter plusieurs casquettes différentes. La distinction est importante parce que l'on pense naturellement à un travailleur comme étant l'individu ou l'équipe, mais dans le Rational Unified Process, le terme *travailleur* fait référence aux rôles qui définissent comment un individu devrait travailler. Un travailleur remplit un ou plusieurs rôles et est le propriétaire d'un ensemble de artefacts. On peut aussi penser à un travailleur comme étant un personnage dans une pièce—un personnage dont le rôle pourrait être joué par plusieurs acteurs différents. Voici des exemples de travailleurs.

- *Analyste système*
  Un individu qui agit en tant qu'analyste système dirige et coordonne la clarification des besoins et la modélisation par les cas d'utilisation, en mettant en valeur les fonctionnalités du système et en lui fixant des contours.
- *Concepteur*
  Un individu qui agit en tant que concepteur définit les responsabilités, les opérations, les attributs et les relations de dépendance d'une ou de plusieurs classes et détermine comment elles s'insèrent dans l'environnement d'implémentation.
- *Concepteur de test*
  Un individu qui agit en tant que concepteur de test est responsable de la planification, de la conception, de l'implémentation et de l'évaluation de tests, y compris la création d'un plan de test et d'un modèle de test, l'implémentation des procédure de test et enfin l'évaluation de la couverture, des résultats et de l'efficacité des tests.

On notera que les travailleurs ne sont pas des individus; ils décrivent plutôt comment des individus devraient se comporter dans le travail ainsi que les responsabilités de chacun. Les individus membres de l'organisation de développement du logiciel portent différentes casquettes ou jouent différents personnages ou rôles.[6] La correspondance entre travailleurs et individus est faite par le chef de projet lors de la planification et de l'allocation du projet en personnel. Cette distribution permet à un individu d'agir en tant que plusieurs travailleurs et à un travailleur d'être joué par plusieurs individus.
Dans l'exemple de la figure 6, un individu, Sylvie, peut être *Travailleur: concepteur de cas d'utilisation* le matin et agir en tant *que Travailleur: critique de conception* l'après-midi. Paul et Marie sont tous les deux des *concepteurs*, quoiqu'ils soient sans doute responsables de classes différentes ou de différents ensembles de conception.

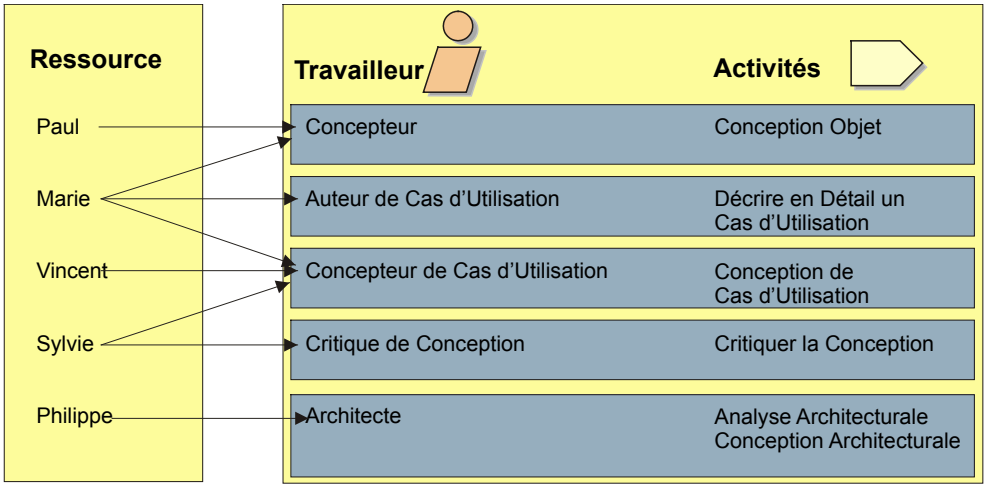

Figure 6 Personnes et travailleurs

Pour chaque travailleur, on s'attend à ce que l'individu désigné comme tel possède un ensemble de compétences. Sylvie doit savoir comment concevoir un cas d'utilisation et comment critiquer une partie de la conception.
Les travailleurs sont généralement désignés dans le processus accompagné du mot *Travailleur*, comme dans *Travailleur: testeur d'intégration*. L'annexe A contient la liste de tous les travailleurs définis dans le Rational Unified Process.

Activités

Une *activité* d'un travailleur particulier est une unité de travail qu'un individu dans ce rôle peut être amené à effectuer. L'activité a un but bien précis, souvent exprimé en termes de création ou de mise à jour de artefacts, tels qu'un modèle, une classe ou un plan. Chaque activité est attribuée à un travailleur particulier. La granularité d'une activité est généralement de quelques heures à quelques jours. En général, elle n'implique qu'un seul travailleur et ne concerne qu'un seul ou un petit nombre de artefacts. Une activité doit pouvoir être utilisée en tant

[6] Toutefois, on dit souvent « Le concepteur de la classe X fait ceci » alors qu'à strictement parler on devrait dire « L'individu agissant en tant que concepteur de la classe X fait ceci ».

qu'élément de planification et de progression; si elle est trop petite, on la négligera et si elle est trop importante, la progression devra s'exprimer en termes de parties d'une activité.
Les activités peuvent être répétées plusieurs fois sur le même artefact, en particulier d'une itération à une autre lorsque le système s'affine et s'agrandit. Des activités répétées peuvent être effectuées par le même travailleur, mais pas nécessairement par le même individu.
En termes d'orientation objet, un travailleur est un objet actif et les activités effectuées par ce travailleur sont les opérations faites sur cet objet. Voici des exemples d'activités.

- *Planifier une itération:* effectuée par le Travailleur: chef de projet
- *Trouver des cas d'utilisation et des acteurs:* effectuée par le Travailleur: analyste système
- *Critiquer la conception:* effectuée par le Travailleur: critique de conception
- *Exécuter un test de performance:* effectuée par le Travailleur: testeur de performance.

Les activités sont généralement accompagnées du mot *Activité*, comme dans *Activité: trouver des cas d'utilisation et des acteurs.*

Étapes d'activité

Les activités se subdivisent en étapes. Les étapes rentrent dans trois catégories principales.

- *Étapes de réflexion*
  Le travailleur comprend la nature de la tâche, rassemble et examine les artefacts d'entrée et formule un diagnostic.
- *Étapes d'actions*
  Le travailleur crée ou met à jour des artefacts.
- *Étapes de critique*
  Le travailleur examine les résultats en leur appliquant certains critères.

On n'effectue pas forcément toutes les étapes chaque fois qu'on effectue une activité, car elles peuvent être exprimées sous forme d'enchaînements alternatifs. Par exemple, *l'Activité: trouver des cas d'utilisation et des acteurs* se décompose en plusieurs étapes.

1. Trouver des acteurs
2. Trouver des cas d'utilisation
3. Décrire comment les acteurs et les cas d'utilisation interagissent
4. Mettre les cas d'utilisation et les acteurs dans des paquetages
5. Présenter le modèle de cas d'utilisation en diagrammes de cas d'utilisation
6. Préparer un examen du modèle de cas d'utilisation
7. Évaluer les résultats

La partie de recherche (étapes 1 à 3) demande de la réflexion; les parties d'accomplissement (étapes 4 à 6) nécessitent de placer le résultat dans le modèle de cas d'utilisation; la partie de critique (étape 7) demande au travailleur d'évaluer le résultat pour juger de sa complétude, de sa robustesse, de son intelligibilité ou d'autres qualités.

Artefacts

Un *artefact* est un élément d'information qui est fabriqué, modifié ou utilisé par un processus. Les artefacts sont les résultats tangibles du projet: les choses que le projet génère ou utilise tandis que l'on progresse vers le produit final. Des artefacts sont utilisés comme entrées par les travailleurs pour effectuer une activité. Ils sont aussi le résultat ou la sortie de telles activités. En termes de conception orientée objet, de même que les activités sont les opérations d'un objet actif (le travailleur), les artefacts sont les paramètres de ces activités.
Les artefacts peuvent prendre diverses apparences et formes.

- Un modèle, tel que le modèle de cas d'utilisation ou le modèle de conception
- Un élément de modèle—un élément à l'intérieur d'un modèle—tel qu'une classe, un cas d'utilisation ou un sous-système
- Un document, tel qu'une étude de rentabilité ou un document d'architecture du logiciel
- Du code source
- Des exécutables

Il faut noter qu'*artefact* est le terme utilisé dans Rational Unified Process. D'autres processus utilisent des termes différents tels que sous-produit, produit de travail, unité de travail, etc., pour désigner la même chose. Les éléments livrables (« deliverables ») sont uniquement le sous-ensemble de tous les artefacts qui se retrouvent entre les mains des clients et des utilisateurs finaux.
Des artefacts peuvent aussi être composés d'autres artefacts. Par exemple, le modèle de conception contient un grand nombre de classes; le plan de développement du logiciel contient d'autres plans: un plan de répartition du personnel, un plan des phases, un plan de métrique, des plans d'itération, etc.
Les artefacts sont presque tous sujet au contrôle de version et à la gestion de configuration. Quelquefois, cela peut se faire seulement en versionnant un artefact «conteneur» lorsqu'il n'est pas possible de le faire pour les artefacts élémentaires contenus dans ce «conteneur». Par exemple, on peut contrôler les versions du modèle de conception complet ou du paquetage de conception, mais pas nécessairement celles des classes individuelles qu'ils contiennent.
En général, les artefacts ne sont *pas* des documents. Plusieurs processus mettent un accent trop prononcé sur les documents, en particulier les documents sur papier. Le Rational Unified Process décourage la production systématique de documents sur papier. L'approche la plus efficace et la plus pragmatique pour gérer les artefacts d'un projet est de les garder *à l'intérieur* de l'outil approprié que l'on a utilisé pour les créer et les gérer. Lorsque cela est nécessaire, on peut produire des documents (des instantanés) à l'aide de ces outils juste au moment où l'on en a besoin.
On peut livrer des artefacts aux différentes parties intéressées à l'intérieur de l'outil ou tout en livrant l'outil, plutôt que sur papier. Cette approche garantie que les informations seront toujours à jour et fondées sur le travail réel du projet; produire ces informations ne doit pas demander d'efforts supplémentaires.
Voici des exemples de artefacts.

- Un modèle de conception exprimé avec Rational Rose
- Un plan de projet exprimé avec Microsoft Project

- Un défaut exprimé avec ClearQuest
- Une base de données des besoins du projet en Requisite Pro

Toutefois, certains artefacts doivent être de simples documents textes, dans le cas, par exemple, d'une source externe au projet ou lorsque c'est simplement la meilleur façon de présenter des informations descriptives.

Rapports

Les modèles et les éléments de modèles peuvent être associés à des rapports. Un *rapport* extrait d'un outil des informations sur un modèle et sur des éléments de modèle. Par exemple, un rapport présente un artefact ou un ensemble de artefact pour une revue. Contrairement aux artefacts habituels, les rapports ne sont pas soumis au contrôle de version. On peut les générer n'importe quand, en réutilisant les artefacts qu'on a utilisés pour les créer.
Les artefacts sont en général désignés à l'aide du mot *Artefact*, comme dans *Artefact: Scenario de cas d'utilisation*.

Ensemble de artefacts

Les artefacts du Rational Unified Process appartiennent à l'une des cinq catégories suivantes, appelées *ensembles d'informations*.

- L'ensemble de gestion
- L'ensemble des besoins
- L'ensemble de conception
- L'ensemble d'implémentation
- L'ensemble de déploiement

L'*ensemble de gestion* regroupe tous les artefacts qui ont trait aux affaires du logiciel et à la gestion du projet.

- Les artefacts de planification, tels que le plan de développement du logiciel, l'étude de rentabilité, l'instance du processus effectivement utilisé pour le projet (le «cas de développement»), etc.
- Les artefacts opérationnels, tels que la description d'une livraison, des évaluations de situation, des documents de déploiement et des données sur les anomalies.

L'*ensemble des besoins* regroupe tous les artefacts qui ont trait à la définition du logiciel devant être développé.

- Le document de vision
- Les besoins sous la forme des nécessités des intervenants, le modèle de cas d'utilisation et les spécifications supplémentaires
- Le modèle métier, si l'on en a besoin pour comprendre les processus métier soutenus par le système

L'*ensemble de conception* contient une description du système à construire (ou tel qu'il a été construit) sous la forme des éléments suivants.

- Du modèle de conception
- De la description de l'architecture
- Du modèle de test

L'*ensemble d'implémentation* contient :

- Le code source et les exécutables

- Les fichiers de données associés ou les fichiers nécessaires pour les produire

L'*ensemble de déploiement* contient toutes les informations livrées, y compris :

- Les scripts d'installation
- La documentation de l'utilisateur
- Le matériel de formation

Dans un processus de développement itératif, les différents artefacts ne sont pas construits, achevés ou même figés dans une phase avant de passer à la phase suivante. Au contraire, les cinq ensembles d'information évoluent tout au long du cycle de développement. On les fait *pousser*, comme le montre la figure 7.

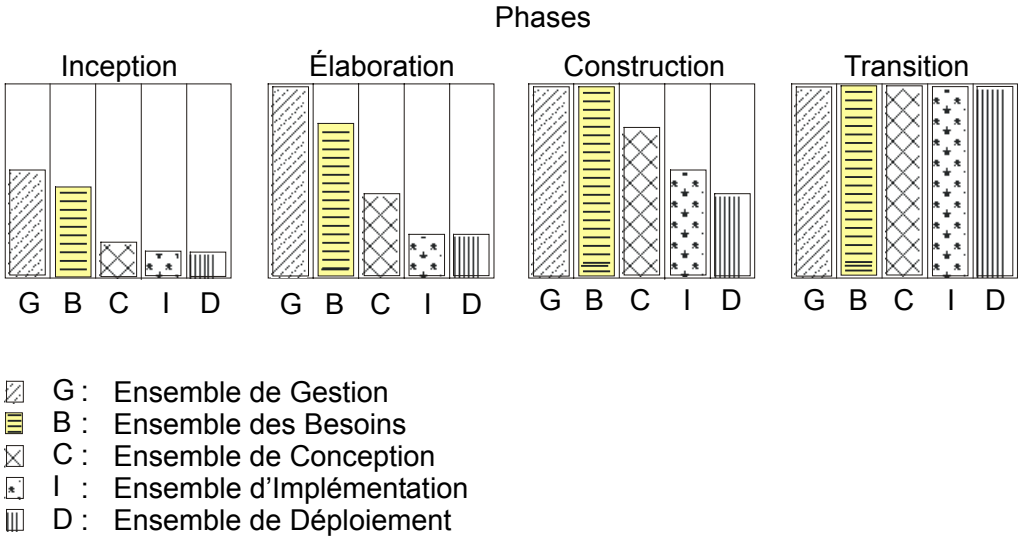

Figure 7 Progression des ensembles d'information

Enchaînement d'activités

Une simple énumération de tous les travailleurs, activités et artefacts ne constitue pas tout à fait un processus. On a besoin d'un moyen pour décrire des suites significatives d'activités qui produisent un résultat valable, et de montrer les interactions entre les travailleurs.

Un *Enchaînement d'activités* («workflow»)est une suite d'activités qui produit un résultat d'une valeur observable. En termes UML, un enchaînement peut s'exprimer par un diagramme de séquence, un diagramme de collaboration, ou un diagramme d'activité. Dans ce livre, on utilisera une forme de diagramme d'activité. La figure 8 donne un exemple d'enchaînement.[7]

[7] À strictement parler, les enchaînements de travaux sont des *classes d'enchaînements de travaux*, pour lesquelles il peut y avoir beaucoup d'*instances d'enchaînements de travaux*.

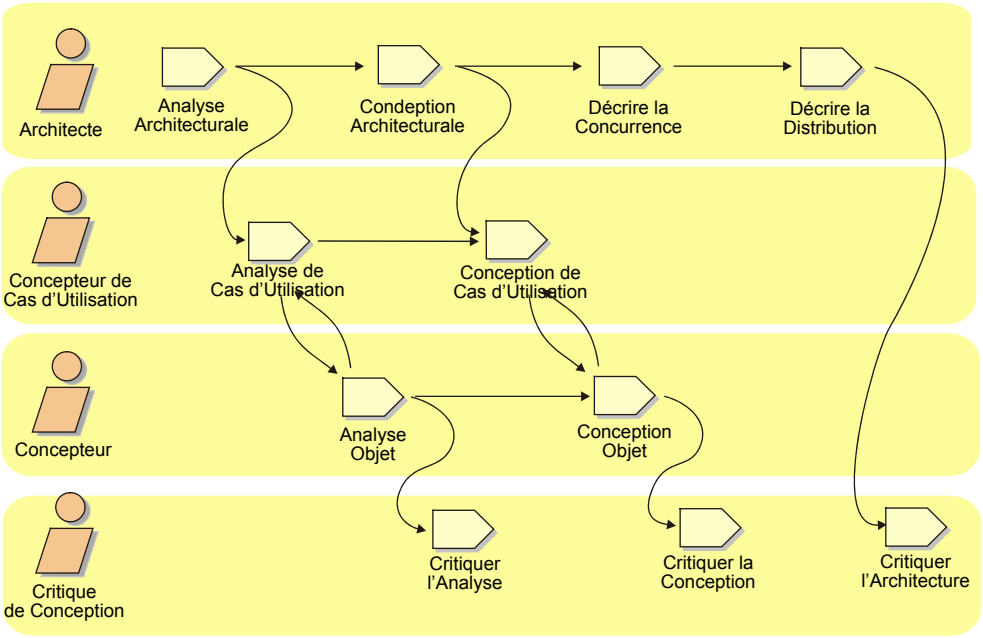

Figure 8 Exemple d'enchaînement d'activités

On notera qu'il n'est pas toujours possible ou pratique de représenter toutes les dépendances entre les activités. Souvent, deux activités sont plus intimement liées que ce que l'on montre, spécialement lorsqu'elles concernent le même travailleur ou le même individu. Les hommes ne sont pas des machines et un enchaînement ne doit pas être suivi à la lettre. Il ne s'agit pas d'un programme que les personnes devraient suivre de façon mécanique.
Il existe plusieurs manières d'organiser un ensemble d'activités en enchaînements. Nous avons organisé le Rational Unified Process en utilisant les suivants.

- Les enchaînements principaux
- Les enchaînements d'itération
- Les détails d'enchaînement

Enchaînements principaux

Il existe neuf *enchaînements principaux* dans le Rational Unified Process et ils représentent une répartition de tous les travailleurs et activités dans des regroupement logiques (voir figure 9). Les enchaînements principaux se divisent en six enchaînements principaux d'ingénierie et trois enchaînements principaux de soutien. Voici les enchaînements de travaux d'ingénierie.

1. L'enchaînement de modélisation du métier
2. L'enchaînement des besoins
3. L'enchaînement d'analyse et de conception
4. L'enchaînement d'implémentation
5. L'enchaînement de test
6. L'enchaînement de déploiement

Les trois enchaînements de soutien sont:

1. L'enchaînement de gestion de projet
2. L'enchaînement de gestion de la configuration et des changements

3. L'enchaînement d'environnement

Bien que les noms des six enchaînements principaux d'ingénierie puissent évoquer les phases séquentielles dans un processus traditionnel en cascade, les phases d'un processus itératif sont différentes et que ces enchaînements sont revisités de nombreuse fois tout au long du cycle de vie. L'enchaînement réel et complet des travaux d'un projet fait s'entrelacer ces neuf enchaînements principaux et les répète avec plus ou moins d'insistance ou d'intensité à chaque itération.

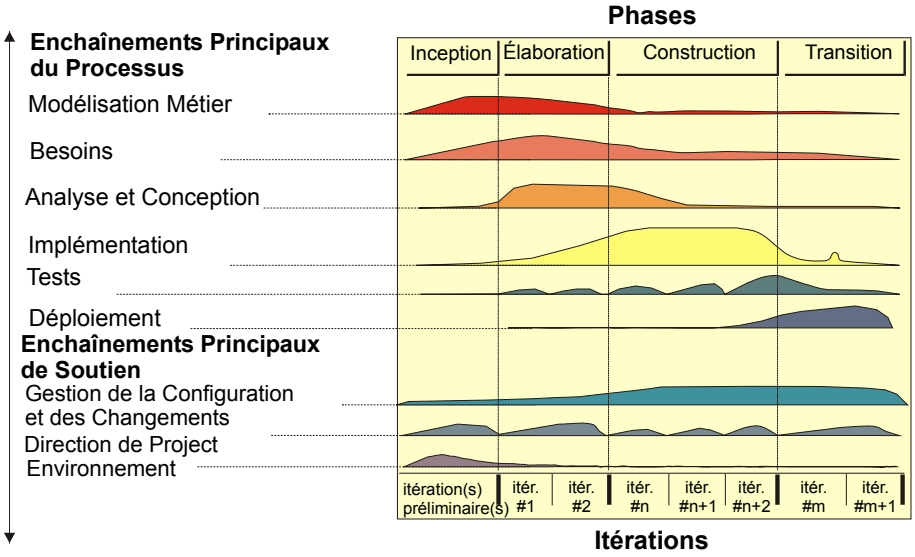

Figure 9 Neufs enchaînements principaux du processus

Enchaînements d'itération

Les enchaînements d'itération sont un autre moyen de présenter le processus, en le décrivant plutôt du point de vue de ce qui arrive pendant une itération typique. On peut considérer qu'ils constituent des instances du processus pour une itération. Il existe un nombre infini de façons d'instancier le processus. Le Rational Unified Process contient les descriptions de quelques enchaînements d'itération typiques.

Détails d'enchaînement

Chacun des enchaînements principaux recouvre beaucoup de matériel. Pour les subdiviser, le Rational Unified Process utilise les *détails d'enchaînements* pour exprimer un groupe particulier d'activités qui sont étroitement liées; elles sont effectuées ensemble ou de façon cyclique; elles sont effectuées par un groupe de personnes qui travaillent ensemble dans un «workshop»; ou elles produisent un résultat intermédiaire intéressant. Les détails d'enchaînements montrent aussi les flots d'information—les artefacts qui sont des entrées et des sorties pour les activités—montrant comment les activités réagissent aux différents artefacts.

Éléments additionnels du processus

Les travailleurs, les activités (organisées en enchaînements) et les artefacts représentent l'épine dorsale de la structure statique du Rational Unified Process.

Mais on peut ajouter d'autres éléments aux activités et aux artefacts pour rendre le processus plus facile à comprendre et à utiliser ainsi que pour mieux guider l'utilisateur. Ces éléments du processus additionnels sont :

- Des principes et conseils
- Des gabarits
- Des guides d'utilisation d'outil
- Des concepts

Ces éléments mettent en valeur les éléments primaires, comme le montre la figure 10.

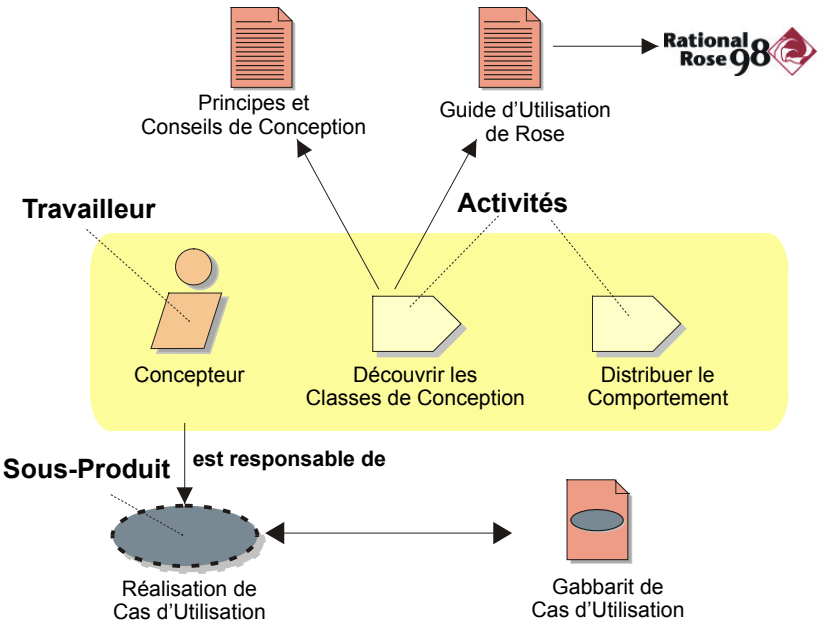

Figure 10 Ajout de gabarits, de guides d'utilisation d'outils et de principes et conseils

Principes et conseils

Les activités et les étapes sont exprimées de façon brève et précise, car le but est qu'elles servent de références pour ce qu'on a besoin de faire. Mais le néophyte, ou l'utilisateur expérimenté qui a oublié comment procéder, voudra plus d'information, cherchera à se faire guider.

On rattache aux activités, aux étapes et aux artefacts des *principes et conseils*. Les principes et conseils sont des règles, des recommandations ou des heuristiques qui servent de support, en complément des activités et les étapes. Ils décrivent des artefacts bien formés, en s'attachant particulièrement à la qualité. Les principes et conseils décrivent des techniques particulières, telles que les transformations d'un artefact en un autre ou l'utilisation du UML. Les principes et conseils s'utilisent aussi pour *évaluer* la qualité des artefacts—sous la forme de listes de vérifications («checklists») associées aux artefacts—ou pour passer les activités en revue. Voici des exemples de principes et conseil.

- Principes et conseils de modélisation, qui décrivent des éléments de modélisation bien formés, tels que des cas d'utilisation, des classes ou des cas de test

- Principes et conseils de programmation, pour les langages tels que C++, Java ou Ada, qui décrivent des programmes bien formés
- Principes et conseils pour la conception d'interface utilisateur
- Principes et conseils qui décrivent comment créer un artefact particulier, tel qu'une liste de risques ou un plan d'itération
- Principes et conseils de travail, qui donnent des avis pratiques sur la façon d'entreprendre une activité, en particulier pour des actions de groupe
- Listes de vérifications utilisées pendant une revue ou par un travailleur qui vérifie qu'une activité est terminée

Certains principes et conseils peuvent demander à être affinés ou spécialisés pour une organisation ou un projet donné afin de tenir compte des spécificités du projet, telle que l'utilisation d'une technique ou d'un outil particulier.
Voici des exemples de ce dernier type de principes et conseils.

- Principes et conseils d'interface utilisateur, tel que la description du style de fenêtrage particulier au projet: palette de couleurs, fontes de caractères, galerie d'icônes, etc.
- Principes et conseils de programmation, telle que la description des conventions de nommage spécifiques du projet

Gabarits

Les *gabarits* (« templates ») sont des « modèles », ou des prototypes, de artefact qui sont associés à la description d'un artefact. Les gabarits sont liés à l'outil avec lequel on doit les utiliser. Par exemple:

- Gabarits Microsoft Word, pour les documents et certains rapports
- Gabarits SoDA pour Microsoft Word ou FrameMaker, qui extraient les informations d'outils tels que Rational Rose, Requisite Pro ou TeamTest
- Gabarits Microsoft FrontPage, pour les divers éléments du processus
- Gabarit Microsoft Project, pour le plan de projet

Comme pour les principes et conseils, les organisations peuvent désirer remanier les gabarits avant de les utiliser pour ajouter le logo de l'entreprise, une identification du projet ou des informations spécifiques au type de projet.

Guides d'utilisation d'outils

Les activités, les étapes et leurs principes et conseils associés fournissent des guides généraux à l'utilisateur. En allant un peu plus loin, les *guides d'utilisation d'outils* («tool mentors») sont des aides supplémentaires qui montrent comment effectuer certaines étapes au moyen d'un outil logiciel particulier. Des guides d'utilisation d'outils sont fournis dans le Rational Unified Process, reliant ses activités aux outils Rational Rose, Requisite Pro, ClearCase, ClearQuest et TestStudio, etc. Les guides d'utilisation d'outils sont le seul endroit où les outils sont directement référencés. Une organisation peut étendre le concept de guide d'utilisation d'outil pour fournir des conseils pour d'autres outils.

Concepts
Certains des concepts clés, tels qu'itération, phase, risque, tests de performance, et ainsi de suite, sont présentés dans des sections séparées du processus, généralement associées à l'enchaînement des activités principal le plus approprié. La plupart des concepts clés sont aussi présentés dans le livre d'introduction.

Une cadre de processus

Avec cette structure, le Rational Unified Process constitue un cadre de processus, un processus générique. Les travailleurs, les artefacts, les activités, les principes et conseils, les concepts et les guides d'utilisation sont des éléments que l'on peut ajouter ou remplacer pour faire évoluer ou pour adapter le processus aux nécessités de l'organisation. Comment réaliser cela est développé plus en profondeur dans l'enchaînement d'environnement.

Résumé

- Le Rational Unified Process est un processus de développement du logiciel couvrant tout le cycle de vie de développement.
- Ce produit est une source de connaissances, toujours à jour, et facilement accessible depuis la station de travail de chaque développeur.
- Il contient des directives et des conseils sur un grand nombre de techniques et d'approches modernes : la technologie objet et le développement à base de composants, la modélisation et UML, l'architecture, le développement itératif, etc.
- Ce n'est pas un produit figé ; au contraire, il est vivant, constamment maintenu et en évolution.
- Il se fonde sur une solide architecture de processus, ce qui permet à une organisation de développement de le configurer et de l'adapter à ses besoins.
- Il intègre les six meilleures pratiques du développement du logiciel.
- Il est supporté par un vaste éventail d'outils.
- Le modèle du Rational Unified Process est construit sur trois entités fondamentales: les travailleurs, les activités et les artefacts.
- Les enchaînements relient les activités et les travailleurs pour former des séquences qui produisent des résultats ayant une certaine valeur.
- Des principes et conseils, des gabarits et des guides d'utilisation d'outils complètent la description du processus en donnant des conseils plus détaillés à l'utilisateur.
- Le Rational Unified Process est un cadre de processus organisé de façon à ce que l'on puisse configurer sa structure statique.