Programmez! le 1er magazine de tous les langages

N° et date de parution : 89 - 01/09/2006
Diffusion : 30000
Périodicité : Mensuel
Programmez_89_20_324.pdf
Page : 20
Taille : 90 %

# Gérer le cycle de vie logiciel

*Les projets informatiques connaissent bien souvent des retards importants. Leur coût se révèle dans bien des cas supérieur aux prévisions et les logiciels produits ne sont pas toujours conformes aux exigences des utilisateurs. Ces difficultés rendent nécessaire une gestion efficace du cycle de vie des logiciels.*

Le cycle de vie des logiciels est la description du processus couvrant les phases de création et de distribution d'un logiciel. Le but de ce découpage est la maîtrise des risques, des délais et des coûts. Il permet, quand il est bien implémenté, d'obtenir une qualité conforme aux exigences. Le cycle de vie est généralement découpé selon les phases suivantes : la définition des besoins, la planification et la gestion de projet, la conception, la réalisation, l'intégration, la qualification, l'exploitation et enfin la maintenance.

### Le cycle de vie décomposé

La phase de définition des besoins permet au client de définir son cahier des charges. On y retrouve les fonctionnalités attendues du logiciel ainsi que les contraintes non fonctionnelles comme les temps de réponse. La phase de planification et de gestion de projet permet de découper le projet en tâches, de décrire leurs enchaînements dans le temps et d'affecter à chacune une durée et un effort. Cette phase permet également de définir les normes qualité ainsi que les règles qui régiront les tests. La phase de conception permet de définir l'architecture du logiciel ainsi que les interfaces entre les différents modules qui le composent. La réalisation permet la production des modules constitutifs du logiciel. Chaque module est testé indépendamment des autres. On appelle cela les tests unitaires. La phase d'intégration permet le regroupement de chaque module avec les autres suivant le plan d'intégration. Pendant la qualification, le logiciel est testé dans des conditions normales d'utilisation et dans des conditions extrêmes de charge. A l'issue de cette phase, le logiciel est prêt à la mise en exploitation.

### Cycle de vie : le modèle incrémental et itératif

Certains modèles éprouvés, permettent de tirer profit des connaissances acquises dans le domaine de la gestion du cycle de vie. Parmi eux, il y a le modèle incrémental et itératif. Ce modèle est dit itératif car chaque version du logiciel constitue une itération. L'identification des risques a lieu rapidement dans le cycle de vie des logiciels, quand il est encore possible de les prendre en compte et de réagir d'une façon efficace. Chaque itération permet d'affiner les composants du logiciel d'une façon prévisible. Le processus mis en œuvre dans ce modèle est représenté par la figure 1.

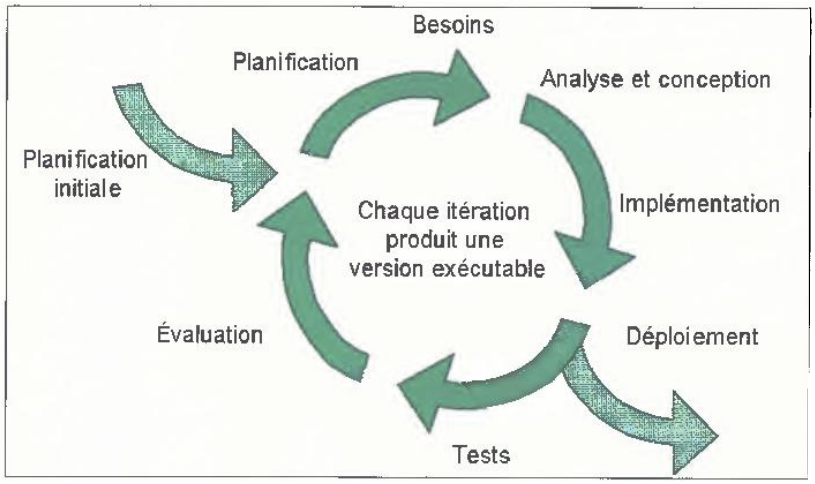


*Figure 1 : modèle itératif*

Les grands éditeurs du marché ont rapidement saisi l'importance que pouvait prendre une gestion efficace du cycle de vie des logiciels. Certains d'entres eux ont développé toute une gamme de solutions axées autour de cette problématique. IBM avec sa méthode RUP (Rational Unified Process) et sa suite d'outils Rational basés sur ce processus. Borland avec ses solutions ALM (Application Lifecycle Management) et sa suite d'outils Caliber.

### RUP

IBM Rational Unified Process (RUP) est un processus de développement d'applications couvrant tout le cycle de vie du logiciel. RUP est également un process framework (cadre de processus) puisqu'il peut être étendu et adapté aux besoins de l'entreprise. RUP est constitué de milliers de Best Practices (meilleures pratiques), de modèles et d'exemples, de directives et de conseils traitant de sujets aussi divers que la technologie Objet, le développement à base de composants, la modélisation, l'architecture ou le développement itératif. La force du processus RUP tient au fait qu'il est utilisable en l'état ou adaptable aux besoins de l'entreprise qui souhaite l'étendre. Son objectif est de promouvoir la qualité et l'efficacité transversale des projets de développement de logiciels. RUP est basé sur un certain nombre de pratiques logicielles éprouvées. Parmis ces pratiques, on peux citer le développement itératif qui permet de prendre rapidement en considération les fréquents changements dans les cahiers des charges. On peut également citer la préconisation d'architectures à base de composants, afin de favoriser la réutilisation et de faciliter l'activité des tests. Citons également la modélisation graphique du logiciel à laquelle RUP accorde une place importante. La modélisation aide à comprendre et à mettre en forme le problème et sa solution. Enfin, une pratique importante préconisée par RUP est la mise en œuvre d'un contrôle de la configuration et des

Programmez! le 1er magazine de tous les langages

N° et date de parution : 89 - 01/09/2006
Diffusion : 30000
Périodicité : Mensuel
Programmez_89_20_324.pdf
Page : 22
Taille : 100 %

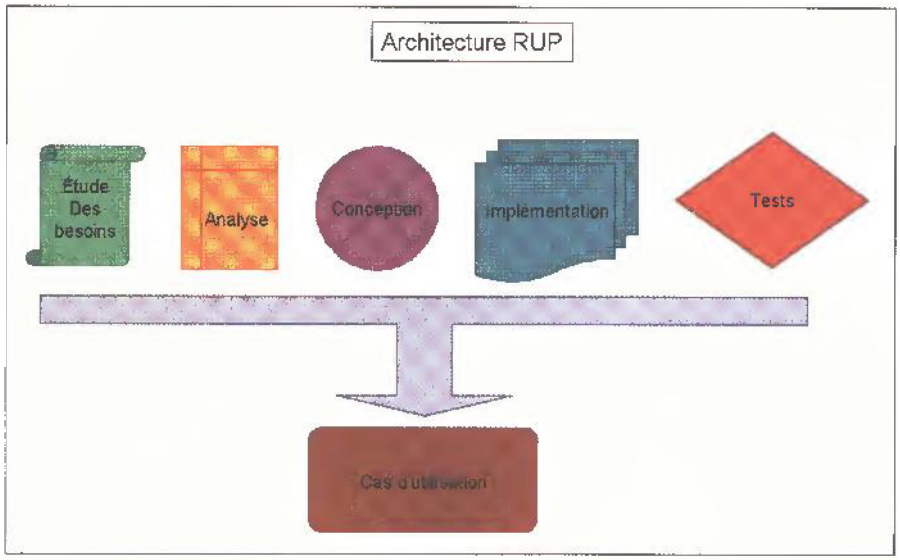


*Figure 2 : Architecture RUP*

changements, afin de conserver la trace de toutes les modifications effectuées sur l'ensemble des composants du logiciel. La force de l'architecture RUP tient au fait qu'elle est pilotée par les cas d'utilisation. Il s'agit d'un point de départ pour les activités de développement. D'abord parce que les cas d'utilisation définissent explicitement le comportement attendu du système, enfin parce qu'ils fournissent un lien important entre le cahier des charges et les composants du logiciel comme le montre la figure 2.

Parallèlement à RUP, IBM fournit une suite d'outils adaptés à chaque phase du cycle de vie des logiciels. Ces outils implémentent le process RUP et lui sont complémentaires. On trouve par exemple Rational Rose pour la conception et la modélisation de processus. La suite Rational Software pour l'implémentation, ainsi que Rational ClearCase pour la gestion de la configuration logicielle. IBM fournit également des solutions complètes comme Rational Software Architect (RSA). Ce produit permet de prendre en charge la conception, la réalisation, les tests unitaires et le contrôle qualité du logiciel.

## Borland ALM (Application Lifecycle Management)

Parmi les grands éditeurs du marché, Borland est celui qui a probablement le mieux compris la nécessité d'offrir des solutions complètes de gestion du cycle de vie. Son offre propose des produits pour chaque phase, mais aussi et surtout, une plate-forme ALM nommée Borland Core SDP. Cette plate-forme propose des solutions adaptées à chaque phase d'un projet, définition du besoin, conception, développement et tests. Elle offre également des fonctionnalités spécifiques à chaque type d'intervenant du projet, analyste, architecte, développeur ou testeur.

La plate-forme Borland Core SDP intègre des workflows automatisés et personnalisables permettant de contrôler le projet et offre des solutions d'automatisation des tâches récurrentes. L'architecture de cette plate-forme est représentée dans la figure 3.

Parallèlement à cette plate-forme, Borland fournit des solutions pour chaque phase du cycle de vie des logiciels. Pour la définition des besoins, le contrôle qualité et l'analyse d'impacts, on trouve la solution Borland Caliber. Cette suite propose l'outil Caliber DefineIT qui permet de faciliter l'expression et la compréhension du besoin par la génération de scénarios compréhensibles par tous les intervenants du projet. Une fois ces scénarios validés, les cas de tests et les modèles UML peuvent êtres automatiquement générés. CaliberRM permet de faire de l'analyse d'impact et facilite la communication dans la définition et la gestion d'un projet. Il centralise les besoins dans un référentiel unique et


*Figure 3 : La plate-forme Borland Core SDP*

permet de veiller au respect de la qualité des composants. Pour la conception, Borland fournit Together. Cette solution offre tous les outils de modélisation permettant aux équipes fonctionnelles et techniques de mieux se comprendre. Pour la gestion des configurations logicielles, on trouve Borland StarTeam qui prend en charge la gestion en configuration des applications et permet d'administrer l'ensemble du processus de livraison des logiciels. Enfin, pour les tests, la suite Silk propose des outils pour tester, mettre au point et surveiller les applications d'entreprise.

## Conclusion

Un certain nombre de pratiques et de règles ont étés mises au point afin de gérer au mieux la production d'un logiciel durant tout son cycle de vie. Elles ont donné naissance à des modèles et des frameworks utilisables en l'état ou adaptables par les entreprises. La forte demande pousse cependant les éditeurs à aller plus loin en proposant des gammes complètes d'outils implémentant les modèles éprouvés et prenant en charge chaque phase du cycle de vie des logiciels. Aujourd'hui, l'offre s'oriente de plus en plus vers des plates-formes complètes et intégrées regroupant tout ce qui est nécessaire à la gestion du cycle de vie des logiciels. Cette orientation est nécessaire parce qu'elle répond à un besoin fort des clients qui souhaitent que la production d'un logiciel informatique soit plus aisée, plus rapide, moins chère et plus fiable.

■ **Ali Meraoumia**
*Consultant SQLi*