


Rubrique :	Pge : 1	
	1/3	

## Open source: 8 conseils pour maîtriser la maintenance des applications

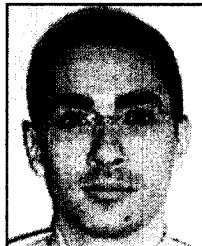
**La maintenance des projets open source est loin d'être triviale. C'est pourtant là que réside le coût caché de ces logiciels. Les conseils de trois spécialistes pour éviter les mauvaises surprises.**

Un nombre croissant d'entreprises assemblent et personnalisent des briques open source pour répondre à leurs besoins spécifiques. Elles économisent ainsi en coûts de licence, accélèrent leurs projets et les standardisent en partie. En utilisant des briques logiciel open source, les entreprises espèrent aussi diminuer leurs coûts de maintenance évolutive et corrective, pris en charge par la communauté.

Mais qu'advient-il lorsqu'une nouvelle version est publiée pour corriger une faille de sécurité ou apporter de nouvelles fonctionnalités? Comment mettre à jour une brique open source modifiée pour les besoins de l'entreprise? Et comment gérer les dépendances entre briques "standards" et "spécifiques"?

ZDNet a demandé conseil à trois spécialistes: Corinne Brunel, directrice d'Uperto ( filiale du groupe Devoteam), Nicolas Hoizey, directeur technique de Clever Age et Laurent Pierre, directeur technique de Linagora.

### 1 - Utiliser uniquement les briques les plus populaires



*Laurent Pierre:* «À l'amorce d'un projet d'envergure à base de logiciels libres, il est important de recenser toutes les briques et de choisir celles qui ont le plus de suivi, le plus grand nombre d'utilisateurs, la meilleure interopérabilité, etc. C'est l'expertise de l'intégrateur qui permet de limiter ce risque».

### 2 - Reverser le code modifié à la communauté

*Laurent Pierre:* «Une autre façon de limiter le risque est de contractualiser une prestation de suivi. Chez nous, cela consiste à reverser au maximum les développements spécifiques pour qu'ils soient acceptés et utilisés par la communauté. Cela garantit que les développements spécifiques de l'entreprise seront réintégrés à la version "standard" de la brique. Dans l'idéal, les nouvelles versions du logiciel ou de la brique prennent en compte la modification de l'entreprise. La mise à jour est donc quasiment transparente. Lorsque ce n'est pas le cas, notre rôle est de minimiser en continu l'écart entre la version de l'entreprise et les briques "communautaires"».

Rubrique :	Pge : 1
	2/3

### » 3 - Communiquer très tôt ses projets de modification à la communauté

*Laurent Pierre:* «Sur un de nos projet de mise en place d'une solution de travail collaboratif, nous avons contacté la communauté Mozilla Sunbird pour modifier la façon dont ce logiciel communique avec le serveur et autoriser les échanges par ce biais. En prévenant très tôt la communauté, on maximise la chance de voir ses modifications prises en compte par celle-ci».

### » 4 - Réaliser des tests de montée en charge et de non régression



Corinne Brunel

*Corinne Brunel:* «Un changement de version d'une brique implique des tests de non-régression et de compatibilité avec les autres briques, comme si l'on migrerait de version une base de données commerciale face à un ERP. L'avantage indéniable de l'open source est le respect des standards qui limite les risques de régression. Il est en général plus facile d'analyser les impacts possibles d'une migration et de gérer ce risque avec l'open source, car la totalité de la chaîne est maîtrisée, du code source au déploiement. Par ailleurs, l'une des complexités de l'assemblage de produits commerciaux est le risque lié à la concurrence entre les éditeurs. Cette notion n'existe pas en open source. C'est l'une des raisons qui font que les produits open source suivent les standards et interopèrent

*Laurent Pierre:* «Il faut aussi réaliser des tests unitaires pour valider les fonctionnalités apportées par la nouvelle brique et réaliser des tests de non-régression, en plaçant l'ensemble du système incluant la nouvelle brique sur une plate-forme de préproduction. Lors des tests unitaires, il ne faut pas oublier toutes les briques qui s'interfaçaient avec la brique ou le bout de code mis à jour. Par exemple, dans un projet de création de portail, quand il faut modifier la version de l'éditeur WYSIWYG (utilisé pour permettre aux utilisateurs d'insérer simplement du contenu), il suffit en général de mettre à jour le logiciel sur le serveur et de revoir les appels qui en sont faits dans les pages web qui le nécessitent. Au-delà des tests fonctionnels, il faut réaliser des tests techniques de montée en charge et d'analyse de performance. Des outils libres très performants sont utilisés dans ce sens (OpenSTA, Tsunami, etc.). Si l'ensemble des tests se passent de façon satisfaisante, on peut passer en production».

### » 5 - Respecter la philosophie des logiciels open source



Nicolas Hoizey

*Nicolas Hoizey:* «Les logiciels open source qui offrent une pérennité optimale sont architecturés autour d'un noyau pauvre fonctionnellement, mais robuste et performant. Stable dans le temps, ce noyau est enrichi par des extensions qui évoluent à leur propre rythme et permettent aux entreprises d'enrichir les fonctionnalités du logiciel au gré de leurs besoins sans toucher au cœur. Il faut respecter au maximum cette philosophie pour éviter les risques de "fork". Car en patchant le noyau, celui-ci devient spécifique et ne peut plus être mis à jour simplement».



Rubrique :	Pge : 1
	3/3

#### 6 - Créer des couches d'abstraction

*Nicolas Hoizey:* «Il arrive que l'interface de programmation (API) d'une brique open source évolue lors d'un changement de version majeure. La meilleure façon d'isoler ce genre de risque est de créer une API propre à l'entreprise. Ainsi les autres classes ou briques logicielles se référant à cette API ne seront pas impactées, et il suffira de mettre à jour le *mapping* entre l'API de l'entreprise et celle de la classe ou de la brique ayant évolué. Cela permet par exemple de changer facilement de moteur de recherche sur une gestion de contenu ou de base de données».

#### 7 - Privilégier les produits open source

*Corinne Brunel:* «Il est fréquent de voir l'open source utilisé pour gérer les évolutions de produits commerciaux. Par exemple, certaines entreprises mettent en place des frameworks ou serveurs d'application java open source pour garantir le respect du standard J2EE, et ainsi la portabilité du code source. Il ne faut pas perdre de vue que les produits open source sont souvent conçus à l'origine par des informaticiens, très conscients de ces problématiques techniques de production et de migration».

#### 8 - Consulter la communauté avant de mettre à jour une brique

*Corinne Brunel:* «La communauté open source dispose de nombreux forums de discussion et de sources d'informations car

la communication fait partie intégrante de la philosophie open source. Si l'on souhaite migrer de version, on trouve facilement de nombreux retours d'expérience permettant d'anticiper les points critiques».