Comment, pour une administration, assurer la mutualisation d'une application informatique ?

Thierry Aimé
Direction Générale des Impôts
86-92 Allée de Bercy
75574 Paris Cedex 12
thierry.aime@dgi.finances.gouv.fr

Résumé

Les administrations en nombre croissant souhaitent mutualiser certains de leurs développements informatiques sur le modèle des communautés Open Source. Nous présenterons d'abord le contexte dans lequel se pose la question de la mutualisation et ce qui la motive.

Ensuite en reprenant les principes, les règles, les pratiques propres au fonctionnement des communautés Open Source et mises en valeur par Eric S. Raymond dans son livre « La cathédrale et le bazar », nous montrerons les difficultés à suivre le modèle pour une administration.

Enfin, nous proposerons une démarche au moyen d'un marché d'assistance, pour une tierce maintenance applicative (TMA) classique. A celle-ci nous ajouterons des activités spécifiques d'animation et d'intégration pour faire vivre la communauté, et pour prendre en charge les reversements. Nous fournirons en particulier des éléments quantitatifs permettant de piloter ces nouvelles activités.

Mots clefs

ingenierie logicielle, open source, maintenance applicative, organisation, marchés publics

1 Introduction

Si les administrations françaises possèdent un coeur de métier spécifique, elles n'en partagent pas moins de vastes domaines applicatifs, comme la gestion des personnels, des ressources, du temps, sans parler évidemment des infrastructures de son système d'information. Inévitablement vient la question de la mutualisation des investissements en matière de développement logiciel, encouragée en cela :

- sur le plan économique, par la possibilité de partager les coûts de maintenance évolutive et corrective ;
- sur le plan pratique, par de nombreuses plates-formes collaboratives dédiées à la mutualisation informatique ;
- sur le plan juridique, par les licences de logiciel libre.

Ainsi un nombre croissant d'administrations publient des développements informatiques. Cette action si elle n'est pas tout à fait triviale, est néanmoins ponctuelle et facile à cerner. Elle peut même être envisagée comme étape ultime du développement initial. La vrai difficulté ensuite va être de faire naître et vivre une communauté d'utilisateurs et de contributeurs capable de mutualiser la charge de maintenance de l'application.

L'objet de cet article est donc de proposer une démarche de mutualisation, adaptant au contexte et moyens des administrations, les modes opératoires des communautés Open Source. Il ne s'agit pas d'un retour d'expérience simplement parce que les marchés intégrant des activités communautaires sont peu nombreux et récents. Approchant voir inspirant l'approche ici présentée, nous pouvons citer le marché SPIP-Agora lancé en début d'année 2007 par le Secrétariat Général du Gouvernement, ou le marché Acube porté par le Ministère des affaires étrangères actuellement en phase de lancement.

2 Contexte

Le point de vue adopté dans cette article est celui d'une Administration ou d'un organisme public. Il n'est vraiment spécifique de celui d'une organisation privée que sur quelques points :

- le recours à une maîtrise d'oeuvre externe se fera dans le cadre des marchés publics ;
- la diffusion de briques du système d'information ne présente pas le risque d'une perte d'un avantage concurrentiel.

Les éléments de contexte qui suivent n'ont rien d'arbitraires. Ils sont le résultat d'observations menées depuis trois ans, auprès des différents ministères quant aux moyens mis en oeuvre pour leurs développements informatiques.

Nous supposerons qu'une société de service a été commanditée dans le cadre d'un marché public pour la réalisation de l'application. En effet, rares sont aujourd'hui les projets réalisés en interne ; les effectifs des services informatiques de l'administration n'ont pas crû à la mesure de l'extension des besoins. Le recours à des marchés de réalisation est la règle, l'administration abandonnant la maîtrise technique des projets pour se concentrer sur la maîtrise d'ouvrage et l'expertise fonctionnelle[1].

Il est aujourd'hui très exceptionnel que des administrations anticipent la mutualisation de leurs applications[2]. Cette hypothèse n'est généralement envisagée qu'une fois la mise en production réalisée, donc bien après les phases de

[1] Il y a des exceptions, à la DGI ou la Gendarmerie Nationale...

[2] Il y a quelques projets sur AdmiSource qui échappent à cette règle, Acube, Geosource, Milimail,...

recettes. En conséquence nous considérerons que la mutualisation intervient sur une application stable.

Évidemment nous supposerons que l'application est publiable sous une licence de logiciel libre. Ce sera possible à condition que :

- l'administration se soit réservée, contractuellement, l'ensemble des droits patrimoniaux sur les développements spécifiques livrés[3] par le titulaire du marché[4] de réalisation ;
- les éventuels composants intégrés dans l'application soient couverts par des licences libres compatibles entre elles (C'est généralement le cas même si parfois de mauvaises surprises ne sont pas à exclure)[5].

Nous ne préjugeons rien de la nature fonctionnelle de l'application promise à la mutualisation. Pour autant des applications d'infrastructures, ou peu marquées en termes métier, sont probablement plus porteuses.

Enfin, le plus important des pré-requis est la volonté de l'administration de rechercher dans une approche Open Source la solution d'une équation difficile consistant à réduire les coûts tout en garantissant la maîtrise et la qualité des services, anticipant en cela :

- l'ouverture du marché de la maintenance qui ne sera plus réservé au titulaire initial du marché de réalisation ;
- le partage des coûts de maintenance grâce aux reversements des autres acteurs ;
- l'amélioration de la qualité et de la robustesse de l'application, grâce aux retours d'utilisateurs opérant dans des contextes différents.

3 Un modèle difficile à transposer

Il n'est certainement pas pertinent de parler d'un modèle Open Source tant la réalité observée des communautés démontre une multitude de situations différentes, où l'invention organisationnelle et technique est incessante. Toutefois il est des règles, des pratiques, des principes largement partagés par l'ensemble des communautés Open Source. C'est ce que Eric S. Raymond a remarquablement décrit dans son livre « The cathedral and the bazaar » [1].

Nous allons à partir des caractéristiques les plus marquantes des communautés Open Source, montrer les difficultés d'une transposition pour une administration.

3.1 Continuité communautaire

Pour une application Open Source, l'élément fondamental est sa communauté que l'on peut définir comme l'ensemble des personnes intéressées par l'application.

Celle-ci se constitue au fil du temps, autour de l'initiateur du projet, généralement un développeur « que quelque chose démangeait »[6] ! Les premières personnes rejoignant le projet disposeront plutôt de profil technique et collaboreront au développement de l'application. Les simples utilisateurs ne rejoindront vraiment la communauté qu'avec les premières versions stables.

Ce qui se révèle caractéristique d'une communauté Open Source c'est l'implication continue d'au moins un développeur, souvent à l'origine du projet [2]. Au fil des années, dépositaire de l'histoire du projet, gardien des procédures, et des règles de fonctionnement, fin connaisseur de l'architecture technique de l'application, il est capable de qualifier et intégrer les contributions de tiers ; il est le garant de la pérennité du projet.

Il n'y a pas de communauté Open Source sans développeur régulièrement impliqué.

La difficulté pour une administration, c'est qu'elle ne dispose pas des éléments de continuité favorable à la constitution de la communauté initiale. En effet elle ne possède aucune ressource propre en développement impliquée sur le projet et le titulaire du marché de réalisation a quitté le projet. La communauté initiale ne sera composée que d'utilisateurs !

Si il n'est pas impossible que quelques développeurs rejoignent à terme la communauté, l'appropriation technique du projet sera difficile car elle ne profitera pas du passage de témoin d'un ancien [7]. Il est très peu probable qu 'un tel effort soit consenti sur une base bénévole.

3.2 Conquérir les utilisateurs[8]

La force d'une communauté Open Source se mesure au nombre de ses membres, gage de dynamisme. En effet une large base d'utilisateurs, augmente les chances de contributions, contributions qui renforceront la qualité du projet et finalement l'attractivité de la communauté. Aussi éloigné que cela puisse paraître de l'esprit Open Source un travail « marketing » est nécessaire. Cela passe en particulier par la réalisation d'un site internet attractif présentant le projet et les ressources disponibles (documentations, forums, listes de diffusions, ...).

Un point souvent négligé est la qualité du « packaging » de l'application, la facilité de son installation, sa faible adhérence au contexte, afin d'abaisser le coût d'entrée dans la communauté. Le processus de développement, itératif, guidé par les besoins des utilisateurs va permettre d'ajuster la généricité de l'application au fur et à mesure de l'extension de ses usages.

L'application portée par une administration ne connaîtra pas cette lente acclimatation à des environnements variés. Elle a d'abord été réalisée pour répondre à un besoin localisé. Le contexte va ainsi largement imprégner le développement de l'application. Au final l 'appropriation de l'application par une autre entité risque d'être difficile et supposera même probablement des adaptations.

---

[3] Au moyen de clauses idoines incluses dans le Cahier des Clauses Administratives Particulières

[4] C'est l'entreprise qui a remporté l'appel d'offre.

[5] On pourra se reporter aux « Guide de choix et d'usage des licences de logiciels libres pour les administrations » :

http://synergies.modernisation.gouv.fr/article.php?id_article=238

[6] « Tout bon logiciel commence par gratter un développeur là où ça le démange. » Eric S. Raymond [1]

[7] « Quand un programme ne vous intéresse plus, votre dernier devoir à son égard est de le confier à un successeur compétent. » [1]

[8] Un chapitre de la cathédrale et le bazar s'intitule « de l'importance d'avoir des utilisateurs » [1]

Les procédures d'installation accompagnant les applications des administrations sont souvent très complexes car elles répondent d'emblée à des exigences fortes de production. Cette complexité sera dissuasive pour l'utilisateur souhaitant simplement évaluer l'application avant une utilisation plus large.

Enfin l'administration n'a généralement pas anticipé, ni produit les éléments de langages technico-fonctionnels susceptibles de figurer sur un site web collaboratif, car il n'y a pas à promouvoir le déploiement de l'application au sein de l'organisation, mais généralement uniquement l'usage de l'application.

3.3 Au service des utilisateurs

Une caractéristique importante des communautés Open Source est l'attention portée aux problèmes et anomalies signalés par les utilisateurs, ceux-ci devenant comme le dit Eric S. Raymond, co-developpeurs de l'application[9]. Cette attention, si elle peut être jugée coûteuse au premier abord, permet de stabiliser et d'améliorer très rapidement l'application.

Dans le cadre de la mutualisation d'une application par une administration, le recours à un marché va délimiter le périmètre des utilisateurs bénéficiant de la maintenance corrective et les autres au risque de morceler la communauté des utilisateurs. Ceci est dommageable et repose sur l'idée fausse que cela va augmenter le nombre des anomalies à corriger[10]. Or le nombre d'anomalies est déterminé par la taille de l'application et la qualité du développement. Ainsi la conséquence d'une base d'utilisateurs élargie est d'intensifier la correction d'anomalies dans les périodes suivant la sortie d'une nouvelle version et donc de stabiliser plus rapidement l'application. Le nombre d'anomalies globalement corrigées par version restera inchangé.

Toutefois il est certain qu'il n'est pas possible de s'engager sur un périmètre utilisateur non défini avec de fortes exigences en terme de réactivité.

3.4 Au service des contributeurs

Une communauté est riche des contributions qu'elle est capable de recevoir. Ainsi une activité essentielle consiste à qualifier et intégrer les contributions des acteurs à la périphérie de la communauté. La nature des contributions possibles est extrêmement vaste, depuis la traduction d'interfaces, la réalisation de supports pédagogiques, la rédaction de guides divers et évidemment le développement de correctifs ou de nouvelles fonctionnalités sous forme de code source. Pour ces dernières une revue de code permettant de contrôler la qualité, le respect des normes de développement et le respect de l'architecture devra être menée. Réalisée par un développeur chevronné disposant d'une longue proximité avec l'application, l'intégration de la contribution devra s'inscrire dans une trajectoire, dans une stratégie d'ensemble.

[9] Traiter vos utilisateurs en tant que co-développeurs est le chemin le moins semé d'embûches vers une amélioration rapide du code et un débogage efficace. [1]

[10] Effectivement, à la limite une application non utilisée n'a pas de bug !

Généralement les administrations comme nous l'avons dit ne disposent pas en interne, d'expertise en matière de développement, capable de trancher au delà de l'intérêt fonctionnel, sur les qualités techniques d'un reversement. Mais même, dans le cadre d'un marché de maintenance, prenant en charge l'expertise technique des reversements l'administration ne risque-t 'elle pas d'être évincée du pilotage effectif de l'application[11] ? Ne se peut-il pas que le titulaire en charge de cette mutualisation, du fait de sa position privilégiée, n'oriente au profit d'un tiers certains choix techniques ? Enfin le titulaire ne peut-il pas se recommander de sa position privilégiée pour acquérir une position hégémonique sur l'offre de support, à la manière de certains éditeurs Open Source ?

3.5 Piloter la communauté

Généralement peu formalisés, les modes d'organisation des communautés Open Source sont variables, fruits d'une construction collective et d'une histoire.

Faiblement hiérarchiques, les organisations vont de la démocratie directe plus ou moins consensuelle (communautés des projets de l'Apache Software Fondation) au despotisme éclairé (Communauté Linux avec Linus Torvald). Ensuite un principe cardinal est le mérite. Ainsi la trajectoire d'une personne est entièrement gouvernée par la reconnaissance de ses pairs au sein de la communauté. La conséquence de cela est que les membres de la communauté occupent leur position à titre personnel.

La communauté constituée par l'administration qui mutualise sera composée, en son coeur, d'agents en situation de subordination directe et surtout de salariés subordonnés à leur employeur mandaté via un marché public sur l'application. Ce type de situation réduit la circulation au sein de la communauté et donc son ouverture.

Un tel contexte est courant chez les éditeurs Open Source et l'on entend parfois le reproche que les applications qu'ils éditent ne sont pas vraiment Open Source. Ce que l'on peut parfois reprocher c'est effectivement le manque d'ouverture des communautés qui entourent les applications, mais le caractère Open Source de l'application reste entièrement déterminé par la licence accompagnant la diffusion.

Pour finir, les différents modes d'engagement des uns et des autres au sein de la communauté seront sources de conflits d'intérêts. Il n'est pas certain que la clarté de la situation de chacun suffise à limiter les risques de conflit. Une part de la gouvernance communautaire devra probablement être réglée contractuellement.

4 Proposition pour l'organisation de la mutualisation

Partant des hypothèses présentées au chapitre 2 et des difficultés à transposer le modèle Open Source, nous allons proposer pour une administration souhaitant mutualiser une application informatique au moyen d'un marché d'assistance, un mode opératoire réalisable.

[11] C'était le sentiment exprimé par le chef de projet de Spip Agora, porté par le secrétariat général du gouvernement.

4.1 La publication de l'application

Avant de recourir à un marché d'assistance il faut au moins publier les codes sources, les binaires et le corpus documentaire. Cette activité ponctuelle est relativement facile à mener, par l'administration elle-même[12]. Elle est un préalable au marché d'assistance, afin de permettre aux soumissionnaires du marché d'assistance de calibrer leurs offres sur les éléments tangibles qui auront été mis en ligne et dont ils auront la charge.

4.1.1 Phase de préparation de la publication

Le premier travail consistera à inventorier tout ce qui peut toucher de près ou de loin l'application à publier. Il s'agira des codes source, des scripts de construction et de tests de l'application, la base documentaire technique et fonctionnelle, les guides d'installation et d'exploitation, les guides utilisateurs et administrateurs, etc.

Ensuite il sera indispensable de choisir un nom de projet qui « sonne bien » en évitant les acronymes imprononçables qui peuplent les systèmes d'information des grandes organisations. On veillera aussi à ne pas empiéter sur une marque déposée.

Pour continuer il faudra réaliser une cartographie (rapport de conformité) en matière de propriété intellectuelle des éléments inventoriés afin de déterminer les licences libres les plus adéquates à une publication[13]. Cette tâche aboutira à la sélection d'une ou de plusieurs licences de logiciels libres et de licences documentaires.

Enfin sur la base des choix en matière de licence il faudra procéder à l'apposition, sur l'ensemble des éléments à publier tant code source que document, des entêtes portant mention des licences et copyright. Les diverses documentations seront consolidées dans un format normalisé.

Attention de prendre bien soin d'expurger les documents de tout élément confidentiel et d'écarter de la publication les éventuels jeux de tests constitués de données réelles !

4.1.2 Phase de publication

La phase de publication commence par la création du projet sur la forge à logiciels et l'ouverture de l'ensemble des services.

Voici la liste des opérations à réaliser pour cette phase :

- publication du code source dans un dépôt versionné de type CVS ou préférablement Subversion ;
- publication des paquets binaires de l'application ;
- publication de la base documentaire ;
- annonce d'un nouveau projet sur la forge.

[12] Néanmoins un marché d'accompagnement à la mutualisation sur AdmiSource prenant en charge la publication de projets portés par l'administration a été mis en place par la Direction Générale de la Modernisation de l'État.

[13] « Guide de choix et d'usage des licences de logiciels libres pour les administrations »

4.2 Marché d'assistance à la mutualisation

Il ne s'agit pas de proposer ici de « CCTP type »[14], car il est probable que notre approche devra être aménagée aux particularités de l'organisation et de l'application à mutualiser. Notre propos est donc de présenter dans les grandes lignes les activités à prévoir et leurs articulations afin de prévenir les difficultés de la mutualisation.

Voici quelques remarques générales, avant que nous n'entrions dans la description des activités du marché d'assistance à la mutualisation.

Tout d'abord le marché devrait être conclu pour la durée maximale autorisée (soit quatre ans, pour un marché de maintenance contenant une activité de support évolutif) ; nous avons vu, au chapitre 3.1 combien la question de la continuité est importante pour le développement de la communauté.

Ensuite le bénéficiaire[15] de l'appel d'offres est bien l'administration à l'initiative du marché. C'est à elle que le titulaire devra rendre des comptes, même si indirectement le cercle des bénéficiaires s'étend potentiellement à la communauté des utilisateurs de l'application.

Enfin il faudra désigner contractuellement la forge à logiciels que le titulaire devra exclusivement utiliser pour l'ensemble du support à la mutualisation. Nous avons déjà pointé les forges AdmiSource, SourceSup ou celle de l'Adullact, chacune identifiée et outillée pour l'herbergement des projets Open Source des administrations. Un compte administrateur relatif au projet sur la forge sera attribué au titulaire du marché, même si ce n'est pas absolument nécessaire.

Pour ce qui est des clauses de propriétés sur les travaux que le titulaire sera amené à réaliser dans le cadre du marché, il convient de prévoir dans le CCAP[16], le transfert des droits patrimoniaux au bénéfice de la personne publique. Il n'est pas utile de pointer contractuellement une licence car ce n'est pas le prestataire qui réalise le versement sous licence libre, mais l'administration bénéficiaire des travaux[17].

Voici, pour la mutualisation de l'application, les activités confiées au titulaire :

- activité d'animation ;
- activité de support correctif ;
- activité de support évolutif ;
- activité de support d'intégration.

Nous utiliserons à rebours la méthode BRR, « Business Readiness Rating for Open Source »[18], pour quantifier nos exigences concernant les activités d'animation et de support correctif. Cette méthode est normalement utilisée pour évaluer la maturité d'applications Open Source. D'autre méthodes similaires existent, OSMM-Capgemini, OSMM-Navica et QSOS, mais elles ne disposent pas d'un modèle

[14] CCTP : Cahier des Clauses Techniques Particulières

[15] L'entité à l'initiative et au profit de qui le marché est ouvert.

[16] CCAP : Cahier des Clauses Administratives Particulières

[17] « Guide de choix et d'usage des licences de logiciels libres pour les administrations »

[18] www.openbrr.org

de notation stricte. Par exemple, pour notre utilisation, la méthode indique qu'un temps d'installation de l'application est acceptable si il est compris entre 30 minutes et une heure. Nous exigerons alors que le temps d'installation soit inférieur à une heure.

Nous allons maintenant décrire chacune de ces activités.

4.2.1 Activité d'animation

Il s'agit d'une activité forfaitaire qui se déroulera sur l'ensemble de la durée du marché. Elle doit aider à constituer la communauté des utilisateurs (Cf. 3.2).

Une des premières tâches sera de réaliser (ou de compléter) un site web (idéalement trilingue) dédié au projet et présentant une identité visuelle sobre et élégante. Sans énumérer précisément les pages attendues, ce site devra décrire fonctionnellement le projet, proposer une FAQ, donner des nouvelles sur l'avancement du projet, fournir des captures d'écran, et permettre un accès simple vers l'espace documentaire et de téléchargement de la forge. A chaque fois que nécessaire le site devra être mis à jour.

Le titulaire postera des annonces sur les sites de référencement communautaire comme Freshmeat[19], sur l'observatoire Open Source de l'IDABC[20], etc. De même il ne faudra pas oublier les médias des communautés Open Source, comme (linuxfr, framasoft, slashdot...).

Le travail d'animation de la communauté, en direction des utilisateurs, s'appuiera sur un seul média afin de ne pas éparpiller l'effort de communication. On préférera le forum, plus facile à consulter pour les nouveaux utilisateurs que les listes de diffusion aux travers de leurs archives.

Le support fourni aux utilisateurs au travers du forum, couvrira les questions d'usage, de déploiement et d'exploitation. Les questions liées au développement seront plutôt routées vers une liste de diffusion dédiée. L'effort que le titulaire consacrera à l'animation du forum pourra se limiter à la prise en charge d'une demie-douzaine de messages par jour ouvré. A un tel niveau, le forum serait vraiment très actif et des utilisateurs avancés devraient prendre en charge une part du support[21].

Les questions récurrentes identifiées sur le forum donneront lieu au fil de l'eau à la constitution d'une FAQ, qui sera publiée (dès qu'elle comportera plusieurs items) sur le site dédié au projet.

Le titulaire qualifiera, sur la version courante dans un délai de sept jours, anomalies et évolutions remontées dans le gestionnaire de suivi ou depuis le forum utilisateur. Il faudra par exemple re-qualifier un ticket d'anomalie en demande d'évolution, fermer la demande d'une évolution déjà implémentée (et mettre à jour la FAQ)...

Les délais pour simplement qualifier les nouveaux tickets du gestionnaire de suivi sont longs mais conforme à ce que l'on observe sur les communauté Open Source (Cf. 4.2.3). Nous verrons dans les chapitres suivants la suite qu'il conviendra que le titulaire apporte aux anomalies et demandes d'évolutions du gestionnaire de suivi.

4.2.2 Activité de reprise

Pour toute maintenance applicative, la première activité pour le titulaire sera de s 'approprier l'application fonctionnellement et techniquement. Cette activité est comme la précédente forfaitaire.

Pour cela il pourra procéder à des opérations de refactorisation[22] afin d'assainir et de renforcer l'application et ainsi minorer la charge du support correctif. La contrainte étant qu'elles ne réduisent pas le périmètre de l'application. La publication préalable sur une forge du code source et de la documentation aura facilité durant la phase d'appel d'offre, l'évaluation de la charge de cette activité.

Le travail de reprise s'intéressera aussi à simplifier, si nécessaire, les procédures d'installation. L'objectif est de disposer d'une installation basique, en moins d'une heure[23], avec au plus un fichier de paramètrage centralisé (Cf. 3.2).

Ce travail de reprise pourra donner lieu à la mise en ligne d'une nouvelle livraison du logiciel.

Dans un délai de trois mois suivant le lancement du marché, le titulaire devra fournir un guide développeur reprenant et complétant les règles de développement, et d'architecture utilisées pour le développement. Il devra aussi indiquer comment contribuer (Cf. 3.4). C'est sur la base des critères et recommandations énoncées dans le guide développeur que seront prises les décisions techniques d'intégrer ou non des correctifs ou des évolutions fournis par des tiers de la communauté. Une fois validé par la personne publique, le guide développeur sera publié sur l'espace documentaire de la forge.

4.2.3 Activité de support correctif

Cette activité forfaitaire va courir sur la durée du marché. Elle consiste pour le titulaire à garantir que le gestionnaire de suivi ne contiendra jamais plus de dix anomalies ouvertes et qu'aucune anomalie ne restera ouverte plus de deux mois. Toutefois il ne sera pas tenu de corriger les anomalies non reproductibles sur la dernière version stable.

L'intégration de patch correctif sur la dernière version stable entrera dans le forfait de l'activité. Toutefois le titulaire sera libre d'intégrer le patch proposé ou de s'en inspirer s'il juge celui-ci d'une qualité insuffisante.

Enfin, tous les trimestres, le titulaire mettra en ligne une nouvelle livraison de l'application[24]. Si la livraison n'intègre que des correctifs, seul le dernier numéro sera incrémenté (ex. 1.2.3 vers 1.2.4), sinon on incrémentera le numéro mineur (1.2.3 vers 1.3.0).

Dans ces limites le prestataire organisera librement son activité corrective.

---

[19] freshmeat.net

[20] http://ec.europa.eu/idabc/en/chapter/5649

[21] Soit une douzaine de messages par jour ce qui donne environ 200 messages par mois. Ce rythme témoigne d'une activité acceptable selon la méthode BRR.

[22] Reprise du code source afin d'en améliorer la lisibilité, l'organisation et l'architecture sans modifier le périmètre fonctionnel.

[23] Une durée d'installation entre 30 minutes et 1 heure est jugée satisfaisante selon BRR.

[24] Le rythme nominal selon la méthode BRR est de 4 livraisons par ans dont deux intégrant des évolutions.

Les exigences en terme de réactivité pour une maintenance corrective sont très souples et loin des standards industriels. En fait elles se veulent conformes à ce que l'on peut observer au sein d'une communauté Open Source[25]. Cette faible exigence en terme de réactivité doit compenser l'ouverture à tous les utilisateurs du bénéfice du support correctif (Cf. 3.3). Enfin le support correctif est complémentaire d'une éventuelle TMA beaucoup plus réactive comme celle portée par le marché de support au logiciel libre de la DGI[26].

4.2.4 Activité de support évolutif

Cette activité sera classiquement déclinée en unités d'oeuvre et déclenchée au moyen de bons de commandes. Décomposée en plusieurs niveaux de complexité, avec des délais de livraison assortis, les unités d'oeuvre seront décrites afin de permettre au titulaire d'évaluer la charge afférente à chacune.

Le bénéficiaire du marché commandera sur la base des éléments dans le gestionnaire de suivi, les évolutions qu'il estime prioritaires. Une fois le bon de commande émis le titulaire mettra à jour le plan d'évolution de l'application (la roadmap).

La livraison pour recette sera mise en ligne en version candidate « release candidate » sur le site communautaire. Une fois la recette prononcée par le bénéficiaire du marché, le titulaire procédera à la publication de la nouvelle version à la prochaine échéance trimestrielle (en renommant la dernière version candidate).

4.2.5 Activité d'intégration

Cette activité a pour objet d'intégrer et de publier les contributions de membres de la communauté après une phase de validation. Elle répond aux exigences présentées au chapitre 3.4. Cette activité sera réglée au moyen de bons de commande.

Les contributions de type documentaire ne supposent pas de travail spécifique autre qu'une relecture critique ; cela peut être délégué à la communauté elle-même. Par ailleurs nous avons vu que l'intégration de patchs correctifs entrait dans l'activité de support correctif. Reste finalement au titre de l'activité d'intégration les évolutions fonctionnelles, sous forme de code source.

Le reversement sera d'abord instruit sur le plan fonctionnel par le bénéficiaire du marché et validé si il est jugé conforme à l'esprit de l'application et à ses buts.

Le titulaire procédera ensuite à une expertise technique vérifiant que le patch concerne bien la version stable courante et est techniquement recevable au regard des règles de développement et d'architecture. Des aller-retours pourront se faire entre le titulaire et les auteurs du patch, via une liste de diffusion dédiée aux développeurs pour des échanges techniques. En cas de refus persistant, le titulaire sera tenu de démontrer en quoi le patch n'a pas atteint le niveau de conformité attendu auprès du titulaire du marché.

Une fois la décision d'intégration entérinée, le bénéficiaire du marché va émettre un bon de commande pour que le titulaire intègre le patch. La charge de travail demandée pour l'intégration sera évaluée en unité d'oeuvre. Pour cela une technique relativement ancienne et simple consistera a se baser sur une estimation de l'effort de développement (en jour/homme) au moyen d'un calcul du COCOMO simple[27] sur la fraction de code source introduite ou modifiée par le patch [5][6]. Cette méthode simple et outillée[28] devra pour donner de bons résultats être calibrée sur l'application. Dans notre cas ce sera facile ; il suffira pour cela de confronter la charge évaluée à la charge constatée du développement de l'application.

L'effort d'intégration du patch sera alors fixé à 30% de l'effort estimé de son développement. Ce coefficient représente en fait la décote moyenne constatée entre la réutilisation d'un code et un re-développement complet [7]. Procéder de la sorte majore probablement l'effort d'intégration du patch, d'emblée conçu pour l'application. Cela laissera des marges pour les tests de non régression.

Une fois la nouvelle livraison stable publiée, le titulaire est tenu d'assurer sur les nouvelles fonctionnalités le même niveau de support que sur l'application initiale.

Le présent montage ne répond pas forcément à l'ensemble des objections avancées en matière de conflits d'intérêts (Cf. 3.5). Une solution originale mise en place, pour le marché de support de l'application Spip-Agora a consisté à interdire contractuellement au titulaire du marché, arbitrant sur le plan technique la mutualisation, d'intervenir dans le développement proprement dit de l'application et cela pour quels clients que ce soit.

4.2.6 Rôles et responsabilités

Le tableau suivant présente les tâches et les responsabilités associées à chacune des catégories de membres de la communauté, utilisateur, développeur, « commiteur » et administrateur.

---

[25] Un taux de plus de 75% des anomalies corrigé sur 6 mois est jugé excellent par la méthode d'évaluation BRR.

[26] Le marché de support logiciel libre de la DGI exige des délais de résolutions de 48h.

[27] COnstructive COst Model est une méthode permettant d'estimer l'effort de développement à partir du nombre de ligne de code .

[28] Le logiciel libre Sloccount (http://www.dwheeler.com/sloccount/) permet le calcul du COCOMO simple.

Utilisateurs

- signaler des anomalies
- proposer de nouvelles fonctionnalités
- aider les utilisateurs moins avancés

Développeurs

- programmer des patchs évolutifs et correctifs
- discuter les choix techniques
- rédiger des documentations
- tester les prochaines livraisons

Commiteurs

- qualifier techniquement les patchs des développeurs (revue de code)
- intégrer les patchs dans le dépôt versionné

Administrateurs

- collecter et qualifier les nouvelles fonctionnalités
- définir la feuille de route pour les prochaines livraisons

Les tâches d'administrateurs sont réservées au bénéficiaire du marché. Ensuite les tâches de « commiteurs » sont le fait exclusif du bénéficiaire du marché et du titulaire. Enfin les autres tâches sont ouvertes à l'ensemble de la communauté.

5 Conclusion

Pour terminer, voici quelques points de repères permettant d'évaluer le retour sur investissement d'une démarche de mutualisation comme celle que nous venons de décrire.

Sur l'ensemble du marché de maintenance, l'effort consenti pour les activités de reprise, de support correctif et de support évolutif, est identique avec ou sans mutualisation. Cet effort représente au moins 50% du coût du logiciel sur l'ensemble de son cycle de vie [6].

Le surcoût de la démarche va essentiellement peser sur l'activité d'animation nécessaire pour enclencher la mutualisation. Mais en conséquence de celle-ci, l'activité de support d'intégration va se substituer à une part du support évolutif. Une économie de 70% du support évolutif ainsi substitué sera généré.

Finalement la mutualisation sera viable pour une administration si le coût de l'animation est compensé par la valeur des reversements. L'équilibre ne pourra être atteint que sur plusieurs années, lorsque la communauté sera suffisamment développée.

A l'échelle plus globale, la mutualisation va générer ce que l'on appelle en terme économique, une externalité positive[29], qui profitera à l'ensemble des acteurs publics ou privés. Difficile à chiffrer, on pourra tenter une estimation à minima au moyen d'un comptage des utilisateurs du support en ligne (Forum), pour en déduire les implantations de l'application.

[29] « Une externalité est le fait de modifier sans échange monétaire en bien ou en mal la situation d'un agent B par la production ou la consommation d'un agent A » Wikipédia.

Bibliographie

[1] Eric Raymond, Bob Young, *The Cathedral & the Bazaar*, O'Reilly, 2001

[2] Andrea Capiluppi et Patricia Lago et Maurizio Morisio, Evidences in the evolution of OS projects through Changelog Analyses. Dans ICSE'03 International Conference on Software Engineering, pages 19-24, Portland, Oregon, May 3-11, 2003

[3] Schaefer H, Metrics for optimal maintenance management. Proceedings Conference on Software Maintenance. IEEE Computer Society Press: Washington, 1985; 114–119.

[4] Jeffery Poulin, Measuring Software Reuse, Addison Wesley, 1997.

[5] Jianjun Deng et Tilman Seifert et Sascha Vogel, Towards a Product Model of Open Source Software in a Commercial. Dans ICSE'03 International Conference on Software Engineering, pages 31-37, Portland, Oregon, May 3-11, 2003

[6] Boehm BW, *Software Engineering Economics*. Prentice-Hall: Englewood Cliffs NJ, 1981.

[7] Eric S. Raymond. *The art of unix programming*. Addison-Wesley, 2004

[8] Karl Fogel, *Producing Open Source Software, How to Run a Successful Free Software Project,* O'Reilly, 2005