RMLL 2007 - Communautés
La gestion de communautés libres

Ir. Robert VISEUR
Assistant, FPMs
robert.viseur@fpms.ac.be

Amiens, 12 juillet 2007

ACADÉMIE UNIVERSITAIRE WALLONIE-BRUXELLES

Qui suis-je ?

- Robert VISEUR (*robert.viseur@fpms.ac.be*).
- Assistant, Faculté Polytechnique de Mons (Economie et Management de l'Innovation)
  - FPMs : doctorat sur les techniques de co-création et leur application à l'Open Source.
- Guideur technologique, CETIC
  - CETIC : expertise Open Source; recherches en qualité logicielle, dont projet européen Qualoss (qualité des logiciels Open Source).
- Fondateur du site LogicielLibre.Net.

Objectifs de la recherche

- Recherche menée dans le cadre d'une thèse de doctorat portant sur le « management de la co-création ».
- Question: « Comment gérer un projet Open Source en communauté ? »
- But: disposer d'un premier ensemble de bonnes pratiques en matière de gestion de projets Open Source.
- Public cible: d'abord les gestionnaires de TPE/PME envisageant d'investir dans l'Open Source.

Organisation de la recherche

- Quatre parties:
  - Revue de la littérature.
  - Etude de projets populaires (Apache, Mozilla, Eclipse,... + forks) sur base de sources publiques (articles scientifiques, documents techniques, jugements d'experts, analyses de chef de projets,...). But: identifier des facteurs de succès et d'échec d'un projet Open Source.
  - Création d'un questionnaire d'étude approfondie. Soumission à 6 chefs de projets francophones.
  - Synthèse des résultats d'enquêtes et première synthèse des propositions de bonnes pratiques.

Revue de la littérature (1/2)

- Trois types d'approche:
  - En terme de gestion de communautés virtuelles.
  - En terme de gestion de projets de co-création (voir plus de détails sur dia suivante).
  - En terme de gestion de projets Open Source (Raymond, Manley, Fogel,...).
- Justification:

  « Un projet Open Source est un type particulier de projet logiciel, faisant appel à des pratiques de co-création, dans le cadre d'une communauté virtuelle »

Revue de la littérature (2/2)

- En terme de gestion de projets de co-création.

  « Implication du client, traditionnel consommateur de valeur, dans la création de valeur »

  Donc: implication du client dans la conception, le développement, la promotion, la distribution ou encore le support des produits de l'entreprise.

  - Identification de risques.
  - Identification des moyens de gestion.
  - Étude d'approches spécifiques
    - Ex.: concept d' « utilisateur de pointe » (Prof. Von Hippel)

Cas pratiques (1/5)

- Cas pratiques:
  - Apache.
  - Eclipse.
  - Mozilla.
  - Plusieurs forks.
    - « Mécanisme de scission d'un projet Open Source en cas d'insatisfaction d'une partie de la communauté »
    - Xemacs, EGCS, Samba-TNG, Dokeos, eGroupware, Joomla, *Nuke, *BSD, Sodipodi, CentOS et Gforge.

Cas pratiques (2/5)

- Apache:
  - Effet indirect du *first mover advantage* (NCSA).
  - Lancé par des utilisateurs de pointe.
  - Architecture modulaire facilitant la mutualisation.
  - Forte structuration (Fondation), croissance de son portefeuille de projets par cooptation et par incubateur. Exemples: Nutch, PHP et Geronimo.
  - Procédures de gouvernance ouvertes.
  - Outillage collaboratif complet.

Cas pratiques (3/5)

- Eclipse:
  - Structure générique et modulaire.
  - Indépendance sous forme de Fondation.
  - Valeur circonstancielle ? (« *Eclipse the Sun* »).
  - Prise en complet dans l'EPL (licence Eclipse) de plusieurs aspects du patrimoine intellectuel dont les marques et les brevets.

Cas pratiques (4/5)

- Mozilla:
  - Echec de la première phase du projet (1998-2001)
    - Code libéré trop compliqué à modifier.
    - Partage des capacités de développement sur la branche propriétaire et la branche Open Source. D'où: dispersion des ressources et démotivation des contributeurs.
  - Succès de la seconde phase du projet (2002-2007)
    - Réingénierie complète du projet.
    - Structuration de l'offre en ligne de produits (présentation familière pour les clients).
    - Plusieurs campagnes de promotion appuyées par la communauté.

Cas pratiques (5/5)

- Etude de forks
  - Raisons identifiées:
    - Fork lié à un arrêt de développement.
    - Fork lié à un manque d'ouverture du leader de projet.
    - Fork lié à une spécialisation du logiciel. Parfois évitable par une meilleure architecture.
    - Fork lié à des règles trop strictes (par exemple, règles de qualité bridant l'innovation).

Enquêtes (1/3)

- Réalisation d'un questionnaire basé sur la revue de la littérature et les études de cas préalables.
- Quatre domaines couverts (1):
  - Connaissance de l'environnement.
    - Connaissance des concurrents et partenaires potentiels, des besoins des clients; recherche de critères de différenciation.
  - Communication.
    - Gestion du nom et, plus largement, de l'identité; organisation du dialogue; politiques de communication des sources et des relations aux autres communautés.

Enquêtes (2/3)

- Quatre domaines couverts (2):
  - Organisation du processus de développement.
    - Incitants à l'innovation; gestion du processus de développement; prise en compte de la propriété intellectuelle, sélection et gestion des contributeurs.
  - Outillage.
    - Utilisation de trousses à outils pour l'innovation; d'outils classiques de travail collaboratifs.

Enquêtes (3/3)

- Questions supplémentaires sur les forks.
- Soumis à 6 chefs de projets Open Source francophones: eZ Publish, Claroline, Exo, Plume CMS, Ekiga, Jext.
- Nouvelles études en cours (**appel aux volontaires !**).

Synthèse des propositions de bonnes pratiques (1)

- Environnement
  - Débuter le projet Open Source par une analyse de l'environnement concurrentiel (clients, concurrents, etc). Raisonnement en termes commercial et communautaire (ex.: PDM clients / PDM contributeurs, concentration économique / concentration communautaire,...).
  - Dégager les orientations du projet, en terme de satisfaction d'un besoin. Base: demande client, support d'un standard, absence de solution libre,...
  - Dégager les critères de différenciation (langage, licence, concept, technologie,...). Attention aux conséquences du « régime d'inappropriabilité ».

Synthèse des propositions de bonnes pratiques (2)

- Environnement (suite)
  - Élaborer les premières pistes de collaboration (dans un but de création d'un environnement mutualiste).

Synthèse des propositions de bonnes pratiques (3)

- Communication
  - Réfléchir sur l'identité de la communauté (valeurs et symboles). Choix réfléchi des noms et logos. Dépôt de marque pour les entreprises commerciales (valorisation de la marque par la diffusion).
  - Choisir des règles de publication du code source. Publication régulière du code mais règles variées. Publication conditionnée à l'atteinte d'un niveau de qualité requis.

Synthèse des propositions de bonnes pratiques (4)

- Communication (suite)
  - Promouvoir la communauté. Acquisition d'utilisateurs, de contributeurs,...
  - Organiser un dialogue avec la communauté. Logique commerciale unilatérale souvent mal perçue. Communauté comme relais de la communication mais indépendance requise. Autres relais (ex.: PR, consortium).
  - Organiser les collaborations avec d'autres projets. Souvent réalisé dans un stade ultérieur du projet. Plusieurs formes de mutualisation: composants externes, sortie d'un composant générique, collaboration inter-société,...

Synthèse des propositions de bonnes pratiques (5)

- Développement
  - Gérer les incitants à l'innovation. Aspect reconnaissance. Rarement explicitement géré dans le Libre. Un exemple intéressant: tarif préférentiel en cas de reversement (B2B).
  - Prendre en compte la propriété intellectuelle. Soit choisir une (ou des) licence, et le modèle d'affaires associé.

Synthèse des propositions de bonnes pratiques (6)

- Développement (suite)
  - Tracer les contributions et s'assurer si nécessaire de la (co-)propriété des contributions.
  - Gérer le processus de développement, c-à-d:
    - Réaliser et mettre à jour un cahier des charges. Spécifier et mettre à jour les objectifs, la *roadmap*,... But: éviter les divergences ultérieures.
    - Proposer une architecture adaptée au public cible et propice à la mutualisation
      - Tout ne doit pas nécessairement être libre
      - Une bonne architecture favorise la mutualisation (« design évolutionnaire »).

Synthèse des propositions de bonnes pratiques (7)

- Développement (suite)
  - Gérer ... (suite)
    - Dériver une première implémentation.
      - Complémentarité de deux approches: « *Don't write code too early* » vs « *Distribuez tôt, mettez à jour souvent* ».
    - Suivre la qualité du projet. Révision par les pairs nécessaire mais non suffisante. Testing = 1 des rôles importants des éditeurs libres. Possibilité d'incitants comme organisation de *debug days*.
    - Gérer la connaissance (différents niveaux de documentations, etc).
    - Gérer les contributions
      - A ne pas surestimer. Souvent: soumission de bogues, traductions,... mais peu d'ajouts fonctionnels.

Synthèse des propositions de bonnes pratiques (8)

- Développement (suite)
  - Sélectionner et gérer les contributeurs
    - Le succès d'une démarche de valorisation par le Libre passe par la capacité à mettre en place un écosystème de mutualisation, avec des co-développeurs, des partenaires, etc. Démarche active souhaitable.
    - En co-création, il est connu que l'utilisateur produit des « choses infâmes ». Problème de la qualité de ses productions. Le logiciel libre y échappe-t-il du fait de la complexité de la technique informatique (barrière) ?
    - Intérêt des réunions physiques pour lier les membres, résoudre le problème de la distance, progresser sur des projets spécifiques. Importance des rôles d'animation, de contact à la communauté (nouvelles compétences).

Synthèse des propositions de bonnes pratiques (9)

- Développement (suite)
  - Question de la mesure du projet (tableau de bord). Voir évaluations qualités (ex.: Qualoss).

Synthèse des propositions de bonnes pratiques (10)

- Outillage
  - Envisager le développement de trousses à outils spécifiques. Voir *user toolkits* de Von Hippel. Rare dans le logiciel libre (une exception: environnement de traduction Ubuntu au sein du Launchpad).
  - Mettre en place ou adopter une plate-forme de co-développement: gestionnaire de sources, gestionnaire de bogues, wikis pour l'écriture de documentation, forums de discussion, messageries instantanées, outils de test,... (Gforge et assimilés + services en ligne: Sourceforge, Berlios, etc).
  - Évaluer l'intérêt de mettre en place des zones d'expérimentation propices à l'innovation.

Synthèse des propositions de bonnes pratiques (11)

- Forks
  - Différents moyens de prévention (cfr. Supra)
    - Adopter un style de gestion ouvert.
    - Soigner son architecture (ex.: *plugin*).
    - Permettre l'expérimentation d'alternatives.

Travaux dérivés et perspectives

- Sur base du questionnaire: proposition d'une grille d'audit donnant un cadre d'évaluation d'un projet Open Source.
- Évaluation automatique succincte de projets hébergés sur Sourceforge. Usage à des fins de sélection en vue d'une étude approfondie.
- Exploitation de ces résultats pour des projets de co-création en général.
- Réalisation en cours d'une nouvelle série d'études de cas par questionnaire.

Des questions?