La prise de décision dans les équipes de la communauté des logiciels libres : Faut-il mettre le bazar dans la poubelle ?

Lacolley Jean-Louis, Doctorant
IAE de Caen Basse Normandie

Loilier Thomas, Professeur
Université de Bretagne Occidentale

Tellier Albéric, Maître de Conférences – HDR
IAE de Caen Basse-Normandie
3, rue Claude Bloch, BP 5160, 14075 Caen cedex, France
Tel : +33 2 31 56 66 81
alberic.tellier@unicaen.fr

Résumé
Le nombre croissant de recherches, colloques et numéros spéciaux de revues sur le thème de la communauté des logiciels libres est révélateur de l'importance que prend cette dernière au sein des Sciences de Gestion. Notre article se focalise sur le processus de prise de décision au sein d'une équipe-projet de cette communauté.
L'intuition de départ, fondée sur notre connaissance de ce terrain et sur l'analyse de différents écrits sur la communauté, est que dans ces organisations les trois conditions nécessaires à l'application du modèle du garbage can de Cohen et al. (1972) sont susceptibles d'être remplies. Aussi, l'objet de cet article est, sur la base des décisions prises par une équipe-projet de développeurs « libres » dans un projet identifié a priori, de mieux comprendre ce processus de décision en utilisant certains éléments d'un modèle déjà existant : celui relatif au fonctionnement des organisations de type garbage can.
Le dispositif méthodologique repose sur l'analyse des mails échangés entre les développeurs d'un logiciel libre. Cette méthode permet d'étudier dans le détail les échanges dans leur forme d'origine, sur une durée longue et sans intervention du chercheur. Même si les échanges portent sur des développements terminés (les versions du logiciel ont bien été diffusées), il est ainsi possible d'avoir accès aux informations échangées au fur et à mesure du déroulement des actions, dans les conditions du « temps réel ».
Le projet étudié est le développement d'un logiciel de comptabilité développé par 29 membres. En employant comme matériau d'analyse les 2256 mails échangés par les développeurs entre le 25 mars 2004 et le 13 mai 2005, il est possible d'identifier douze situations de gestion dans lesquelles des problèmes, des solutions, des participants et des occasions de choix sont repérés.
Les situations de gestion sélectionnées sont ensuite analysées de manière systématique grâce à la grille des caractéristiques du fonctionnement de type garbage can établie par Huault et Romelaer

(1996). Ces caractéristiques présentent en effet l'avantage d'avoir été observées au moins une fois dans des organisations de type garbage can dans des recherches cliniques. Le contenu des différents mails a été traité par une procédure de triple codage afin de valider ou non la présence de ces caractéristiques.
Ce traitement des données permet de repérer les dimensions les plus fréquentes sur la période analysée et d'en tirer des enseignements sur la prise de décision au sein de ce type d'équipe virtuelle. En particulier, seulement deux situations très proches du mode garbage can sont repérées.
Finalement, cette recherche permet de valider le fait qu'une équipe-projet de la communauté des logiciels libres peut s'apparenter à une anarchie organisée, organisation type utilisant comme processus de prise de décision le garbage can model. Cependant, il ressort que ce mode de fonctionnement n'est utilisé que rarement. Dès lors, dans la dernière partie de l'article, nous proposons des pistes de réflexions théoriques pour caractériser les autres modes de décision utilisés.

Mots clés : décision, Garbage Can Model, open-source, équipe-projet, logiciel.

Le développement des logiciels libres interroge vivement la communauté académique. De nombreux chercheurs et équipes de recherche (notamment Lerner et Tirole, 2002 ; Lakhani et von Hippel, 2003 ; ou encore von Krogh et von Hippel, 2006), des revues scientifiques (voir par exemple les numéros spéciaux de *Research Policy* de 2003 et de *Management Science* en 2006) ont investi cet objet d'étude tant du point de vue de sa réussite commerciale, de ses conséquences sur les systèmes d'information que de ses fondements organisationnels. C'est sans doute sur ce dernier point que la production scientifique a été la plus abondante tant le modèle organisationnel du « monde libre » s'avère déroutant pour la théorie des organisations.

En particulier, la question lancinante de sa nouveauté est le point de départ de recherches fort instructives visant à circonscrire les caractéristiques de cette communauté pour en tirer un modèle global créant souvent un nouveau type dans telle ou telle classification. Par exemple, les travaux de Demil et Lecocq (2006) suggèrent que les projets *open source* relèvent d'une structure de gouvernance inédite, la *« bazaar governance »*, qui se fonde notamment sur un contrat très particulier (*« open license contract »*) appuyé par des incitations et un contrôle faibles.

De manière générique, l'étude des phénomènes nouveaux pose deux questions majeures. La première a trait à la place des faits dans l'analyse. En effet, un phénomène nouveau, et le modèle des logiciels libres n'échappe pas à la règle, génère souvent des « textes fondateurs » dans l'analyse des faits de gestion. Dans le cas du monde libre, ces textes sont en particulier ceux de Raymond (notamment *La Cathédrale et le Bazar* de 1998) qui décrivent le fonctionnement précis des équipes de développement de cette communauté. La thèse défendue par l'auteur est que ce modèle présente des caractéristiques de décentralisation et d'autonomie, d'anarchie (le bazar) qui l'opposent en tout point au modèle classique hiérarchique et structuré des équipes de développement de Microsoft (caractérisé par la métaphore de la cathédrale). Or ces textes, largement utilisés par la communauté académique, sont des interprétations plus ou moins stylisées de faits « bruts », observés ou vécus par les acteurs, donnant eux-mêmes souvent lieu à ré-interprétation(s) elles aussi plus ou moins stylisées. Or, la stylisation de faits déjà stylisés peut poser problème dans la mesure où les modèles généraux qui en découlent peuvent être très éloignés de ces faits originels sur lesquels ils sont sensés se fonder. Dit autrement, des stylisations successives peuvent provoquer une distorsion des faits qui aboutit au final à des interprétations

erronées[1]. Cela ne signifie nullement que les modèles généraux en question ne sont pas instructifs mais que leur validité interne ne peut se fonder (que) sur ces faits stylisés. Selon nous, il s'avère donc important de compléter ces faits stylisés par des données « brutes » attachées au fonctionnement concret des équipes projets de développement des logiciels libres.

La question de la nouveauté d'un objet d'étude pose en filigrane une autre interrogation : celle de la réponse conceptuelle devant être fournie pour comprendre la nouveauté empirique. Schématiquement, deux postures se détachent. La première consiste à choisir la symétrie en inventant de nouveaux concepts pour comprendre la nouveauté du phénomène. La seconde choisit de mobiliser dans le stock disponible des concepts déjà existants pour comprendre cette nouveauté empirique. Cet article s'inscrit dans cette seconde posture[2]. Précisément, nous cherchons à comprendre comment se prennent les décisions dans les équipes projets de cette communauté. Notre intuition, fondée sur notre connaissance de ce terrain, est que dans ces organisations les trois conditions nécessaires à l'application du modèle du *garbage can* de Cohen et *al.* (1972) sont susceptibles d'être remplies. Aussi l'objet de cet article est, sur la base de situations de gestion concrètes (des décisions prises par une équipe-projet de développeurs « libres » dans un projet identifié *a priori*), de mieux comprendre ce processus de décision en utilisant certains éléments d'un modèle déjà existant : celui relatif au fonctionnement des organisations de type *garbage can*.

D'un point de vue davantage épistémologique, notre attitude est de type « behavioriste ». En tant que chercheurs, nous considérons que *« tout ce qui se passe dans une organisation est le produit de comportements concrets d'individus concrets »* (Romelaer, 1994, p.49). Plus précisément, cette attitude s'inscrit dans la lignée de celle de March qui revendique un rapport au terrain assez particulier qui suit une séquence immuable : observation d'un terrain concret, identification des concepts clés pour mieux comprendre la situation de gestion étudiée et enfin modélisation. Cela signifie qu'il s'avère fondamental de tenir compte des déclarations des acteurs de la situation et des données formelles de l'organisation étudiée. Pour reprendre Romelaer (1994, p.50), *« l'intérêt de l'observation d'une situation concrète réside dans le fait qu'elle permet d'identifier des phénomènes non encore observés et d'y voir à l'œuvre des concepts dont les recherches*

[1] Plus généralement, la stylisation pose aussi le problème des conditions d'utilisation des données secondaires dans les sciences de gestion.

[2] Ce qui ne signifie en aucun cas que nous la considérons comme supérieure à la première.

précédentes n'avaient pas fait apparaître la pertinence ; la recherche se donne donc la possibilité de surprise et de progrès conceptuel ; en aucun cas il ne s'agit que de valider une hypothèse : il faut voir dans le terrain, dans la réalité et le concret des maîtres absolus ». Cette possibilité de « progrès conceptuel » est notamment due au fait que l'ambiguïté, centrale dans les organisations, génère souvent des fonctionnements particuliers qui, eux-mêmes, sont potentiellement sources de création conceptuelle par le chercheur.

Après avoir présenté le terrain étudié et montré l'intérêt de la mobilisation du modèle du *garbage can* pour mieux comprendre ses processus décisionnels, nous montrons que le contexte organisationnel (la communauté des logiciels libres) des situations de gestions choisies réunit les conditions d'application de ce modèle (ce qui constitue un « début de preuve »). Ensuite, des situations de gestion sont identifiées et analysées de manière systématique grâce à la grille des caractéristiques du fonctionnement de type *garbage can* établie par Huault et Romelaer (1996). Il s'agit donc de vérifier si ce modèle est effectivement utilisé. Cette analyse, dont la méthode est présentée, s'appuie en particulier sur le recueil systématique des 2256 mails échangés par les membres d'une équipe-projet entre le 25 mars 2004 et le 13 mai 2005. Enfin, ces résultats sont discutés et remis en perspective.

1. LE FONCTIONNEMENT DES EQUIPES *OPEN SOURCE* : POURQUOI MOBILISER LE *GARBAGE CAN MODEL* ?

La première partie de l'article revient sur le développement de la communauté *open source* et synthétise les principales recherches qui ont détaillé le fonctionnement des équipes de développeurs. Le contexte de la recherche étant présenté, les objectifs et le cadre théorique sont précisés.

1.1. PRESENTATION DE LA COMMUNAUTE DES LOGICIELS LIBRES ET DE SES EQUIPES PROJETS

1.1.1. Le développement des logiciels libres : quelques repères historiques

Apparus au milieu des années quatre-vingts, les logiciels libres mobilisent aujourd'hui plusieurs dizaines de milliers de programmeurs indépendants. L'analyse des liens unissant ces informaticiens géographiquement disséminés a conduit à retenir le terme de « communauté » ou de « réseau communautaire » pour caractériser ce modèle innovateur. L'histoire du logiciel libre est en fait indissociable de celle d'Unix, système d'exploitation développé à partir de la fin des

années soixante dans les laboratoires Bell, filiale d'AT&T (Logerot, 2003). Etant en situation de monopole dans le secteur des télécommunications, AT&T s'était engagée auprès du gouvernement fédéral américain à ne pas se développer dans l'informatique (machines et logiciels). C'est la raison pour laquelle, dès 1975, la société a offert aux universités et centres de recherche un accès gratuit au code source d'Unix.

En 1984, Richard Stallman, un ancien chercheur du laboratoire d'intelligence artificielle du M.I.T. (*Massachusetts Institute of Technology*), décide de développer un système d'exploitation totalement libre qui sera compatible avec Unix. Il baptise d'ailleurs son projet « GNU » pour « *Gnu is Not Unix* ». En 1985, Stallman crée la *Free Software Foundation* pour récolter des fonds pour le projet GNU. Son ambition est de proposer une alternative à la logique propriétaire qui est alors en train de s'imposer dans le monde de l'informatique. Il définit un nouveau type de licence, la *GNU General Public Licence* (GNU-GPL), dans laquelle le propriétaire du code source accorde aux utilisateurs le droit de le copier, le distribuer et le modifier. Pour garantir ces niveaux de liberté, ces logiciels sont « protégés » par le « *copyleft* » (l'opposé du *copyright*) qui interdit de dissimuler le code source du logiciel ainsi que celui de tous ses dérivés. Chaque utilisateur peut ainsi, dans une logique globale de création de valeur, améliorer le logiciel qu'il a reçu (gratuitement ou non) et ainsi « rendre » à la communauté ce qu'elle lui a « donné ».[3]

En 1990, bon nombre d'éléments constitutifs du système d'exploitation de Stallman sont réalisés mais des retards ont été pris sur la conception du noyau du système. Or, en 1991, Linus Torvalds, étudiant finlandais de l'université d'Helsinki, modifie le système Unix pour qu'il fonctionne sur des micro-ordinateurs. Une fois les lignes et les codes diffusés sur Internet, il reçoit rapidement des améliorations d'internautes programmeurs qu'il intègre à son système d'exploitation qu'il a baptisé Linux (le X est une référence à Unix). En mars 1994, Torvalds publie la version 1.0 du noyau Linux (Linux Kernel) avec une aide en ligne, bientôt suivie par d'autres versions comportant de plus en plus de fonctionnalités.

Progressivement, des programmes fonctionnant sous Linux vont être développés. La gamme des logiciels libres est aujourd'hui assez étoffée et il existe plusieurs sous-communautés distinctes

[3] Tout utilisateur d'un logiciel libre dispose de la possibilité de l'acquérir gratuitement (don) mais est aussi invité à fournir un contre don qui peut prendre au moins trois formes : contribuer à la pérennité du produit en augmentant mécaniquement le nombre de ses adeptes ; tester le produit et ainsi faire bénéficier l'ensemble de la communauté des utilisateurs de l'expérimentation ; concevoir des programmes informatiques libres, fondés sur le *copyleft*, qui viennent renforcer l'offre *open source* (Loilier, 2002).

indissociables de projets très clairement identifiés, parfois concurrents : projet OpenOffice (développement d'une suite bureautique concurrente de Microsoft Office), projet Apache (serveur Web libre), Sendmail (agent de messagerie)… Ces équipes-projets constituent la « communauté des logiciels libres ».

1.1.2. Le fonctionnement des équipes

Au sens strict, la communauté des logiciels libres est un ensemble d'équipes virtuelles globales (un « réseau d'équipes ») travaillant chacune sur des projets distincts s'inscrivant tous au sein d'un objectif partagé : la construction d'une offre « libre » intégrée en informatique (logiciels, serveurs, système d'exploitation…). Il s'agit donc d'équipes-projets constituées de développeurs indépendants disséminés géographiquement et, le plus souvent, non intéressés d'un point de vue commercial. Ils sont issus d'organisations et de cultures différentes, sont réunis temporairement (pour la durée du projet) et utilisent principalement les Technologies de l'Information et de la Communication (désormais TIC) comme support de communication et de mise en place du projet global.

Même si cette communauté a été qualifiée de modèle « bazar » (Raymond, 1998), l'analyse de son fonctionnement montre la présence d'unités de pilotage qui ont autorité sur le reste du réseau et, plus largement, de procédures d'évaluation, de contrôle et de sanction (O'Mahony, 2003). Les projets sont généralement pilotés par des administrateurs (en fait des programmeurs mainteneurs) en nombre limité qui contrôlent le développement et décident ou non d'intégrer les contributions nouvelles.

Le cycle de développement d'un logiciel libre suit généralement les étapes suivantes (Edwards, 2001). 1- Un « leader » (l'initiateur du projet) propose une version 0 d'un logiciel en précisant ses objectifs et en fournissant le code source. 2- Des « contributeurs » décident librement d'intégrer le projet (création d'une équipe virtuelle), téléchargent ces éléments et identifient les problèmes et les possibilités d'amélioration. 3- Les propositions sont adressées au leader et à l'équipe virtuelle *via* une *mailing list* dédiée. 4- Les corrections sont discutées, parfois évaluées par des procédures de vote en ligne. 5- Le leader introduit les solutions validées et propose une version 1 en téléchargement, un nouveau cycle débute.

Dans certains cas, le fondateur d'un projet délègue une partie de son autorité à différents administrateurs qui ont en charge le développement de modules liés. C'est ainsi que Linus

Torvalds s'est entouré de « lieutenants de confiance » (« *trusted lieutenants* ») pour concevoir différents *patchs* et assurer la sélection des propositions des membres de la communauté (Markus et *al.*, 2000, p. 22). Il y a dans ce cas une collection de « leaders » avec des zones de compétences clairement définies. L'autorité est cependant légitime et non statutaire. Officiellement, tout individu peut librement intégrer ou quitter une équipe de développement[4].

La taille des équipes de développeurs est généralement assez faible. Par exemple, dans la version 1.0 de Linux Kernel, 76,6 % des modules ont été développés par des équipes de moins de 10 personnes (Ghosh et David, 2003). La multitude des petites équipes est permise par la grande modularité du projet qui permet la spécialisation, le développement sans contrainte de coordination (Moon et Sproull, 2000) puisque c'est l'architecture générale du système qui assure la cohérence de l'ensemble. Si les listes de diffusion (« *mailing-lists* ») ont pour mission première d'informer les membres sur des questions techniques, elles jouent également un rôle prépondérant dans l'organisation des projets et la répartition des tâches. Au delà, des sites centralisent les projets et les classent en fonction du stade de développement. L'exemple le plus connu est celui de Sourceforge.net. Cette plate-forme offre notamment des outils de développement et gère des forums de discussion. En décembre 2006, 136 923 projets étaient référencés sur ce site et 1 460 827 utilisateurs enregistrés.

Les observations qui ont été menées sur le fonctionnement des forums de discussion et des *mailing-lists* (von Krogh et *al.*, 2003) montrent que les nouveaux entrants sont « observés » pendant quelques semaines, voire quelques mois, avant d'être véritablement intégrés aux discussions. Pendant cette phase d'évaluation, véritable processus d'adhésion (« *joining script* ») aux valeurs et habitudes de la communauté (Kim, 2000 ; Garcia et Steinmueller, 2003 ; Kuk, 2006), les participants doivent démontrer leurs compétences en offrant une contribution jugée significative, c'est-à-dire « donner » à la communauté avant de pouvoir « recevoir »[5].

[4] Les projets *open source* respectent en effet la règle de la « bifurcation » (« *forking* ») : tout individu en désaccord avec les décisions prises par l'administrateur peut créer un nouveau projet pour exploiter d'autres choix techniques (Raymond, 1999).

[5] Edwards (2001) distingue ainsi les « *learners* », nouveaux entrants qui souhaitent intégrer la communauté et les « *insiders* », membres qui participent activement aux projets et qui jouissent d'une réputation technique.

1.2. LA QUESTION DE RECHERCHE, LE CADRE THEORIQUE ET LE DISPOSITIF EMPIRIQUE

L'objectif de la recherche est de confronter un modèle classique de prise de décision, le *garbage can*, au cas du développement des logiciels libres. Dans un premier temps, les principales caractéristiques de ce modèle sont présentées. Il s'agit ensuite de montrer pourquoi la mobilisation de ce modèle semble pertinente dans le cas de l'*open source*. Enfin, le projet étudié est décrit.

1.2.1. Le *garbage can*, un modèle de prise de décision

Partant du constat que le modèle rationnel de prise de décision ne permet pas de rendre compte correctement de la façon dont bon nombre de décisions sont prises dans les organisations, Cohen et *al.* (1972) ont formalisé un modèle de prise de décision, le *garbage can model* (GCM désormais), sensé mieux rendre compte de la prise de décision dans les organisations dans lesquelles règne l'ambiguïté. Même si ce modèle concerne potentiellement toutes les organisations (à un endroit donné de la structure, à un moment donné de son histoire), il s'applique avant tout dans des collectifs qui remplissent trois conditions : des préférences incertaines, une technologie floue (les procédures ne sont pas bien comprises par l'ensemble des membres de l'organisation) et une participation fluctuante des acteurs. Dans de tels groupes ou situations, on parle alors d'anarchies organisées.

La décision n'est alors plus un processus séquentiel (de type problème-analyse-décision) mais la rencontre de différents flux : des occasions de choix, des solutions, des problèmes et des participants. La particularité du modèle réside dans le découplage (de très important à total) de ces quatre flux. A chaque occasion de choix (dans chaque poubelle) les participants présents (pour des raisons diverses) déversent à différents moments des solutions et des problèmes (qui se rencontrent ou ne se rencontrent pas). Du caractère très indépendant de ces flux découle un certain nombre de caractéristiques d'un collectif décidant selon le mode *garbage can* :

- le lien de causalité problème-solution laisse la place à un ordre plus temporel. Ce qui compte c'est que le problème et la solution se rencontrent au bon moment avec les bons participants porteurs de solutions. Cela signifie aussi que ce sont les solutions qui peuvent être à la recherche de problèmes et non seulement l'inverse.
- Un problème ne trouve pas forcément sa solution dès la première occasion de choix.

- Une décision dépend fondamentalement de sa structure de choix (qui y participe) mais aussi de sa structure d'accès (la relation entre problèmes et choix). Concrètement, les acteurs s'impliquent dans telle ou telle décision en fonction de son importance, du temps dont ils disposent, de la symbolique associée à cette décision (« il faut y être »).

De nombreuses observations empiriques ont validé ce modèle[6]. En particulier, Romelaer et Huault (1996) ont proposé une grille permettant de passer des trois conditions nécessaires caractéristiques des anarchies organisées à un ensemble de critères exhaustifs et opérationnels suffisants pour mettre au jour l'utilisation effective du GCM au sein d'un collectif organisé (Tableau 1).

Tableau 1. Caractéristiques de fonctionnement des organisations de type *garbage can*

Dimension	**Code**
1 : Une décision négative à un moment peut être positive lorsqu'une autre circonstance décisionnelle se présente.	Circonstance
2 : Une décision peut avoir plusieurs objectifs partiellement compatibles entre eux ; chacun de ces objectifs peut être poursuivi par d'autres moyens que ceux qui sont considérés lors de la décision.	Objectifs
3 : Chaque décision dépend de façon cruciale de la structure des contacts des acteurs.	Contacts
4 : L'entreprise utilise dans certains cas des routines de diagnostics ; ces routines sont parfois appliquées hors du contexte pour lequel elles ont été élaborées.	Routines
5 : Dans les décisions, certaines personnes interviennent en raison de leur fonction sans être nécessairement fortement impliquées.	Fonction
6 : Certaines personnes qui n'ont aucune raison formelle de s'impliquer dans une décision s'y impliquent néanmoins.	Implication
7 : L'intensité de la participation de chaque acteur dans une décision varie au fil du temps.	Participation
8 : La nature des données pertinentes dans une décision est incertaine ; peut aussi être vue de façons différentes par différents acteurs.	Données
9 : La signification des décisions est sujette à des interprétations controversées.	Controverses
10 : Dans une décision, la recherche de solutions s'effectue assez souvent par reprise de solutions connues, par recherche de solutions au voisinage de solutions connues, ou en faisant appel à des personnes se trouvant dans l'entourage immédiat des décideurs.	Voisinage
11 : Les préférences des décideurs dans une même situation, ou vis-à-vis de différentes situations, sont incertaines et/ou changeantes.	Préférences
12 : Dans une décision, le décideur ou les acteurs peuvent s'engager dans la décision sans préférence claire, pour explorer l'environnement et découvrir leurs propres préférences.	Engagement
13 : Dans certains cas, une décision est un lieu dans lequel s'investissent des partisans de solutions toutes faites à d'autres problèmes.	Solutions
14 : La génération d'alternatives dans une décision dépend de la structure d'accès à la décision : qui est informé, qui peut intervenir, qui doit intervenir.	Structure
15 : Tous ceux qui peuvent intervenir dans une décision n'interviennent pas nécessairement, parce qu'ils sont occupés ailleurs.	Intervention
16 : Les idéologies et les cultures sont des facteurs importants dans les décisions : déclaration de la direction générale, culture d'entreprise…	Idéologies
17 : La structure du champ externe de l'entreprise est également un facteur qui influe sur les décisions.	Champ

[6] Une présentation plus exhaustive du modèle et des différents essais de validation empirique est proposée par Romelaer (1994).

18 : Dans certains cas, une décision prise peut être remise en cause ; chaque décision isole une tranche temporelle dans un processus qui n'a pas, par nature, de début ni de fin clairement observable.	Processus
19 : Dans certains cas, une décision a essentiellement une fonction symbolique.	Symbolique
20 : L'implication dans une décision est aussi un signe de pouvoir ; certains acteurs voudront participer à une décision sans que la question traitée n'ait pour eux beaucoup d'importance.	Pouvoir
21 : Chaque décision est le produit de la rencontre, au moment même de la décision, de l'ensemble des problèmes, des solutions, des acteurs et des occasions de choix alternatives ; chaque décision dépend fortement des conditions locales qui existent là où elle est prise et au moment où elle est prise.	Rencontre

Source : D'après Romelaer P. et Huault I., 1996, p. 23.

1.2.2. Pourquoi mobiliser le *garbage can model* ? Indices et début de preuve

La Cathédrale et le Bazar (Raymond, 1998) constitue l'un des textes références du monde des logiciels libres, à la fois pour les développeurs de cette communauté et pour le monde académique qui le mobilise régulièrement. A titre d'exemple, dans le numéro spécial de *Management Science* de juin 2006 dédié à l'analyse de l'OSS (*Open Software System*), ce texte est cité dans neuf des onze articles qui composent ce numéro de la revue. Il s'agit aujourd'hui d'un texte quasi mythique qui, le premier, a décrit précisément le fonctionnement des équipes de développement de logiciels libres.

Son auteur, Eric Raymond, est l'un des développeurs les plus connus de cette communauté. Un article lui est consacré dans l'encyclopédie libre Wikipédia. D'un point de vue méthodologique et épistémologique, son article s'appuie sur un projet qu'il a lui-même lancé, Fetchmail (un projet de gestion de messagerie), afin de tester certaines théories et certains fonctionnements surprenants du génie logiciel suggérés par l'histoire de Linux. L'auteur se situe donc dans une posture délibérée de chercheur-acteur qui provoque les évènements afin de mieux comprendre une réalité[7] ou tout au moins un enchaînement d'évènements au sein d'un groupe d'acteurs. La lecture de ce texte montre qu'il mélange allègrement des faits plutôt bruts à des faits déjà interprétés et souvent stylisés (c'est-à-dire s'appuyant sur des représentations simplifiées). L'objectif de ce texte est de discuter de l'existence de deux styles de développement fondamentalement différents, le modèle « cathédrale » de la majorité du monde commercial, à opposer au modèle « bazar » du monde de Linux et de montrer que *« ces modèles dérivent d'hypothèses opposées sur la nature de la tâche consistant à déboguer un logiciel* » (Raymond, 1998, p.1).

[7] Si Eric Raymond faisait partie de la communauté académique des chercheurs en sciences sociales, il se définirait sans doute lui-même comme un constructiviste qui provoque des évènements pour mieux interpréter de fonctionnement d'un collectif, collectif dont il fait lui-même partie.

L'analyse systématique de ce texte permet de repérer l'existence des trois conditions d'existence du GCM. Le tableau suivant présente dans le détail les verbatims issus du texte utilisés comme outils de repérages de ces conditions.

Tableau 2. Mise en évidence des conditions d'existence du GCM dans les équipes de développement des logiciels libres.

Conditions d'existence du GCM	Repérages de la présence de ces conditions grâce à l'analyse du texte de Raymond (1998)
C1 : Les préférences sont incertaines : ou bien le décideur a des préférences floues ou changeantes, ou bien il a des préférences peu cohérentes entre elles, ou encore il y plusieurs décideurs qui ont des préférences peu cohérentes entre elles.	• Préférences changeantes des acteurs *« Après avoir envoyé à Carl Harris (*) mon premier lot de modifications apportées à popclient, j'ai (**) découvert qu'il avait pratiquement perdu tout intérêt pour popclient depuis quelque temps »* p. 4. • Préférences floues *« Je ne pense pas qu'on puisse planifier tel ou tel but. On y est attiré par des idées de conception si puissantes qu'après coup les résultats paraissent tout naturels, inévitables, prédestinés. La seule manière de tomber sur ce genre d'idées est d'avoir beaucoup d'idées – ou d'avoir assez de discernement pour reconnaître les bonnes idées des autres, même si elles vous mènent plus loin que vous ne pensiez aller »* p.12. • Bifurcation du projet *« Le projet prit un jour radical le jour où Harry Hochheiser m'envoya un premier jet de son code destiné à faire suivre le courrier. J'ai réalisé pratiquement tout de suite qu'une implantation fiable de cette fonctionnalité rendrait pratiquement obsolètes tous les autres modes de distribution du courrier »* p.10. • Préférences des acteurs peu cohérentes entre elles *« Pour rendre Fetchmail aussi bon que j'entrevoyais qu'il pouvait l'être, il ne faudrait plus me contenter d'écrire du code répondant à mes propres besoins, il me faudrait prendre en compte les désirs et les souhaits des autres pour des fonctionnalités que je n'utiliserai pas »* p.12. *« Quelques utilisateurs européens m'ont ennuyé jusqu'à ce que j'ajoute une option pour limiter le nombre de messages transférés à chaque session. J'ai mis du temps à m'y résoudre et aujourd'hui encore je n'en suis pas très content »* p.13. *« Certains utilisateurs de Fetchmail m'ont demandé de modifier le logiciel de manière à stocker des mots de passe chiffrés dans le fichier rc, de telle sorte que les petits malins ne puissent les lire directement. Je ne l'ai pas fait car cela n'apporte aucune protection supplémentaire »* p.14-15.
C2 : La technologie est floue : les liens entre les actions et les « sorties » du processus décisionnel sont mal connus ; les conséquences d'une action ne peuvent être cernées qu'avec une grande imprécision.	• Réutilisation d'une solution à une autre fin « *Linus Torvalds, par exemple, n'a pas essayé d'écrire Linux en partant d'une page blanche. Au contraire, il a commencé par réutiliser le code et les idées de Minix, un minuscule système d'exploitation ressemblant à Unix tournant sur les 386. La totalité du code de Minix a été abandonnée ou réécrite complètement depuis – mais tant qu'il était là, ce code a servi de tuteur à l'enfant qui deviendrait Linux »* p.3 *« Dans le même esprit, je me suis mis en quête d'un utilitaire POP raisonnablement bien écrit, afin de l'utiliser comme base pour mon développement »* p.3 • Apprentissage par essai-erreur *« Ou, dit autrement, on ne comprend souvent vraiment bien un problème qu'après avoir implanté une première solution. La deuxième fois, on en sait parfois assez pour le résoudre correctement. Ainsi, si vous voulez faire du bon travail, soyez prêt à recommencer au moins une fois »* p.4. *« On a souvent réécrit les idées et les prototypes des modes trois ou quatre fois avant d'atteindre une forme finale stable »* p.5. • Dissociation découverte/compréhension d'un problème *« En général, il n'est pas nécessaire que la personne qui comprenne un problème soit la*

	personne qui l'ait identifié. Quelqu'un trouve le problème dit Linus Torvalds et c'est quelqu'un d'autre qui le comprend. Je (Linus Torvalds) pousserai le bouchon jusqu'à dire que le plus difficile est de trouver le problème » p.7
C3 : La participation est fluctuante : les acteurs d'une situation de choix y ont une implication variable dans le temps et en intensité ; la délimitation de ceux qui peuvent intervenir dans le processus décisionnel n'est pas précise ; la décision est une scène dans laquelle les « acteurs » entrent et de laquelle ils sortent assez librement.	• Changement de leader d'un projet *« Quand un programme ne vous intéresse plus, votre dernier devoir à son égard est de le confier à un successeur compétent – 5ème leçon mise en évidence par Raymond du mode Bazar »* p.5 *« Sans même avoir à en discuter Carl et moi savions que nous poursuivions un objectif commun : chercher la meilleure solution. La seule question d'importance pour chacun de nous était de nous assurer que le programme serait en de bonnes mains avec moi. Une fois que j'en apportai la preuve, il me fit place nette avec bonne volonté. J'espère agir de même quand mon tour viendra »* p.5. • Implication variable des acteurs *« Et les exemples de collaborations occasionnelles rendues possibles par l'Internet, à la manière de Linux, ne manquent pas »* p.5. • Augmentation au cours du projet du nombre des acteurs *« Ainsi, introduire de nouveaux bêta-testeurs ne va sans doute pas réduire la complexité du bogue le plus profond du point de vue du développeur, mais cela augmente la probabilité que la trousse à outils d'un bêta-testeur sera adaptée au problème de telle sorte que le bogue saute aux yeux de cette personne »* p.8.

(*) : Carl Harris est le leader initial du projet, celui qui l'a initié en lui donnant comme nom « popclient ».
(**) : la première personne du singulier concerne bien entendu l'auteur lui-même. Dans le cas contraire, le nom de la personne concernée est précisé entre parenthèses.

L'analyse du texte de Raymond montre qu'une équipe de développement libre peut être assimilée à une anarchie organisée. Il s'agit donc bien d'un début de preuve d'existence du modèle *garbage can* au sein d'équipes « libres » puisque Cohen et *al.* (1972) ont montré que dans de telles organisations ce modèle explique fréquemment mais pas automatiquement les décisions prises. Autrement dit, les conditions C1, C2 et C3 sont des conditions nécessaires mais pas suffisantes pour statuer sur la présence effective de décisions de type GCM. Comment passer d'un début de preuve fondé sur des faits stylisés à une preuve émanant, quant à elle, d'un champ empirique d'investigation ? La réponse à cette question est donnée par l'approche behavioriste de March, déjà évoquée dans l'introduction de cette recherche. Celle-ci se fonde notamment sur l'observation de situations concrètes dans lesquelles les acteurs agissent là encore de manière concrète. Sans l'appui de telles observations, il nous apparaît difficile, pour ne pas dire impossible, de savoir si oui ou non il existe des situations dans lesquelles le modèle *garbage can* explique la prise de décision. La preuve irréfutable ne peut être fournie que par la mise au jour de processus de type *garbage can* dans une équipe de développement au sein de situations concrètes. Ces situations concrètes peuvent être assimilées à la notion de « situation de gestion » décrite par Girin (1983, 1990). Celle-ci *« se présente lorsque des participants sont réunis et doivent accomplir, dans un temps déterminé, une action collective conduisant à un résultat soumis à un*

jugement externe » (Girin, 1990, p.143). Cette définition appelle trois remarques. Tout d'abord, la notion de participant n'implique pas que le résultat poursuivi soit l'objectif ultime de chacun. Chacun agit à l'intérieur de certaines contraintes et ne dispose que de ressources limitées. Les participations peuvent être liées à une obligation, une condition ou encore une opportunité. Ensuite, la réunion des participants peut être physique mais aussi virtuelle comme dans le cas qui nous intéresse. Enfin, le résultat est constitué par la production des participants faisant l'objet du jugement. Il peut être extrêmement quantifié (un chiffre) ou n'être qu'une orientation générale.

Du point de vue du chercheur, la mobilisation de la notion de situation de gestion dans un projet de recherche doit tenir compte de deux points cruciaux. Premièrement, l'ensemble des situations de gestion auquel un collectif d'acteurs est confronté est le plus souvent hétérogène : des situations récurrentes, habituelles mais aussi des situations très spécifiques, parfois mêmes uniques. La sélection des situations de gestion étudiées est donc un aspect important du projet de recherche. En second lieu, l'analyse d'une situation de gestion doit être globale. Il n'est pas possible de séparer une situation de son contexte tant pour les acteurs eux-mêmes que pour le chercheur. D'abord parce que les contextes qui traversent les situations de gestion sont aussi des éléments porteurs de sens et de significations pour les acteurs eux-mêmes leur permettant d'adopter une conduite appropriée. Ensuite, parce que le chercheur ne comprendra pas correctement les phénomènes étudiés sans l'analyse du contexte, primordiale pour la validité interne du cas mais aussi pour sa validité externe (ou tout au moins pour le processus de généralisation analytique). Par ailleurs, parler du contexte au singulier est sans doute incorrect dans la mesure où plusieurs contextes de niveaux différents viennent agir sur la situation de gestion. Dans notre cas, on peut parler de trois niveaux de contexte distincts :

- un macro contexte qui concerne la culture du mouvement *open source*, ses origines, son histoire dans lequel les valeurs de solidarité, d'échange non marchand (logique du don/contre-don) et de liberté sont centrales.
- un méso-contexte composé du contexte propre au projet étudié et à l'équipe qui le mène. Ces deux premiers niveaux de contexte sont communs à toutes les situations de gestion analysées.
- enfin, un micro-contexte, qui est propre à chacune des situations de gestion étudiées.

Le premier niveau de concept a été décrit de manière détaillée dans cet article dans les sous-parties relatives à la communauté des logiciels libres et à ses équipes. Les deux autres contextes

sont présentés ci-après : le projet étudié (Grisbi) et les situations de gestion identifiées au cours de ce projet.

1.2.3. Sélection et présentation du projet étudié

Cette recherche porte sur l'étude d'une équipe projet dédiée à un logiciel libre de comptabilité baptisé Grisbi. Le choix de ce projet s'explique essentiellement par opportunisme méthodique (Girin 1989) : le fondateur du projet a accepté de répondre à nos questions tout au long de la recherche ; il est plus aisé de comprendre les échanges puisque Grisbi est un logiciel de comptabilité ; le référencement sur Sourceforge.net permet d'obtenir la totalité des mails entre développeurs.

C'est en Avril 2000 que débute l'aventure Grisbi. L'objectif de son fondateur est de proposer une alternative *open source* au logiciel propriétaire « Money » de Microsoft. Ce qui n'est au début qu'un « modeste » logiciel libre de comptabilité personnel devient, au fil des ans, une référence incontournable du monde libre de la comptabilité. Ainsi, dans un numéro spécial d'avril 2005 consacré aux « Stars ou futurs standards[8] » le e-magazine *01Net* le classe parmi les 100 logiciels libres de référence. Projet traduit en onze langues différentes, Grisbi a été téléchargé 180 000 fois depuis mars 2004. Aujourd'hui, le fondateur ainsi que deux autres administrateurs sont épaulés par 26 développeurs s'occupant du développement proprement dit, mais aussi de l'assemblage des différents modules, des traductions des documents et de la rédaction des codes. L'équipe de développement est ainsi composée de 29 membres.

Grisbi est un logiciel libre de gestion de comptabilité personnelle. Il est développé en langage de programmation C originellement sous Linux. Une version fonctionnant sous Windows est désormais disponible. Le principe de base d'un logiciel de comptabilité est de permettre de classer des opérations financières pour pouvoir ensuite les analyser. Comme le mentionne le manuel de l'utilisateur, les développeurs cherchent à privilégier *« la simplicité et l'efficacité pour un usage de base, sans toutefois exclure la sophistication nécessaire à un usage plus avancé. Les fonctionnalités futures tenteront toujours de respecter ces critères. »* (www.grisbi.org/documentation/fr/grisbi-manuel003.shtml#toc5, décembre 2006)

[8] http://www.01net.com/article/291881.html

Grisbi est annoncé par ses développeurs comme étant en totale conformité avec la logique française de la comptabilité. Les principales fonctions sont la gestion multi-comptes et multi-utilisateurs, la gestion multi-devises, les possibilités de virements entre comptes, la description des opérations, la planification d'opérations avec validation automatique ou manuelle, la surveillance des échéances, l'analyse et l'édition de rapports financiers et l'impression des états.

2. LA PRISE DE DECISION DANS UNE EQUIPE DE LA COMMUNAUTE *OPEN SOURCE* : ANALYSES, RESULTATS ET COMMENTAIRES

2.1. PROCEDURE DE COLLECTE ET DE TRAITEMENT DES DONNEES

L'objectif de la recherche est de vérifier si la prise de décisions au sein d'une équipe-projet de la communauté *open source* est de type *garbage can*. Il s'agit d'une recherche longitudinale sur un processus de conception et de développement de versions successives d'un logiciel. L'analyse des mails échangés entre les développeurs d'un logiciel constitue une méthode de recherche qui permet une étude détaillée des échanges dans leur forme d'origine, sur des durées pouvant être longues et sans intervention du chercheur. Même si les échanges portent sur des développements terminés (les versions du logiciel ont bien été diffusées), le chercheur a la possibilité d'avoir accès aux informations échangées au fur et à mesure du déroulement des actions, dans les conditions du « temps réel » (Chanal, 2000). Les développements suivants ont pour objectif de préciser ce choix méthodologique et les procédures de collecte des données et de codage.

2.1.1. Choix méthodologique et collecte des données

L'approche behavioriste (Romelaer, 1992, p. 49) retenue pour l'étude de situations de gestion concrètes, ne peut se faire, dans l'analyse d'équipes virtuelles, qu'avec un matériau permettant d'avoir la vision la plus holistique possible des faits émanant de celle-ci. Etant séparé dans l'espace et dans le temps des travaux de l'équipe, le chercheur ne peut plus observer de manière synchrone les actions des agents, mais doit dès lors, recourir à la forme maximisant ses chances de captation de la réalité : le mail.

Au sein de la communauté des logiciels libres il existe un apprentissage progressif des normes et règles sociales de la communauté (Kuk, 2006, p. 1034). Parmi ces règles et habitudes, les utilisateurs intègrent notamment le mail comme étant l'outil de communication légitime et quasi primordial de cette dernière. Le mail est le seul outil asynchrone permettant aux équipes

virtuelles de communiquer. De plus, de façon pragmatique, le mail en tant que document est le seul qui fait l'objet d'un archivage systématique.[9]

Un projet *open source* bénéficie généralement de différentes *mailing-lists* (listes de diffusions). Certaines sont dédiées aux utilisateurs et d'autres aux développeurs. Pour le logiciel Grisbi, nous avons comptabilisé 12 *mailing-lists* différentes[10] qui chacune concerne un niveau d'implication dans le développement du logiciel différent. C'est un cas très fréquent dans le monde du logiciel libre. La structure de participation se décompose en deux étages. Le premier est consacré au report de problèmes remarqués par les beta-testeurs et les utilisateurs des versions stables mais aussi aux commentaires et aux encouragements. Le second étage est consacré au développement (Lee et Cole, 2003, p. 637). Il semble, pour notre étude dont l'objectif est de mettre en exergue les logiques de prise de décision au sein d'un projet *open source*, que l'analyse de ce second étage « développement » soit suffisante.

Nous avons donc étudié le forum de discussion des développeurs du logiciel libre de comptabilité Grisbi (http://sourceforge.net/mailarchive/forum.php?forum_id=39354). Le référencement du projet Grisbi sur Sourceforge.net permet d'obtenir l'intégralité des mails échangés par les membres de l'équipe de développement. Dans cet article, nous utilisons l'ensemble des mails dédiés à huit versions successives du logiciel (de la 0.4.4 à la 0.5.5, voir le graphique 1), soit 2256 mails. La période couverte par l'analyse s'étend du 25 mars 2004 au 13 mai 2005. Notre recherche ne porte pas sur les débuts du projet du fait d'un changement d'hébergeur. Lors de la création de Grisbi en avril 2000 l'hébergeur utilisé par la communauté Grisbi était *TuxFamily*, mais à la suite de problèmes techniques, ces derniers ont migré vers Sourceforge.net. Une procédure de sélection des données en deux étapes a été menée.

Il s'agissait tout d'abord d'identifier pendant la période des situations de gestion et d'en sélectionner un échantillon présentant un intérêt certain par rapport à l'objectif de la recherche. L'identification des situations de gestion est rendue aisée par les règles en vigueur au sein des équipes-projets et les fonctionnalités présentes sur Sourceforge.net. D'une part, les développeurs indiquent systématiquement l'objet du message envoyé et ne traitent que d'un seul sujet par

[9] Il existe un autre support de communication, l'ICR (*Internet Relay Chat*) qui est un outil synchrone de dialogue. L'inconvénient de ce dernier est dans la logique même de synchronisation des communications. Comme il n'y a pas de logique de stockage dans ces conversations, l'utilisation de ce support ne semble pas être le matériau adéquat pour des développeurs éclatés tant spatialement que temporellement.

[10] Ces 12 mailing-lists concernent les utilisateurs, les documentations en français et en anglais, les développeurs, les archives et les CVS (Control Version System).

message[11]. Il est donc aisé d'identifier pour chaque message l'objet de la communication : proposition d'une solution, relevé d'un dysfonctionnement… Un développeur souhaitant réagir doit reprendre le même objet. D'autre part, les outils de référencement permettent, avec l'option *Threaded* de retrouver l'intégralité des mails portant sur un sujet identifié.

Il s'agissait ensuite de sélectionner des situations de gestion présentant un intérêt pour un travail sur la prise de décision. Dans la plupart des théories du choix, le processus de décision n'est possible que s'il y a présence de quatre éléments : des problèmes, des solutions, des participants et des occasions de choix (March et Simon, 1999). A partir de la lecture chronologique de la totalité des mails échangés pendant la période d'observation, nous avons donc cherché à sélectionner les situations de gestion comprenant ces quatre éléments (tableau 3). Cette procédure de sélection des situations de gestion a été menée par trois chercheurs. Seules les situations sélectionnées par les trois ont été conservées.

Finalement, 12 situations de gestion ont été identifiées et analysées sur la période (voir l'annexe A). Le graphique 1 permet de situer ces situations par rapport aux différentes versions de Grisbi. Ce graphique présente également les mails envoyés et analysés. La lecture chronologique de ces mails a permis, en répliquant le mode opératoire utilisé pour le texte de Raymond (1998), de valider la présence des conditions C1, C2 et C3 nécessaires à un fonctionnement *garbage can*.

Tableau 3. La sélection des situations de gestion : quatre critères de choix

Eléments	Définition	Exemples de situations non retenues
Problèmes	Ce qui peut soucier les membres de l'organisation, à l'intérieur comme à l'extérieur : *bugs*, fonctionnalités, répartition des tâches, recrutement… Tous ces problèmes demandent attention.	Des mails portent sur le référencement du logiciel sur différents sites ou sur une couverture presse. Certaines informations comptables sont diffusées par le Web Designer auprès des développeurs.
Solutions	Ce qui est proposé par quelqu'un. La solution peut être nouvelle ou s'appuyer sur des pratiques anciennes.	Des mails nombreux restent sans réponse. Des utilisateurs font part de leur difficulté à comprendre ou à utiliser une fonctionnalité mais personne ne propose de solution.
Participants	Ceux qui interviennent au cours du processus de décision : apporteur de problèmes, de solutions, décideur… Les membres d'un projet *open source* étant bénévoles, chaque « entrée » dans le processus correspond à une « sortie » ailleurs (l'activité professionnelle notamment). Les variations	Un problème est rencontré par un utilisateur qui ne comprend pas certaines fonctionnalités. Le développeur concerné précise le fonctionnement.

[11] Ces règles de conduite sont compilées dans le document « Comment Poser Les Questions De Manière Intelligente » disponible sur différents sites (par exemple le site français de Linux). Parmi ces règles d'or, il y a notamment « la description précise et explicite du problème » et la mention systématique de l'objet du mail : *« Sur les mailing lists ou les newsgroups, le sujet du mail est une occasion en or pour attirer l'attention d'experts qualifiés en 50 caractères ou moins. Ne le gaspillez pas en babillages… N'essayez pas de nous faire apitoyer sur votre sort, utilisez plutôt l'espace disponible pour décrire le problème de manière très concise. »*

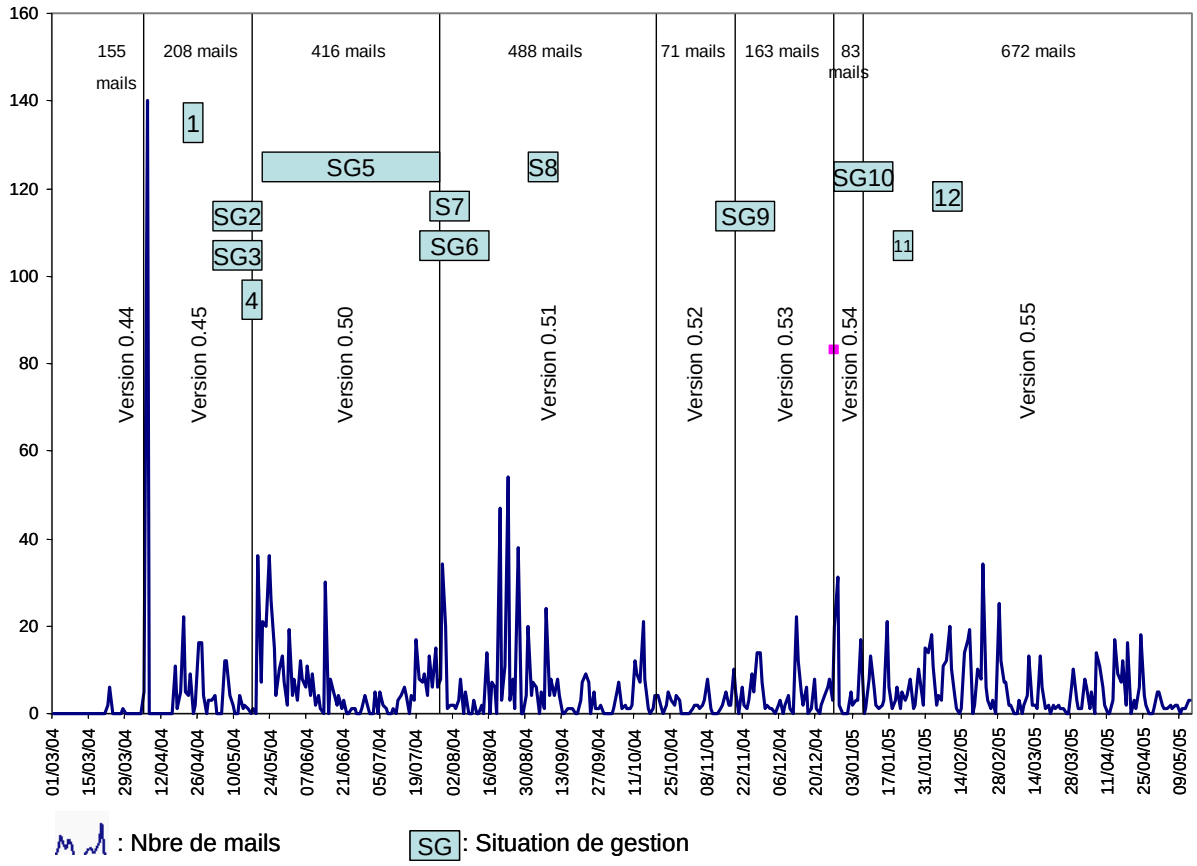
	importantes de participation proviennent des autres sollicitations sur le temps des participants.	
Occasions de choix	Ce sont les occasions où l'organisation doit adopter un comportement de décision. Elles peuvent se produire régulièrement (livraison d'une nouvelle version) ou revêtir un caractère exceptionnel (départ d'un développeur, problème urgent…).	Un membre de l'équipe souligne un problème mais l'équipe n'est pas en situation de pouvoir choisir : des informations sont manquantes, les développeurs travaillent sur une autre question…

2.1.2. La procédure de codage

Afin de repérer ou non des prises de décision en mode *garbage can* dans chacune des situations de gestion, il était indispensable d'avoir à disposition des caractéristiques suffisamment précises pour pouvoir être utilisées dans une analyse de contenu. Les 21 dimensions caractéristiques du fonctionnement en mode *garbage can* recensées par Romelaer et Huault (1996) (tableau 1) ont été utilisées. Ces caractéristiques présentent l'avantage d'avoir été observées au moins une fois dans des organisations de type *garbage can* dans des recherches cliniques. Elles ont été employées ici comme des catégories *a priori* (Miles et Huberman, 1991) suffisamment précises pour ne pas nécessiter de redéfinitions ultérieures (Lincoln et Guba, 1985) et présentées sous forme de thèmes.

Les situations de gestion sélectionnées ont fait l'objet « d'analyses thématiques qualitatives » (Bardin, 1993, p. 148) destinées à vérifier la présence ou l'absence des catégories. Le contenu des différents mails utilisés a été découpé en unités d'analyses, elles-mêmes classées dans les catégories définies. Les unités d'analyse retenues sont des portions de phrases, des phrases entières, voire des groupes de phrases se rapportant à un même thème. Le matériau empirique a été traité et synthétisé au moyen d'une méta-matrice non ordonnée (Miles et Huberman, 1991). Ce protocole a été testé par une procédure de triple codage (Weber, 1990). Il s'agissait de vérifier la définition des unités d'analyses et leur classification dans les catégories (fiabilité inter-codeurs) et de traiter les divergences. En complément, le nombre de mails envoyé par chaque développeur pendant la période étudiée et dans chaque séquence a été calculé.

Graphique 1. Mails envoyés, versions diffusées et situations de gestion étudiées dans le projet Grisbi :



N.B. La position des situations de gestion sur l'axe des ordonnées n'a aucune signification.

2.2. RESULTATS ET COMMENTAIRES

Dans ce dernier paragraphe, les résultats issus de la catégorisation des données sélectionnées sont tout d'abord présentés. Ils font ensuite l'objet d'une discussion qui est l'occasion de revenir sur la validité du GCM au sein de l'équipe étudiée.

2.2.1. La prise de décision au sein de l'équipe Grisbi est-elle de type *Garbage Can* ?

Le tableau 4 présente le résultat du codage des 12 situations de gestion étudiées. Ce tableau autorise trois types de commentaires. Il est tout d'abord possible de conclure sur la présence des 21 dimensions caractéristiques du *garbage can* dans chacune des situations. Il semble ensuite intéressant de présenter en détail les situations les plus proches du mode *garbage can* (SG5 et SG12). Enfin, il est possible de repérer les dimensions les plus (ou les moins) fréquentes sur la

période analysée afin d'en tirer des enseignements sur la prise de décision au sein de ce type d'équipe virtuelle.

Tableau 4. Résultat du codage des situations de gestion à partir de la grille Romelaer-Huault (1996)

	Situations de gestion (nbre de jours) [nbre de participants] /nbre de mails/												
Dimensions caractéristiques du GCM	1 (9) [2] /15/	2 (15) [6] /28/	3 (14) [5] /14/	4 (2) [2] /7/	5 (69) [9] /61/	6 (28) [5] /17/	7 (17) [7] /46/	8 (18) [8] /32/	9 (23) [5] /23/	10 (22) [7] /29/	11 (3) [5] /11/	12 (12) [17] /59/	∑
1. Circonstance													**0**
2. Objectifs													**0**
3. Contacts					X	X	X				X		**4**
4. Routines		X	X					X		X			**4**
5. Fonction					X	X			X				**3**
6. Implication					X				X	X	X	X	**5**
7. Participation						X				X		X	**3**
8. Données		X	X	X	X			X				X	**6**
9. Controverses			X		X			X				X	**4**
10. Voisinage	X	X			X	X	X		X		X		**7**
11. Préférences		X		X								X	**3**
12. Engagement	X				X								**2**
13. Solutions											X		**1**
14. Structure				X	X				X				**3**
15. Intervention		X			X	X			X	X	X	X	**7**
16. Idéologies	X	X			X						X	X	**5**
17. Champ					X		X						**2**
18. Processus	X			X	X		X			X		X	**6**
19. Symbolique			X		X							X	**3**
20. Pouvoir			X									X	**2**
21. Rencontre					X								**1**
TOTAL	**4**	**6**	**5**	**4**	**14**	**5**	**4**	**3**	**5**	**5**	**6**	**10**	

Sur le premier point, force est de constater que le nombre de dimensions présentes dans chacune des situations est relativement faible. Les 21 dimensions proposées par Romelaer et Huault (1996) peuvent être considérées comme des conditions de présence d'un mode *garbage can*. En d'autres termes, si plusieurs de ces conditions sont remplies, il est possible de conclure que la décision et de type *garbage can*. Bien entendu, il est délicat de déterminer le nombre minimum de conditions devant être présentes pour que l'on puisse valider la présence du *garbage can*. On peut noter cependant que 8 situations sur les 12 étudiées présentent moins d'un quart des conditions requises (SG : 1/3/4/6/7/8/9/10). Dans celles-ci, les préférences (D11) semblent clairement établies au niveau du décideur final ou par convergence des préférences individuelles des acteurs engagés. Ces derniers s'engagent dans le processus de décision (D12) avec des préférences qui ne sont pas forcément identiques mais qui apparaissent comme compatibles. Il

n'y a pas donc pas forcément besoin d'arrangements entre les acteurs ni de contreparties. Cette convergence est rendue sans doute aisée par l'absence de solutions « toutes faites » portées par certains acteurs (D13). L'équipe cherche essentiellement à élaborer des solutions spécifiques. La décision prise n'est pas influencée par des différences de valeur, en particulier parce que l'on cherche essentiellement à répondre à des problèmes techniques à faible portée symbolique (D19). Les rôles sont clairement définis, ils ne sont pas remis en cause. Le leader ne voit pas forcément dans la situation une occasion d'afficher son statut et son pouvoir. Les acteurs « périphériques » ne considèrent pas la situation comme de nature à renforcer leur position, à gagner en légitimité où à conquérir un droit à participer (D20). Ils s'engagent parce que le problème ou la solution est important pour eux.

Le tableau 4 permet ensuite de mettre en évidence une situation de gestion (SG5) très caractéristique d'un mode *garbage can*. Cette situation, et dans une moindre mesure la situation 12, méritent, compte tenu des objectifs de cette recherche, quelques commentaires supplémentaires. Il est en particulier intéressant de pointer les dimensions repérées simultanément dans ces deux situations.

Les situations 5 et 12 correspondent à des moments dans la vie de l'équipe où la nature du problème, les données pertinentes pour le régler et les critères à privilégier font l'objet de controverses (D8). Dans la SG5, le choix du *packager* d'utiliser la seule version 10 de Mandrake (voir annexe A) est discuté. Cette décision individuelle est critiquée notamment parce qu'elle n'est pas vue unanimement comme cohérente avec l'histoire du projet. La SG5 apparaît ainsi comme une séquence pendant laquelle un problème amène les participants à rappeler des choix passés. Or, la signification de ce qui a déjà été réalisé et les interprétations des membres sont diverses voire contradictoires (D9). Dès lors, les décisions à prendre présentent un fort contenu symbolique (D19) : certains acteurs voient dans la décision une occasion de rappeler les finalités du projet et leurs valeurs. Elles font aussi l'objet de controverses, certains échanges sur les motifs premiers de ces décisions étant assez « vifs » (D9). Les membres de l'équipe, ressentent très vraisemblablement l'importance de la situation et participent largement aux débats (les SG5 et SG12 sont caractérisées par un nombre important de mails et de participants). De même, des acteurs qui n'on pas *a priori* de raisons de participer (ils sont non-membres) s'impliquent néanmoins en donnant leurs avis et en proposant des solutions (D6). En revanche, certains membres qui, de par leur statut, devraient s'impliquer, ne le font pas. Dans certains mails,

certains acteurs regrettent d'ailleurs ces absences (D15). Dans les deux situations, des acteurs n'hésitent pas à rappeler des principes fondateurs, les grandes règles de la communauté *open source*. L'idéologie et la culture apparaissent bien comme des facteurs qui influencent les décisions, notamment en restreignant le nombre des « possibles » (D16). Finalement, des décisions concernant les acteurs sont prises : « démission » d'un packager (SG5) et « exclusion » d'un développeur (SG12). Cependant, alors que le problème de départ n'est pas traité, les échanges sur le sujet s'arrêtent. Dans la SG5 le manager de projet décide de repousser la question à une autre version.

SG12 : Un packager : *« Non seulement je considère que j'ai été trompé sur la marchandise car les promesses sur les fonctionnalités ne sont pas tenues, mais je me considère également trahi car mes comptes sont plombés à cause de la présence récurrente de bogues » Le [Web Designer] a réussi à imposer des choix techniques auxquels une majorité était opposée et a refusé de mettre en place certains autres. »*
Le *project manager* : *« Je te prie donc d'avoir l'élégance de bien vouloir te désinscrire des mailing-lists de Grisbi auxquelles tu es encore inscrit, de te désinscrire du projet SourceForge, de supprimer ton adresse email des administrateurs des mailing-lists, de quitter le canal IRC ainsi que tout ce que j'ai pu oublier ».*
Source : Mails envoyés pendant la SG12

Il y aurait donc bien au sein des équipes-projets de la communauté *open source* des séquences dans lesquelles les acteurs prennent des décisions sur un mode de type *garbage can*. Pour terminer cette présentation des résultats il faut cependant s'arrêter sur les dimensions les plus (et les moins) repérées sur la période d'observation afin de mettre en lumière des spécificités propres au cas étudié. Le tableau 4 indique tout d'abord que les dimensions 10 et 15 sont celles que l'on retrouve le plus fréquemment (7 fois sur 12). Les développeurs de la communauté sont des bénévoles qui doivent parfois arbitrer entre les exigences du développement du logiciel et leurs obligations professionnelles. Ainsi, la participation au développement peut être fluctuante. Dans une situation de gestion particulière, les acteurs qui devraient intervenir pour traiter le problème ou proposer une solution, ne le font pas toujours parce qu'ils sont trop occupés ailleurs (D7/D15). Pour pallier ces absences, les « leaders » sont amenés parfois à traiter une question qui ne les concerne pas beaucoup au départ (D5) en mobilisant largement les personnes de leur entourage (D3, repérée 4 fois) et/ou en cherchant à reprendre des solutions qui « ont fait leur preuve » dans un autre contexte (D10).

Dans la SG6, un développeur a pris du retard dans l'écriture d'une procédure de compilation. [Project Manager 1] propose d'attribuer ces tâches de compilation à un « doc Writer » de l'équipe. *« Il faudrait envoyer ça à [Doc Writer] qui fait la documentaion, je sais pas s'il est inscrit sur cette liste et moi je suis pas chez moi en ce moment... du coup j'ai pas son adresse. [Project Manager 2], tu pourrais le faire ? »* Après quelques recherches, il apparaît que la personne pressentie est indisponible. [Project Manager 1] décide donc de réaliser lui-même la tâche. *« [Doc*

Writer] est en vacances au soleil (le veinard). J'essaierai de faire la modification moi même si j'ai le temps. » Cependant, c'est finalement le développeur initial qui fournira quelques jours plus tard la procédure complète ayant réussi à se libérer.
Source : Mails envoyés pendant la SG6

En revanche, comme les échanges se font par l'intermédiaire des TIC et que toute personne intéressée par le projet peut avoir accès et participer aux forums de discussion, on note parfois des situations où des personnes s'impliquent sans aucune raison *a priori* (D6, repérée 5 fois).

Dans la SG9, un utilisateur de Grisbi qui ne fait pas partie de l'équipe propose aux développeurs un mode d'emploi pour installer Grisbi sur Mac. Le manager du projet se réjouit de cette proposition qui permet de travailler sur un point délaissé par l'équipe. Il se charge de communiquer des informations sur l'état d'avancement du travail. *« Nous sommes friands de tests et de remontées sur des systèmes autres que GNU/Linux, ton feedback nous sera précieux… Je crois que [Developer 3] s'était proposé, mais vu qu'il a déjà beaucoup de boulot avec la version Windows, je ne suis pas contre que quelqu'un d'autre s'y mette (enfin, si ça lui va). Merci pour ce howto et merci pour ta contribution à Grisbi ! »*. Cependant, l'utilisateur multipliera les erreurs : *« Honte à moi je n'avais pas fait de tests de bases… Vous pouvez m'élire boulet du jour! »*. Le manager de projet intervient alors pour nommer un responsable de l'intégration de Grisbi sous Mac.
Source : Mails envoyés pendant la SG9

Finalement, on retrouve bien ici l'image d'un réseau de développeurs dont les frontières sont floues, la répartition des tâches pas toujours clairement établie et la participation fluctuante (plus globalement la condition C1). Les observations montrent également que les conflits existent dans la communauté et que les échanges peuvent être assez « violents ». C'est le cas notamment quand des acteurs ont l'impression qu'un membre cherche à imposer une solution qui « l'arrange » ou qui est en contradiction avec les choix précédents. Les données pertinentes dans une décision et les critères à privilégier peuvent faire l'objet de controverses (D8 repérée 6 fois). Cependant, les développeurs semblent revenir très rarement sur certains choix. Les contraintes du développement simultané de plusieurs modules que l'on regroupe ensuite et les impératifs de compatibilité entre les versions obligent sans doute les développeurs à ne pas remettre en cause des choix importants. Cela est rendu possible par l'existence d'une hiérarchie au sein de l'équipe. Un noyau dur de développeurs qui participent beaucoup a autorité sur le reste de l'équipe. Ces éléments expliquent sans doute pourquoi D1 et D2 ne sont jamais repérées dans les 12 situations. Le rejet d'une solution ou encore l'abandon d'une fonctionnalité semble quasi-définitif (D1). Quand un acteur tente de montrer qu'il est possible d'atteindre des objectifs discutés par d'autres moyens, il est la plupart du temps réprimandé (D2). En d'autres termes, quand les participants considèrent qu'un acteur chercher à imposer une solution toute faite à un problème pour des motifs personnels, ils ont tendance à refuser la discussion.

Dans la SG8, un membre s'interroge sur la compilation de la version 051 en préparation.
Réponse des développeurs : - *« Question compilation, la 0.5.1pre 1 se comporte comme la 0.5.0 (merci les développeurs) donc théoriquement, tu n'auras que quelques lignes dans le [fichier spécification] à ajouter. Si tu veux tester, voici les modifications… Comme tu reprends la suite sur une même version de Mandrake, s'il y avait eu un problème, le packager l'aurait déjà vu et corrigé. »… « Si on regarde l'historique des mails de cette liste, on verra que j'ai eu une longue conversation avec [Developer 1] au sujet des spécifications ».*
Source : Mails envoyés pendant la SG8

Enfin, les observations montrent que les situations de gestion n'aboutissent pas forcément à la résolution du problème. Même s'il est possible de déterminer précisément les débuts et fins des situations de gestion par l'analyse des mails, ces situations *« isolent une tranche temporelle dans un processus qui n'a pas de début ni de fin clairement observables »* (Romelaer et Huault, 1996, p. 23). Dans certains, cas, les acteurs reprennent des problèmes et solutions déjà évoqués. De même, comme le montre le graphique 1, les acteurs n'attendent pas la fin des situations de gestion avant de diffuser les différentes versions. En d'autres termes, les situations de gestion « traversent » les différentes versions : une version est mise en ligne alors que les développeurs travaillent sur des problèmes identifiés. Le report de la décision ou la mise entre parenthèses des problèmes semble une caractéristique du fonctionnement des équipes *open source* (D18).

2.2.2. Commentaires et enseignements

Cette discussion porte sur deux thèmes distincts : l'utilisation des données secondaires dans la recherche en management et les conséquences de la validité plutôt faible du GCM dans la majorité des SG étudiées.

Le point de départ de cette recherche est une intuition provenant de données secondaires (Raymond, 1998). De ces données, nous tirons un début de preuve de la présence effective de décision de type GCM puisque nous avons démontré qu'une équipe-projet libre peut être considérée comme une anarchie organisée. Le texte de Raymond (1998) peut être envisagé comme un récit plus ou moins stylisé de pratiques générales qui ne font pas explicitement référence à des situations de gestion concrètes. Il ne permet donc pas de mettre en évidence des situations de gestion effectives dans lesquelles le processus de décision du type GCM intervient. En revanche, sa mobilisation a clairement mis en évidence le fait que les équipes de développement libres peuvent être considérées comme des anarchies organisées, organisations dans lesquelles il est fréquent de trouver des décisions prises sur un mode de type *garbage can*. Nos résultats ont montré que les décisions de type GCM sont plutôt rares. Ils permettent ainsi de

valider empiriquement, dans le cas d'une équipe d'innovation libre, que les conditions C1, C2, C3 ne s'avèrent pas globalement suffisantes pour conclure sur le pouvoir explicatif du GCM. Plus précisément, il peut exister des organisations qui sont fortement sous ces conditions mais n'utilisent pas régulièrement un modèle de décision de type *garbage can*. Nous retrouverons ici l'idée que même dans une anarchie organisée le GCM correspond bien à une situation peu fréquente. Il nous semble que l'on peut émettre l'hypothèse que pour que l'on soit en présence d'un GCM, il semble nécessaire qu'une ou plusieurs de ces conditions soient remplies, de préférence plusieurs, et ce avec « suffisamment d'intensité ». A notre connaissance, personne n'a proposé d'outil de mesure fiable de l'intensité de présence de ces conditions[12]. Au delà de la construction de cet outil de mesure, ces résultats nous montrent qu'il n'est pas possible selon nous de discuter de la validité du GCM dans une organisation *« sans études extrêmement concrètes de microprocessus très spécifiques »* (March, 1994, p.25) sous peine de dégager des résultats insuffisamment robustes[13].

Si le modèle du *garbage can* ne permet pas de caractériser la majorité des SG étudiées dans cet article, il s'avère utile de tenter d'émettre des hypothèses rivales ou tout au moins de proposer des pistes de réflexions théoriques fécondes pour caractériser les autres modes de décision utilisées par l'équipe Grisbi. Il ne s'agit pas ici d'apporter de preuve au sens scientifique du terme mais plus modestement d'identifier des modèles de décisions ayant le statut de candidats sérieux pour prétendre à la compréhension de la prise de décision au sein du projet Grisbi. Deux faits empiriques majeurs nous ont guidés dans la première exploration de ces pistes :

- la place centrale de l'idéologie et de la symbolique dans les SG étudiées et plus généralement dans le monde libre. Dans cinq des SG étudiées, la dimension 16 a été repérée (tableau 4). Plus généralement, de nombreuses analyses[14] ont montré combien la communauté des logiciels libres s'est construite sur un ensemble de valeurs fortes (solidarité, altruisme, liberté) qui sont structurantes.

[12] Ces développements relatifs à l'insuffisance de la présence de C1 à C3 pour statuer sur l'existence du GCM dans une organisation doivent beaucoup à un échange avec P. Romelaer.

[13] Sauf à travailler « en chambre » par expérimentation informatique comme l'ont fait eux-mêmes Cohen et *al.* (1972) et Masuch et LaPotin (1989).

[14] Par exemple Ferrary et Vidal (2004) et Garcia et Steinmueller (2003). Pour un recensement récent voir Von Krogh et Von Hippel, (2006).

- le rôle du temps dans la décision. En particulier, dans quatre des SG étudiées, le choix a été fait de reporter la décision à une date ultérieure pour des raisons de priorité basse (SG4, SG5) ou sans raison explicite (SG6 et SG10).

Sur la base de ces faits, trois modèles ou parties de modèles peuvent selon nous être mobilisés pour mieux comprendre la prise de décision au sein de Grisbi. Deux premièrs permettent de mettre en avant la prégnance de l'idéologie : le modèle des organisations génératrices d'action de Starbuck (« *self-designing organizations* ») et celui de l'incrémentalisme cognitif de Johnson (1988). Le concept de quasi résolution de conflit issu de l'entreprise adaptative de Cyert et March (1963) fournit une analyse prometteuse du phénomène de report de décision présenté ci-dessus.

- La prégnance de l'idéologie

L'un des points centraux de la théorie des *« self designing organizations »* est le lien entre idéologies et prise de décision. En regroupant sous le terme d'idéologie un ensemble de rites, de valeurs, de croyances et de symboles, Starbuck (1982) montre que les organisations oscillent entre le traitement rationnel (ou quasi-rationnel) de problèmes et la génération d'actions. Celles-ci sont justifiées *a posteriori* par la mobilisation de l'idéologie de l'organisation. Autrement dit, un résultat peut être jugé comme bon parce qu'il est compatible avec l'idéologie de l'organisation. De plus, l'idéologie est elle-même génératrice de problèmes et invite à une représentation relativiste des problèmes qui sont souvent volontairement vagues et mal définis ou remodelés afin de justifier des intentions d'action (Starbuck, 1983).

> Dans la SG5, le packager annonce qu'il ne souhaite plus assurer le paquetage de Grisibi sous Mandrake. *« Mandrake prend des orientations qui ne me conviennent pas et j'ai donc décidé de changer de distribution. Il va donc falloir trouver un nouveau packageur Mandrake pour les versions suivantes… Bon courage. »*
> Des discussions s'engagent alors sur les mérites respectifs des différentes distributions Linux.
> *- « Tu vas utiliser quoi au fait ? Personnellement j'ai de plus en plus envie de passer à Fedora Core et vu ce que vous dîtes je crois que je vais faire le pas. La Fedora est dans l'esprit Linux Debian GPL ; il n'y a pas cette pollution commerciale et ses contraintes, donc c'est plus fiable et on trouve quand même tous les derniers programmes par le biais des contributeurs et tout ça sans avoir besoin d'attendre des mois ou de devoir payer une redevance à un quelconque club …*
> *- « Merci de ces tuyaux. Il y a un endroit où cela serait recensé ? En fait c'est un peu un retour aux sources pour moi car le premier linux que j'avais installé était une RedHat 6.0. »*
> *- « Ci-dessous la totalité des fournisseurs utilisables pour Fedora… »*
> Source : mails échangés pendant la SG5

En sus de l'importance de l'idéologie, nous avons repéré une certaine logique d'irréversibilité des décisions. Il n'a en effet pas été possible de mettre en évidence des remises en cause des décisions prises dans les 12 SG étudiées (D1). L'adaptation de l'équipe par petit pas au sein d'un

ensemble agrégé de croyance, rites et valeurs laisse penser qu'il pourrait exister un incrémentalisme cognitif au sein d'une équipe de développement libre fondé notamment sur le paradigme organisationnel (Johnson, 1988).

Dans la SG7, les développeurs ont une volonté constante de reprendre des options précédentes et d'aller vers une compatibilité renforcée avec les distributions Linux sélectionnées (notamment Fedora). *« Si on veux que Grisbi ait une chance d'entrer dans Fedora, il faut que j'utilise leurs paquets et leurs spécificités ; « latex-ucs » n'existe pas dans Fedora et comme le fichier qui nous intéresse est dans un paquet qui a déjà passé la 1ère étape de test, je suis obligé de l'utiliser ; c'est sûr que l'on se fait jeter si on ajoute un paquet en dépendance qui a de forte chance d'entrer en conflit avec le reste et en plus fera doublon. »*
Source : mails échangés pendant la SG7

Explorer cette piste passerait nécessairement par la mise au jour du paradigme des membres de l'équipe Grisbi et par l'analyse des mécanismes précis grâce auquel ce paradigme agit sur le processus de décision. Un paradigme étant construit par les participants mais échappant ensuite à leur contrôle, il s'avèrerait sans doute instructif de préciser le rôle du paradigme dans la génération des problèmes et des solutions recevables (c'est-à-dire compatibles avec le paradigme). Pour continuer dans ce sens, il est utile de remarquer que si l'on accepte l'idée que le paradigme s'apparente à une institution, cette piste de recherche rejoindrait en partie celle de Levitt et Nass (1989). En associant la théorie néo-institutionnelle et le GCM, ceux-ci ont mis en évidence que les contraintes institutionnelles conduisent à une hétérogénéité des décisions plus limitée que dans le GCM. On retrouverait alors les deux tendances opposées nécessaires à toute organisation innovatrice : une tendance au désordre (*via* le GCM) permettant l'exploration et une tendance à l'ordre (*via* le paradigme) plutôt favorable à l'exploitation[15].

- le report de la décision

Dans plusieurs des SG étudiés, des décisions ont été reportées à une date ultérieure. En l'état actuel, cette procrastination décisionnelle est difficile à interpréter. Plusieurs hypothèses plus ou moins rivales sont émises, sans qu'il soit possible de trancher :

- s'agit-il d'une tactique managériale permettant de gagner du temps pour attendre une occasion de choix plus propice ?

- la procrastination décisionnelle peut-elle être analysée par le concept de quasi-résolution de conflit de Cyert et March (1963) ? Est-elle une forme d'attention séquentielle qui permet de

[15] Cette remarque rejoint celle de Lassalle – de Salins (2006) repérée dans le cas du Codex Alimentarius, organisation intergouvernementale (dépendant notamment de l'OMS) chargé d'élaborer des normes dans le domaine alimentaire.

traiter successivement (*via* le report) des problèmes qui exigeraient des solutions incompatibles s'ils étaient considérés en même temps ?

- A partir du moment où une certaine irréversibilité des choix a été mise en évidence, le report décisionnel n'est-il pas une tactique de report des irréversibilités ? L'objectif serait alors de maximiser l'adaptabilité (ce que traduit le terme « *postponement* » en anglais), principe prôné notamment par Simon (1991).

CONCLUSION

L'objectif de ce papier était de confronter la prise de décision des équipes projets au *GCM*. Cette confrontation a nécessité deux temps. Premièrement, la mobilisation de données secondaires a permis de valider l'intuition originale consistant à considérer qu'une équipe projet libre est une anarchie organisée. Cette validation constitue un début de preuve de la présence du GCM puisqu'elle est une condition nécessaire mais non suffisante de l'existance effective de ce modèle. Elle a donc dans un deuxième temps été complétée par l'analyse de situations de gestion concrètes. Cette analyse a montré que ce modèle était effectivement visible mais dans un nombre limité de cas. L'apport de cette recherche est donc triple. Tout d'abord, nous avons mis en évidence au sein d'un groupe d'individu relativement restreint (une équipe–projet) la présence effective du GCM alors que la majorité des validations empiriques du modèle concerne des grandes organisations (en particulier des Universités). Ensuite, l'équipe-projet en question est « virtuelle » (Jarvenpaa et Leidner, 1999). Or, à notre connaissance, aucune validation empirique du modèle n'avait été effectuée sur ce type d'organisation aujourd'hui en plein développement. Enfin, les travaux de March prennent peu en compte les comportements innovateurs dans lesquels l'incertitude joue un rôle important. De ce point de vue, notre recherche, par sa focalisation sur un projet d'innovation, apporte un éclairage instructif sur l'utilisation du GCM dans ce contexte.

L'objet étudié, une équipe dédiée au développement d'un logiciel libre, présente également un intérêt certain. La capacité des firmes à exploiter les savoirs externes est un élément crucial de leur capacité d'innovation (Cohen et Levinthal, 1990, p : 128). Partant de ce constat, Chesbrough (2003) développe l'*Open Innovation Model* qui traduit l'importance d'allier R&D interne et idées en provenance de l'extérieur. C'est ainsi qu'une multitude de communautés libres d'accès,

chercheurs universitaires, communautés virtuelles et même mondes virtuels[16], deviennent de véritables viviers d'idées et de capacités créatrices pour des entreprises désireuses « d'absorber » ces connaissances externes. Parmi celles-ci, la communauté des logiciels libres offre de nouvelles possibilités aux entreprises désireuses de réaliser ce qu'elles ne pourraient accomplir seules dans des domaines liés à l'informatique. De nombreuses entreprises se sont d'ores et déjà lancées dans cette expérience. A titre d'exemple, la technologie Safari, développée par Apple *via* la communauté, est utilisée comme navigateur par Nokia dans sa série de téléphones mobiles S60. De même, la NASA a utilisé des modules développés en *open source* pour l'analyse des données d'une sonde en mission sur Mars (Norris, 2004). Dans ce type d'organisation, il semble impératif pour l'initiateur du projet de comprendre les motivations et les comportements des développeurs, plus largement le fonctionnement des équipes « libres ». Au-delà, ce type de coopération entreprise privée – organisation non marchande pose le problème de la gouvernance la plus appropriée. Dans ce contexte, cette recherche sur la prise de décision au sein d'une équipe dédiée à la conception d'un logiciel peut présenter un réel intérêt managérial. En se référant à Romelaer (1994), il s'avèrerait ensuite possible de mettre au jour un certain nombre de préceptes opérationnels permettant un management (satisfaisant) des organisations de type GCM.

Cette recherche n'est pas exempte de limites. Celles-ci concernent en particulier le matériau empirique retenu. Malgré l'effort de justification du projet retenu, il est clair que les contraintes d'accès aux données ne nous ont pas permis de suivre le déroulement du projet dès son début mais de le prendre en cours. Or les phases amont d'un projet sont généralement associées à une incertitude plus élevée que les phases suivantes. Il aurait donc été instructif de confronter le GCM à la prise de décision dans ces phases amont. Par ailleurs, il conviendrait sans doute d'augmenter le nombre de situations de gestion étudiées pour rendre les résultats plus robustes.

Outre cet accroissement du matériau empirique analysé, d'autres prolongements de ce projet de recherche sont envisagés. En particulier, la mobilisation du GCM n'ayant pas permis de caractériser la majorité des processus de décision dans les situations de gestion étudiées, nous souhaitons recourir à d'autres théories explicatives. La prégnance de l'idéologie dans la communauté des logiciels libres nous incite en particulier à tester la validité de deux autres

[16] L'exemple des tests-produit de Toyota ou d'*American Apparel* (chaîne de vêtements pour hommes, femmes, enfants et animaux) sur *Second Life* en est un bon exemple.

théories des organisations et de la décision : l'incrémentalisme cognitif de Johnson (1988) et le modèle des organisations génératrices d'action de Starbuck (1982, 1983).

REFERENCES

Bardin, L. (1993), *L'analyse de contenu*, Paris : PUF.

Chanal, V. (2000), La structuration d'un projet d'innovation par la communication électronique, Actes de la *9ème Conférence de l'AIMS*, Montpellier, 24-26 mai.

Chesbrough W. H. (2003), *Open Innovation- The New Imperative for Creating and Profiting from Technology*, Harvard Business School Press, Boston MA.

Cohen, M. D. et D. A. Levinthal (1990), Absorptive Capacity: A New Perspective on Learning and Innovation, *Administrative Science Quarterly*, 35 : 1, 128-152.

Cohen, M. D.; J. G. March et J. P. Olsen (1972), A Garbage Can Model of Organizational Choice, *Administrative Science Quarterly*, 17 : 1, 1-25.

Cyert, R.M. et J.G. March (1963), *A behavioral theory of the firm*, Englewood Cliffs N.J., Prentice Hall Inc.

Demil, B. et X. Lecocq (2006), Neither market nor hierarchy nor network: The emergence of bazaar governance, *Organization Studies*, 27 : 10, 1447-1466.

Edwards, K. (2001), Epistemic Communities, Situated Learning and Open Source Software Development, *Working Paper for Epistemic Cultures and the Practice of Interdisciplinarity*, NTNU Workshop, Trondheim.

Ferrary, M., et P. Vidal (2004), Les leçons de management de la communauté Linux, *Actes de la 13ème Conférence de l'AIMS*, Le Havre, 1-4 juin.

Garcia, J.M., et W.E. Steinmueller (2003), The Open Source Way of Working: a New Paradigm for the Division of Labour in Software Development?, *INK Research Working Paper*, SPRU, 1, University of Sussex.

Ghosh, R.A., et P.A. David (2003), The Nature and Composition of the Linux Kernel Developer Community: a Dynamic Analysis, *Working Paper*, NSF Open Source Software Project, Stanford Institute for Economic Policy Research.

Girin, J. (1983), Les situations de gestion, Avant propos, *in* Berry M. (dir), *Le rôle des outils de gestion dans l'évolution sociaux complexes*, CRG/Polytechnique, Rapport pour le Ministère de la recherche et de la technologie.

Girin, J. (1989), L'opportunisme méthodique dans les recherches sur la gestion des organisations, *Journée d'étude La recherche-action en action et en question*, AFCET, Collège de systémique, Ecole Centrale de Paris, 10 mars.

Girin, J. (1990), Analyse empirique des situations de gestion : éléments de théorie et de méthode, in A-C. Martinet (dir.) *Epistémologies et Sciences de Gestion*, Economica : Gestion G, 141-182.

Jarvenpaa, S.L., et D.E. Leidner (1999), Communication and Trust in Global Virtual Teams, *Organization Science*, 10:6, 791-815.

Johnson, G. (1988), Rethinking Incrementalism, *Strategic Management Journal*, 9 : 1, 75-91.

Kim, A. J. (2000), *Community Building on the Web: Secret Strategies for Successful Online Communities*, Addison Wesley, London.

Kuk, G. (2006), Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List, *Management Science*, 52, 1031-1042.

Lakhani K.R. et E. Von Hippel (2003), How open source software works:, Free user-to-user Assistance, *Research Policy*, 32 : 6, 923-943.

Lassalle – de Salins, M. (2006), Les organisations intergouvernementales comme cible des stratégies politiques des entreprises. Peut-on influencer des lieux d'indécision ponctuée de décisions ? , Actes de la *15ème Conférence de l'AIMS*, Annecy/Genève, 13-16 juin.

Lee, G. K. et R. E. Cole (2003) From a firm-based to a community based model of knowledge creation: The case of the Linux Kernel development, *Organization Science*, 14 : 6, 633–649.

Lerner, J. et J. Tirole (2002), Some simple economics of open source, *Journal of industrial economics*, 50 : 2, 197-297.

Levitt, B. et C. Nass (1989), The Lid on the Garbage Can: Institutional Constraints on Decision Making in the Technical Core of College-Text Publishers, *Administrative Science Quarterly*, 34 : 2, 190-207.

Lincoln, Y.S. et E.G. Guba (1985), *Naturalistic Inquiry*, London : Sage.

Logerot, P. (2003), *Linux ou Windows ?*, Paris : Dunod.

Loilier, T. (2002), Gestion de l'innovation : quels enseignements tirer du cas des logiciels libres ?, *Finance-Contrôle-Stratégie*, 5: 3, 141-168.

March, J. G. et H.A. Simon (1999), *Les Organisations*, Dunod : Paris.

March, J.G. (1994), L'avenir de la gestion vu par JG March, *Revue Française de Gestion*, 100, 22-27.

Markus, M.L., B. Manville et C.E. Agres (2000), What Makes a Virtual Organization Work?, *Sloan Management Review*, 42 : 1, 13-26.

Masuch M. et P. LaPotin (1989), Beyond Garbage Cans: an AI model of organizational choice, *Administrative Science Quarterly*, 34 : 1, 38-67.

Miles, M.B., et A.M. Huberman (1991*), Analyse des données qualitatives, recueil de nouvelles méthodes*, Bruxelles : De Boeck Université.

Moon, J.Y. et L. Sproull (2000), Essence of Distributed Work: the Case of the Linux Kernel, *First Monday*, 5 : 11.

Norris, S. J. (2004), Mission-Critical Development with Open source Software: Lessons Learned, *IEEE*, 21 : 1, 42-49.

O'Mahony, S. (2003), Guarding the Commons: How Community Managed Software Projects Protect their Work, *Research Policy*, 32 : 7, 1179-1198.

Raymond, E.S. (1999), The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, Sebastopol: O'Reilly and Associates.

Raymond, E.S., (1998), La Cathédrale et le Bazar, *http://www.linux-france.org/article/these/cathedrale-bazar.*

Romelaer, P. (1994), James March, *Revue Française de Gestion*, 98, 48-60

Romelaer, P. et I. Huault (1996), La gestion des cadres à l'international et le modèle du Garbage Can, *Revue Française de Gestion*, 109, 19-37.

Simon, H.A. (1991), *Models of my Life*, New York: Basic Books.

Starbuck, W. (1982), Congealing Oil: Inventing Ideologies to Justify Acting Ideologies Out, *Journal of Management Studies*, 19 : 1, 3-27.

Starbuck, W. (1983), Organizations as Action Generators, *American Sociological Review*, 48 : 1, 91-102.

Von Krogh, G. et E. Von Hippel (2006), The Promise of Research on Open Source Software, *Management Science*, 52 : 7, 975-983.

Von Krogh, G., S. Spaeth et K.R. Lakhani (2003), Community, Joining, and Specialization in Open Source Software Innovation: a Case Study, *Research Policy*, 32 : 7, 1217-1241.
Weber, R.P. (1990), *Basic Content Analysis*, Newbury Park: Sage.

ANNEXE A. PRESENTATION DES SITUATIONS DE GESTION ETUDIEES

SG1. « Patch pour la version Windows », 9 jours, (23-04/01-05) 2004, 15 mails, 2 participants

- **Objet :** Mise en ligne d'un « patch » Windows qui corrige les problèmes de lancement de Grisbi sur certaines versions de Windows.
- **Description de la SG :** Le « patch » (programme permettant d'utiliser Grisbi sous Windows) a été mis en ligne par l'un des deux protagonistes. Le second lui a fait remarquer qu'il contenait encore quelques bogues et que certaines sous-parties du programme (notamment des « .exe ») manquaient. Ces deux analystes programmeurs ont ensuite coopéré sans conflit ni controverse pour résoudre ces différents problèmes et rajouter les éléments manquants sans l'aide de tiers.

SG2. « Version testing pre2 », 15 jours (6-05/20-05) 2004, 28 mails, 6 participants

- **Objet :** Distribution d'une préversion à tester donnant lieu à controverse sur les règles à suivre pour effectuer cette programmation.
- **Description de la SG :** Un nouveau programme s'appuie toujours sur une base existante, qui a été validée par la communauté des logiciels libres (on parle de sources officielles), disponible sur le site SourceForge. Or, certains analystes-programmeurs (le terme employé est celui « d'empaqueteur ») de l'équipe souhaitent utiliser d'autres sources non officielles qu'ils jugent plus pratiques et plus performantes. D'autres demandent plus de clarté en la matière, certaines situations techniques précises (correctifs effectués sur des paquets) ne faisant pas l'objet de procédures explicites. Les leaders du projet Grisbi s'y opposent. La controverse a aussi lieu sur la forme puisque l'un des empaqueteurs est choqué par l'utilisation de l'impératif par l'un des leaders du projet dans un mail *(« il n'y a pas de discussion à avoir : les empaqueteurs DOIVENT créer leurs paquets à partir des sources officielles disponibles sur SourceForge »)*. Cette utilisation de l'impératif est ressentie comme une atteinte à la liberté de cet empaqueteur, qui se caractérise lui-même comme un « bénévole volontaire ».

SG 3. « Version 050pre2 », 14 jours (6-05/19-05) 2004, 14 mails, 5 participants

- **Objet** : Controverse technique suite à la mise en ligne de la préversion 0.5.0.
- **Description de la SG :** La mise en ligne de cette nouvelle version de Grisbi a été effectuée par l'un des leaders sur la base du travail et de l'engagement d'un des analystes programmeurs du groupe. Or cette version contenait encore trop de bogues et ne contenait pas certains « paquets » nécessaires au bon fonctionnement de l'ensemble. Une controverse s'est alors développée pour savoir s'il n'aurait pas été plus judicieux d'attendre avant de mettre en ligne cette préversion et, plus généralement, si l'engagement de l'analyste programmeur n'a pas été erroné. La question est alors de savoir si, en s'engageant de la sorte, cet analyste programmeur ne s'est pas écarté des règles de fonctionnement du groupe, au delà de la communauté des logiciels libres, afin de s'approprier les bénéfices de la mise en ligne de cette version (puisque c'est lui qui l'a « signée »).

SG4. « Gestion des cartes bleues », 2 jours (21-05/22-05) 2004, 7 mails, 2 participants

- **Objet :** Problèmes dans la gestion des cartes bleues, proposition d'une solution par un développeur.
- **Description de la SG** : Les échanges portent sur une solution qui permettrait de gérer les cartes bleues à débit différé. Le problème a été mis en évidence en 2002 mais est resté en suspend. Un développeur propose une solution à l'équipe. Le manager du projet Grisbi répond à la proposition en pointant les difficultés et les différences de point de vue. Les échanges montrent un désaccord sur la solution à retenir. Le développeur propose que l'on travaille sur la question. Le manager du projet préfère différer le traitement du problème. Le sujet ne sera plus abordé par la suite.

SG5. « Paquets Mandrake », 69 jours (23-05/30-07) 2004, 61 mails, 9 participants

- **Objet :** Problèmes techniques et remarques suite à la mise en ligne des « paquets Mandrake » de la version 0.5.0.
- **Description de la SG :** Cette séquence concerne les pratiques de mise en ligne des « paquets » de la version 0.5.0 sous Mandrake. Un paquet (package) est un fichier compressé comprenant les fichiers informatiques, les informations et procédures nécessaires à l'installation d'un logiciel sur un système d'exploitation. Le packageur doit s'assurer de la cohérence fonctionnelle du système ainsi modifié. Mandrake est une « distribution Linux », c'est à dire un ensemble cohérent de logiciels rassemblant notamment un système d'exploitation et des logiciels issus du projet GNU. La mise à disposition de la seule version Mandrake 10 fait l'objet de controverses. Le packager annonce le 24 juillet 2004 qu'il abandonne le paquetage de Grisbi sous Mandrake.

SG6. « Compilation de Grisbi sous Windows », 28 jours (21-07/ 18-08) 2004, 17 mails, 5 participants

- **Objet :** Procédure à suivre pour compiler Grisbi sous Windows. En informatique, un compilateur est un programme informatique qui permet de traduire un langage (langage source) en un autre (langage cible). Ici, la compilation sert à traduire le code source de Grisbi écrit dans un langage de programmation en un autre langage permettant un fonctionnement sous l'interface graphique de Microsoft Windows.
- **Description de la SG :** Un membre de l'équipe demande des précisions sur la procédure de compilation au développeur chargé d'écrire cette procédure. Les échanges montrent qu'un retard a été pris dans la rédaction de la procédure. Durant la séquence, les échanges portent sur le recensement des éléments déjà réalisés et sur les personnes susceptibles de reprendre cette tâche afin de la terminer. Un manager du projet se propose de réaliser la tâche. Cependant, c'est le développeur initialement prévu qui termine finalement le travail.

SG7. « Préparation de la version 0.5.1. », 17 jours (25-07/10-08) 2004, 46 mails, 7 participants

- **Objet :** Mise en ligne de la pré-version de Grisbi 0.5.1., tests, problèmes et suggestions.
- **Description de la SG** : Un « project manager » met en ligne la pré-version de Grsibi 0.5.1. Il invite les développeurs à la tester et à rapporter les problèmes. Il annonce d'emblée que la version finale officielle sera publiée dans une semaine. Les autres membres de l'équipe font part de différents commentaires sur des problèmes rencontrés. Les décisions liées prises pendant cette séquence indiquent une volonté constante de reprendre des options précédentes et d'aller vers une compatibilité renforcée avec les distributions Linux sélectionnées.

SG8. « Nouvelles listes de diffusion », 9 jours (31-08/08-09) 2004, 32 mails, 8 participants

- **Objet** : Un « project manager » propose de créer deux nouvelles listes de discussion (une en français et une en anglais) pour ne pas encombrer les listes dédiées aux développeurs. Une controverse éclate alors entre différents membres de l'équipe sur le nombre de listes et les solutions techniques à retenir.
- **Description de la SG :** La proposition initiale suscite une polémique assez vive sur les solutions à retenir. Ces échanges se font entre le Web designer, les deux project managers, un développer, un packager, un contributeur régulier et deux autres non membres. Le débat se termine quand le Web Designer menace de détruire les solutions en place. *« Si des listes users sont créées, je détruis purement et simplement la hierarchie grisbi sur mon serveur et stoppe les pour-parler avec les autres newsmasters pour essayer de distribuer plus largement ces groupes. »*. La décision est prise de ne pas créer une liste de diffusion.

SG9. « Comment installer Grisbi sur Mac OS X ? », 18 jours (15-11/01-12) 2004, 23 mails, 5 participants

- **Objet :** Création d'un mode d'emploi pour installer Grisbi sur un ordinateur Mac (Apple).
- **Description de la SG :** Un utilisateur de Grisbi propose aux développeurs un mode d'emploi pour installer Grisbi sur Mac. D'autres acteurs, notamment des non-membres de l'équipe de développement, testent le mode d'emploi, font quelques propositions et soulignent des problèmes. L'initiateur du débat reconnaît alors ses erreurs et s'excuse. *« Honte à moi je n'avais pas fait de tests de bases… Vous pouvez m'élire boulet du jour! »*. Le manager de projet intervient pour nommer un responsable de l'intégration de Grisbi sous Mac.

SG 10. « Livraison de la version 0.5.4.», 22 jours (27-12-2004/17-01-2005), 29 mails, 7 participants

- **Objet :** Résolution de problèmes constatés après la livraison de la version 0.5.4.
- **Description de la SG :** Un manager du projet annonce que la version 0.5.4. est disponible. Après la mise en ligne, différents développeurs soulignent des problèmes. Des corrections sont alors proposées. Certaines sont intégrées par le manager du projet, d'autres sont repoussées à une version ultérieure. Quelques problèmes spécifiques à la version 0.5.4 sont réglés après la mise en ligne de la version 0.5.5.

SG. 11 « Etat de ma fortune nette », 3 jours (22/01-24/01) 2005, 11 mails, 5 participants

- **Objet :** Problème relatif à l'affichage des états des comptes des utilisateurs de Grisbi.
- **Description de la SG :** Au début les échanges portent essentiellement sur ce que doivent contenir ou non les états des comptes des utilisateurs : devises, soldes initiaux… La discussion tourne vite aux reproches concernant l'implication des utilisateurs de Grisbi. La décision est reportée.

SG 12. « Is Grisbi Dead ? », 12 jours (05/02-16/02) 2005, 59 mails, 17 participants

- **Objet :** Séquence remettant en cause l'existence de Grisbi.
- **Description de la SG :** Les premiers mails débutent en demandant à ce que les développeurs participent plus au projet Grisbi, puis au fur et à mesure de l'avancement, le contenu des mails dégénère pour arriver à la fin de la séquence au limogeage d'un des développeurs.