Gouvernance des projets open source

Dr Ir Robert Viseur[1 2]

1. CETIC
Avenue Jean Mermoz, 28, B-6041 Charleroi (Belgique)
robert.viseur@cetic.be

2. UMONS Faculté Polytechnique
Rue de Houdain, 9, B-7000 Mons (Belgique)
robert.viseur@umons.ac.be

RÉSUMÉ. Les logiciels open source sont utilisés par la majorité des entreprises ; certaines n'hésitent pas à jouer un rôle moteur dans leur développement. Ces pratiques amènent des défis en matière de gouvernance des systèmes d'information et des projets open source. Le fork, comme scission d'une communauté, peut être la conséquence grave d'une mauvaise gouvernance. Après un état de l'art consacré aux concepts de gouvernance open source et de fork, nous proposons deux ensembles d'études de cas, le premier sur des grands projets analysés dans la littérature scientifique, le second, sur un ensemble de forks. Sur cette base, nous développons différents moyens permettant de limiter le risque de forks, identifions quatre logiques de gouvernance open source, discutons l'intérêt des stratégies d'inner source pour organiser une transition entre les stratégies propriétaires et open source, montrons les similitudes en termes de besoins en gouvernance entre projets d'open source innovation et insistons enfin sur l'impact du degré d'ouverture de la gouvernance sur la concurrence.

ABSTRACT. The open source software are used by the majority of companies; some of them do not hesitate to play a leading role in their development. These practices result in challenges related to the governance of information systems and open source projects. The fork, as division of a community, can be the serious consequence of poor governance. After a state of the art dedicated to the concepts of open source governance and fork, we propose two sets of case studies, the first ones about major projects discussed in the scientific literature, the second ones about a set of forks. On this basis, we develop various ways to limit the risk of forks, identify four logics of open source governance, discuss the interest of inner source strategies to organize a transition between proprietary and open source strategies, show similarities in terms of needs in governance between open source software and open source innovation projects, and further stress the impact of the governance on competition.

MOTS-CLÉS : open source, inner source, gouvernance, fork.

KEYWORDS: open source, innersource, governance, fork.

Congrès INFORSID, 34ème édition, Grenoble, 31 mai – 3 juin 2016

Introduction

Le logiciel libre est défini à partir de 4 libertés: la liberté d'exécuter le logiciel, de l'étudier, d'en redistribuer des copies et de le modifier (`gnu.org`). Créé en 1998, le terme « *open source* » est parfois préféré à celui de « *free software* » (logiciel libre). Il est défini par l'OSI (`opensource.org`) sur base d'une liste de 10 critères appelée Open Source Definition (OSD), incluant notamment l'accès au code source, la liberté de redistribution, l'autorisation de créer des oeuvres dérivées et la protection du nom des auteurs. Le terme « *open source* » est associé à un type de licence logicielle, à une approche du développement logiciel, à un type de communauté et à un type de modèle d'affaires (O'Mahony, 2007). Dans la suite de ce papier, les termes « logiciel libre » et « *open source* » seront considérés comme équivalents (l'acronyme FLOSS est parfois utilisé pour éviter les débats de terminologie). Les licences de logiciels libres et *open source* respectent les quatre libertés et dix critères, qui impliquent la mise à disposition du code source du logiciel (condition nécessaire mais non suffisante).

L'organisation des projets *open source* est à l'origine considérée comme organique, qualifiée de « bazar » par comparaison à la « cathédrale » des entreprises ou des organisations historiques (Raymond, 2001). Les communautés ont longtemps été vues comme auto-organisées. En pratique, la réalité est cependant plus contrastée, les structures de gouvernance se révèlant plus ou moins complexes, en fonction de la nature des projets, de la diversité des acteurs ou des interactions avec d'autres projets. L'intérêt du secteur privé s'est aussi accru avec le temps, comme le montrent la libération du code des logiciels Netscape et le lancement du projet Mozilla en 1998 (O'Mahony, 2007 ; Viseur, 2011, 2013a), la libération de Netbeans par Sun Microsystems en 2000 (Jensen et Scacchi, 2010), les investissements d'IBM dans Linux en 2001 (West, 2003) ou, plus récemment en France, le projet Capella porté par Thales. Le virage commercial de l'*open source* a été longuement commenté par Fitzgerald (2006).

Le *fork* est un mécanisme de fractionnement d'une communauté que l'on retrouve généralement dans le domaine du logiciel libre. Certains *forks* ont fait l'objet d'une médiatisation particulière (p.ex. LibreOffice.org / OpenOffice.org). En tant qu'échec d'une collaboration dans un contexte d'innovation ouverte impliquant une communauté d'utilisateurs, le *fork* constitue un sujet d'étude concret et qui peut être instructif quant aux limites d'un système de gouvernance (p.ex. manque de gouvernance ou divergence irréconciliable de stratégies).

Notre papier est organisé en 3 sections. La première section propose un état de l'art sur le concept de gouvernance *open source* et de *fork*. La seconde section présente deux ensembles d'études de cas. Le premier se base sur des études de grands projets analysés dans la littérature scientifique ; le second, sur un ensemble d'études de *forks* célèbres dans l'histoire de l'*open source*. La troisième section discute les résultats trouvés dans les études de cas.

1. État de l'art

1.1. Gouvernance open source

La gouvernance *open source* est rarement définie ; elle est souvent associée à différentes structures, règles, pratiques et normes incluant par exemple les licences ou les processus de communication (Markus, 2007). La diversité des projets *open source* en termes de tailles, de maturité, de licences ou de culture permet par ailleurs de supposer des mécanismes de gouvernance fort différents. La complexité du concept s'est par ailleurs encore accrue avec le développement de l'*open source* au delà du logiciel : *open content*, *open data*, *open hardware*,... Cette extension a été décrite et qualifiée d'*open source innovation* par Raasch (2009) puis par Pénin (2012), qui en a plus longuement défini le concept.

Markus (2007) définit la gouvernance *open source* comme l'ensemble des moyens mis en œuvre pour l'orientation, le contrôle et la coordination d'organisations et d'individus totalement ou partiellement autonomes pour le compte d'un projet de développement *open source* auquel ils contribuent collectivement.

La gouvernance d'un projet ou d'un ensemble de projets se structure généralement progressivement en suivant trois phases (de Laat, 2007). Dans une première phase du projet, la gouvernance est informelle et liée aux règles fixées par la licence (p.ex. GPL). Le *leadership* est exercé par des développeurs sur base de leurs performances (méritocratie). Dans une seconde phase, dès lors que la taille du projet augmente, un ensemble d'outils formels et explicites sont mis en place. Il s'agit de la modularisation du code source, de la division des rôles et de la délégation de la prise de décision, de la formation et la transmission des valeurs, de la formalisation des procédures et du régime d'exercice du pouvoir (autocratique ou démocratique). Dans une troisième phase, le projet, confronté au monde extérieur, doit assurer sa pérennité sur les plans matériel, financier et légal. Cela passe notamment par l'institutionnalisation des projets et la création de fondations (p.ex. Apache, Debian ou Mozilla).

La gouvernance d'un projet *open source* peut s'analyser sur plusieurs niveaux (Jensen et Scacchi, 2010). Le premier niveau d'analyse est *micro*. Il concerne les participants individuels au projet (p.ex. actions, ressources et interactions). Le second est *meso*. Il concerne les équipes de projets (p.ex. collaboration, *leadership*, contrôle et résolution de conflits). Le troisième est *macro*. Il concerne les écosystèmes inter-projets (p.ex. collaboration, autorité, contrôle et résolution de conflits).

Markus (2007) propose un ensemble d'éléments permettant de caractériser la gouvernance d'un projet *open source* : la propriété des actifs (p.ex. *copyrights*, licences ou marques), les objectifs du projet (p.ex. charte et vision), la gestion de la communauté, le processus de développement logiciel (p.ex. identification des besoins et affectation des tâches), la résolution de conflits et le changement de règles, et l'utilisation de l'information et des outils (p.ex. modalités d'accès aux outils et aux répertoires de code source).

Les licences *open source* constituent un des moyens de gouvernance. Ces licences sont généralement classées en deux grandes familles : les licences *copyleft* (dites aussi gauches d'auteur ou réciproques) et les licences permissives (dites aussi académiques ou non *copyleft*) (Alspaugh *et al.*, 2009 ; Fitzgerald, 2006 ; Lerner et Tirole, 2005 ; Montero *et al.*, 2005 ; Muselli, 2008). Une licence permissive (p.ex. BSD ou MIT) autorise l'utilisateur à placer le programme sous une nouvelle licence, libre ou même propriétaire (caractère appropriable). Une licence copyleft (p.ex. LGPL, GPL, AGPL ou MPL) « *lie l'octroi des droits à l'obligation de ne redistribuer le logiciel et ses modifications que sous la même licence que celle par laquelle le licencié a obtenu ces droits* » (Montero *et al.*, 2005). Elle confère dès lors au logiciel un caractère inappropriable. On parlera de copyleft faible, lorsque le *copyleft* s'applique uniquement à un composant logiciel, ou de *copyleft* fort, dans le cas où toute œuvre dérivée doit adopter la licence *copyleft* du composant logiciel (caractère contaminant).

Différents niveaux d'ouverture sont possibles au sein des communautés *open source*. Apache apparaît par exemple comme une communauté structurée avec une gouvernance transparente et ouverte. À l'opposé, le modèle *open core*, où seul un noyau générique et quelques modules sont *open source* (p.ex. Zenoss), le logiciel complet étant essentiellement propriétaire, laisse peu de place à la communauté (Viseur, 2013c). Aucune activité communautaire n'est d'ailleurs attendue et le pouvoir est concentré entre les mains de l'éditeur *open core*. L'Open Governance Index permet en pratique de quantifier le degré d'ouverture d'un projet en termes de transparence, de prise de décision, de réutilisation et de structure communautaire (Laffan, 2011, 2012). Cet indice comporte 13 métriques relatives à 4 domaines de gouvernance : l'accès au code source, le processus de développement, la création d'œuvres dérivées et la communauté.

Les pratiques *open source* ont connu une traduction en entreprise pour le développement de composants internes. Ce modèle est qualifié d'*inner source* (Stol *et al.*, 2011). Deux mises en œuvre sont distinguées : l'*inner source* basé infrastructure et l'*inner source* basé projet. Dans le premier cas, des porteurs de projets individuels sont invités à mettre à disposition leur projet sur une infrastructure de développement mutualisée inspirée par les plates-formes de développement *open source*. Dans le second cas, une équipe de développement interne prend en charge, dans un but de mutualisation, le développement et le support de composants critiques exploités dans différentes lignes de produits commercialisées par l'entreprise.

1.2. Fork

Bar et Fogel (2003) définissent le *fork* comme une situation se produisant « *lorsqu'un groupe de développeurs prend le code d'un projet de logiciel libre pour en démarrer un autre* ». Nyman et Mikkonen (2011), dans une étude de 566 projets hébergés sur Sourceforge.net et présentés par leurs administrateurs comme des *forks*, identifient des motivations à *forker* classables en quatre catégories : les motivations techniques (ajout de caractéristiques, spécialisation, portage, amélioration), les

changements de licence, les adaptations locales (langues ou particularités régionales) et la relance de projets abandonnés. Wheeler (2007) relativise la dangerosité présumée du *fork* et l'associe à un mécanisme de saine compétition. Il le compare au principe de la motion de censure dans un parlement ou à une grève. Le *fork* permettrait à la communauté des développeurs d'attirer l'attention des *leaders* sur des demandes non prises en compte. Nyman *et al.* (2011b) y voient même une « *main invisible* » garante du caractère durable et de la continuité des projets. La capacité à *forker* garderait par ailleurs « *les communautés vivantes, et les entreprises honnêtes* » (Moody, 2009). Elie (2006) voit dans le *fork* « *un droit essentiel* » mais insiste aussi sur le « *risque de se couper en même temps de la richesse du tronc commun* ». Il y voit souvent la conséquence de « *systèmes de régulation mal définis* ». Le mérite relèverait moins dans les communautés de logiciels libres de la compétence technique que du charisme et de la capacité à vivre dans le conflit. Notons que, sur le plan technique, le *fork* fait aussi partie des pratiques normales de certains outils de gestion de code source (p.ex. Git) ou de certains hébergeurs de projets (p.ex. Github).

2. Études de cas

2.1. Android, Apache, Mozilla, MySQL et Netbeans

Différents projets ont fait l'objet d'études sur la gouvernance ou sur les facteurs de succès (incluant des questions de gouvernance). Nous retiendrons les cas d'Android (Laffan, 2011, 2012), d'Apache HTTP Web Server (Viseur, 2016b), de Mozilla (Viseur, 2013a), de MySQL (Välimäki, 2003) et de Netbeans (Jensen et Scacchi, 2010). Ces derniers bénéficient d'études de cas déjà publiées dans la littérature scientifique. Ils couvrent une large diversité de projets et permettent une compréhension des mécanismes de gouvernance mis en œuvre, en termes notamment de prise de décision, de contribution ou de licence.

2.1.1. Android

Le système d'exploitation Android (`www.android.com`) a été lancé en novembre 2007. Il marquait l'entrée de Google sur le marché du mobile. Il est soutenu par l'Open Handset Alliance (`www.openhandsetalliance.com`), un consortium d'entreprises actives dans les technologies mobiles. La publication sous licence *open source* Apache a été réalisée en octobre 2008, parallèlement au lancement du téléphone HTC G1. La part de marché du nouvel OS a rapidement grandi par la suite, atteignant près de 80% du marché (source IDC, Q4 2014). Laffan (2012) souligne le caractère fermé du projet Android, avec un Open Governance Index de 23%, à comparer à Symbian (58%) et Meego (61%). Laffan met en particulier en évidence les processus de décision unilatéraux, le processus fermé d'accès au code source (« *code committer* »), le processus fermé de contribution au code source, l'opacité du processus de contrôle et de prise de décision autour de l'Android Compliance Program ou encore l'absence de volonté d'évoluer vers un

modèle de gouvernance plus ouvert. Des variantes de l'Android officiel existent (p.ex. CyanogenMod).

2.1.2. Apache HTTP

Le projet de serveur Web Apache HTTP (`httpd.apache.org`) a été démarré en 1995 sur base du code source du serveur du NSCA par un groupe de développeurs géographiquement distribués, connus sous le nom d'Apache Group. Il est publié sous licence Apache. En 1999, les membres de l'Apache Group ont fondé l'Apache Software Foundation, dont l'objectif était de fournir un support financier, légal et organisationnel au serveur Apache HTTP. La Fondation a par la suite évolué vers une structure multi-projets et a acquis le sponsoring d'entreprises IT de taille mondiale comme Hewlett-Packard, IBM et Microsoft. Elle se distingue notamment par sa structure d'incubation, permettant l'entrée de logiciels novateurs dans le portefeuille de projets et leur accès aux outils de gestion des projets (p.ex. gestionnaires de sources et gestionnaires de bugs). L'incubateur différencie nettement les logiciels matures, de haute qualité, et les nouveaux projets devant encore faire leurs preuves. Il permet ainsi un renouvellement dans la production logicielle (innovation) sans nuire à la réputation des logiciels de référence.

2.1.3. Mozilla

La Fondation Mozilla (`www.mozilla.org`) a été créée en juillet 2003 sur les cendres de la société Netscape, rachetée en mars 1999 par AOL Time Warner. Elle est à l'origine du logiciel Mozilla, issu de la libération de technologies de la société Netscape en 1998. Elle prend aujourd'hui en charge le développement du navigateur web Firefox ainsi que d'autres projets d'outils de développement ou de logiciels pour les utilisateurs finaux. Un important travail de réécriture et de modularisation du code source fourni initialement par Netscape ainsi qu'un travail sur l'ergonomie du logiciel ont permis à Firefox d'atteindre une part de marché autour des 25%. La communauté s'appuie sur une organisation assez hiérarchisée, incluant par exemple l'existence de responsables de modules (« *module owners* »), à l'origine de fréquents conflits. La licence MPL, écrite pour le projet afin de faciliter l'agglutination de codes sources sous des licences différentes et de protéger le travail des contributeurs, a changé à plusieurs reprises, en raison de réactions communautaires ou de dépendances à d'autres projets, selon un mode souvent collaboratif.

2.1.4. MySQL

MySQL (`www.mysql.com`) est une base de données développée depuis 1995. Le logiciel a été placé sous licence GPL en 2000, tandis que la société MySQL a été créée en 2001 pour valoriser la technologie. L'entreprise applique un modèle de double licence : une version *open source* cohabite avec une version commerciale (i.e. payante). L'entreprise tire dès lors ses revenus de la prestation de services mais également du paiement de licences (plus de 50 % de son chiffre d'affaires en 2003) par les clients achetant la version propriétaire. L'existence des deux branches du logiciel est garantie par la totale propriété du code source. Toutes les contributions sont à cette fin vérifiées et réécrites par l'entreprise. MySQL AB a dû faire face à un

fork suite au rachat de MySQL AB par Sun Microsystems puis de ce dernier par Oracle Corporation. Ce *fork* s'appelle MariaDB (`mariadb.org`) et a été initié par Michael Widenius, fondateur de MySQL ; il bénéficie du support de plusieurs entreprises importantes, dont Google, et est soutenu par un consortium baptisé Open Database Alliance (`www.opendatabasealliance.com`).

2.1.5. Netbeans

Netbeans (`netbeans.org`) est un projet d'environnement de développement pour le langage Java, mis en *open source* sous licence CDDL par Sun Microsystems en 2001. Le projet offre une large autonomie aux développeurs. Par contre, Sun Microsystems a conservé pendant longtemps une empreinte forte sur les structures décisionnelles chapeautant l'organisation. Un projet de fusion avec son principal concurrent *open source*, Eclipse, sponsorisé par IBM, a échoué pour cause de différences techniques et organisationnelles entre les deux sponsors ainsi que de risque de perte d'image (*leadership* technologique). L'ascendant a finalement été pris par Eclipse.

2.1.6. Synthèse

Les projets dont les caractéristiques sont synthétisées dans cette section présentent une grande diversité en termes de licences (licences Apache, licence GPL, licence MPL, licence CDDL, double licence,...), de modèles économiques (diverses formes d'édition logicielle et/ou de mutualisation des développements), d'organisation et d'évolution du rapport à la communauté (contrôle par un éditeur, autonomisation progressive d'une structure de mutualisation ouverte à la communauté,...). Les utilisateurs et contributeurs semblent rencontrer davantage de difficultés à faire accepter leurs codes sources ou à intégrer les organes de décision sur base du mérite lorsqu'un projet est contrôlé par une entreprise, ce qui peut se traduire par des *forks* (Android, MySQL).

2.2. Forks

2.2.1. Méthodologie

Plusieurs *forks* ont fait l'objet d'études plus ou moins approfondies dans la littérature. Citons la famille des systèmes d'exploitation BSD (Weber, 2004), Roxen (Dahlander et Magnusson, 2008), GCC (Fogel, 2004), CVS (Bart et Fogel, 2003), Spip (Elie, 2006) ou LibreOffice.org (Gamalielsson et Lundell, 2012). Ces résultats seront exploités. Nous avons procédé à l'étude de 26 *forks* de projets populaires ; ces projets sont 386BSD, Claroline, Compiere, CVS, Dokeos, Ext JS, FreeBSD, GCC, GNU Emacs, KHTML, Mambo, MySQL, NCSA HTTPd, NetBSD, OpenERP, OpenOffice.org, PhpGroupware, PhpNuke, Qt, RHEL, Roxen, Samba, Sodipodi, Sourceforge, Spip et Xfree86. Nous avons exploité des documents existants : livres, articles scientifiques, articles de presse, actualités postées sur des portails informatiques ou *open source*, communiqués et pages de sites de projets,... Chaque *fork* a fait l'objet d'une fiche descriptive, reprenant la chronologie du *fork*, les auteurs

impliqués et leurs motivations. Les résultats ont été synthétisés au sein d'un tableau récapitulatif, reprenant le nom du projet initial, le nom du *fork*, la (ou les) motivations(s) du *fork* et l'impact du *fork* sur le projet original (rapport technique interne). L'impact a été évalué au sens des issues possibles identifiées par Wheeler (2007).

2.2.2. Motivations à déclencher un fork

Sept motivations à *forker* ont été identifiées : l'arrêt du projet original (19%), les motivations techniques - nouvelle spécialisation, vues techniques divergentes ou objectifs techniques différents - (42%), le changement de licence (15%), le conflit autour de la propriété d'une marque (12%), les problèmes de gouvernance du projet (38%), les différences culturelles fortes (8%) et la recherche de nouvelles pistes d'innovation (4%).

En pratique, les études des *forks* qui précèdent montrent que les *forks* qui réussissent (et sont donc susceptibles de porter préjudice à l'éditeur original, lorsqu'il y en a un) démarrent généralement pour une raison importante. L'arrêt du support d'un produit libre populaire entraîne souvent un *fork* (cf. NCSA HTTPd, 386BSD, Red Hat Linux et Roxen). Le *fork* libre réussit généralement mais peut cohabiter avec une version fermée du produit (cf. Red Hat Linux, Roxen, Sourceforge). Un *fork* peut intervenir suite à l'apparition de divergences techniques. Les systèmes *BSD ont ainsi souvent adopté des spécialisations techniques distinctes (portabilité, sécurité,...). C'est la cause la plus fréquente (42%). La gouvernance du projet apparaît à l'origine du conflit pour environ un tiers des cas étudiés (38%). Le problème porte généralement sur le manque d'ouverture des équipes de développement : prise en compte des contributions extérieures (cf. OpenOffice.org), discussion des objectifs du projet (cf. Sodipodi), réticences face à un mode de développement communautaire (cf. OpenOffice.org, Dokeos, PHP Nuke),... La propriété de la marque apparaît également comme une source de conflit (cf. Claroline, Mambo et OpenOffice.org). Elle peut être liée à la question de la gouvernance (cf. propriété des actifs) car la marque permet en pratique à l'éditeur de conserver un contrôle sur l'évolution du projet. La marque cristallise dès lors les tensions entre un sponsor (p.ex. éditeur *open source*) et sa communauté. Le problème de licence apparaît parfois à l'origine d'un *fork*, qu'il n'affecte pas le type de licence (cf. Xfree86) ou qu'il entraîne au contraire une augmentation (cf. Ext JS) ou une réduction (p.ex. arrêt de la branche libre) de la liberté du logiciel. La mise sous GPL ou AGPL facilite par ailleurs les échanges entre projets, dès lors que la licence d'origine peut difficilement être changée. Le changement de licence n'est cependant pas un motif dominant à *forker* (15%). Les *forks* qui ont été suscités par Theo de Raadt, leader d'OpenBSD, peuvent être justifiés, au moins en partie, par des prises de position politiques ou idéologiques. Cette configuration paraît finalement assez marginale dans le paysage *open source*. Les chocs culturels -communauté vs entreprise (cf. KHTML), communauté vs administration (cf. Spip)- apparaissent par contre comme une cause possible (8%), illustrant la difficulté à aligner des stratégies communautaires et commerciales.

2.2.3. Conséquences des forks

Les études montrent que les *forks* qui se traduisent par une disparition totale du projet d'origine ne sont pas majoritaires (19%). À l'exception du serveur Apache, de X.Org, de Joomla ou d'Inkscape, une cohabitation apparaît dans plus de la moitié des cas étudiés (54%). Dans certains cas, les échanges de codes continuent d'ailleurs (cf. FreeBSD, NetBSD, OpenBSD). Une fusion ultérieure des projets (cf. GCC et EGCC) est possible. La divergence progressive peut par contre rendre la fusion malaisée (cf. Webkit et KHTML). L'échec total du *fork* intervient dans moins d'un cas sur cinq (19%).

3. Discussion

Nous identifions ci-dessous quatre logiques de gouvernance *open source* ainsi que différents moyens permettant de gérer le risque de *forks*, discutons ensuite l'intérêt des stratégies d'*inner source* pour organiser une transition entre les stratégies propriétaires et *open source*, montrons les similitudes en termes de besoins en gouvernance entre projets d'*open source innovation* et insistons enfin sur l'impact du degré d'ouverture de la gouvernance sur la concurrence.

3.1. Logiques de gouvernance

Plusieurs logiques de gouvernance émergent des différents cas traités dans cette recherche.

3.1.1. Logique individuelle

Par exemple : majorité des petits projets hébergés sur GitHub (`github.com`). Elle s'applique à la majorité des projets *open source* publiés, maintenus par une personne sur des dépôts publics. L'autorité y est exercée de manière informelle par l'auteur du logiciel. Les procédures de travail ne sont pas formalisées. La communauté est souvent petite et les contributions sont rares, réduisant la vitalité du projet mais aussi les sources de conflit.

3.1.2. Logique commerciale

Par exemple : MySQL ou Zenoss. Elle s'applique aux projets d'édition *open source* menés par des entreprises privées. Il s'agit généralement de projets valorisés selon le modèle de la double licence, associant une version communautaire du logiciel sous licence réciproque et une version commerciale payante techniquement différenciée. L'entreprise peut éventuellement chercher le soutien de la communauté (dans le cas du modèle *open core* la communauté est inexistante) mais veille à garder le contrôle sur le projet, notamment par la signature d'accords de contributeurs organisant le partage des droits patrimoniaux (Poo-Caamaño et German, 2015 ; Valimaki, 2003).

3.1.3. Logique communautaire

Par exemple : Apache HTTP ou Chamilo. Elle s'applique aux projets de grande taille, dont le succès a nécessité la mise en place d'une structure de soutien (p.ex. fondation ou association), permettant d'en assurer la pérennité, et la mise en place de procédures organisationnelles. La prise de décision est ouverte aux membres de la communauté sans volonté d'entraver la prise de responsabilité liée au mérite.

3.1.4. Logique industrielle

Par exemple : Eclipse ou Netbeans. Elle s'applique aux projets développés par des acteurs industriels ayant pour objectif une mise en commun de l'effort de développement (coopétition). La gouvernance peut paraître similaire à celle des grands projets à logique communautaire structurés en fondation. Elle peut cependant différer à la marge, par exemple par l'imposition de droits d'inscription limitant l'accès aux fonctions dirigeantes et permettant aux grandes entreprises de conserver un contrôle sur les objectifs du projet.

3.2. Gouvernance et inner source

Les logiques de gouvernance identifiées *supra* peuvent être rapprochées des différents modes d'*inner source*. L'*inner source* basé infrastructure peut ainsi être comparé à la logique individuelle (porteurs de projets individuels profitant d'une infrastructure de développement mutualisée) ; l'*inner source* basé projet, à la logique industrielle (objectif de mutualisation d'actifs critiques).

Un des défis auquel les entreprises font face réside dans la mise en *open source* des projets internes. Dès lors que les projets sont anciens, leur libération influence les modèles d'affaires et les stratégies mises en œuvre par l'entreprise (West, 2003). Sur le plan patrimonial, l'organisation d'une transition depuis une stratégie propriétaire vers une stratégie *open source* par le recours à des licences hybrides a déjà été appliquée par Sun Microsystems (Community Source et licence SCSL). Ce cas a fait l'objet d'une étude par Muselli (2007). L'application des principes de l'*inner source*, basé infrastructure ou basé projet, à l'intérieur de l'entreprise, pourrait également servir de transition vers une externalisation complète en *open source*. *Inner source* et *open source* partagent en effet un ensemble de préoccupations communes quant à la motivation des contributeurs ou de support aux utilisateurs du composant logiciel. Ce mode d'évolution pourrait cependant entraîner des habitudes de contrôle sur le processus de développement susceptibles d'ultérieurement provoquer des conflits avec la communauté *open source*.

3.3. Causes de forks liés à la gouvernance

Cette section compare les résultats des études de cas avec l'étude précédente de Nyman et Mikkonen (2011), discute ensuite la réduction du risque de *fork* et examine enfin du lien entre les *forks* et la logique de gouvernance.

3.3.1. Étude de Nyman et Mikkonen (2011)

Comparé à l'étude de Nyman et Mikkonen (2011), notre recherche regroupe plusieurs motivations sous le label de « motivation technique » et met en évidence trois causes supplémentaires : les questions de gouvernance, les difficultés liées aux différences culturelles et les conflits liés à la propriété d'une marque. Ceci étant, les changements d'orientation technique occupent aussi une place prépondérante dans notre étude. La reprise de projets arrêtés apparaît plus importante. Ces différences peuvent s'expliquer par le spectre plus large des causes prises en compte mais aussi par la nature différente des projets : Nyman et Mikkonen (2011) se basent sur un ensemble des projets pris sur Sourceforge.net, qui héberge de nombreux projets de taille réduite, alors que avons basé notre étude sur des projets populaires ayant déjà une communauté active, jouant un rôle de régulation et de responsabilisation.

3.3.2. Architecture de participation

Le risque de *fork* pour divergences techniques est élevé. Il peut cependant être limité en adoptant dès le départ une architecture modulaire. MacCormack *et al.* (2006) parlent d'architecture de participation, et y voient un moyen pour simplifier la compréhension du code et les contributions. Le modèle noyau-extension en est un exemple, permettant l'adaptation du logiciel sans en toucher le cœur. L'éditeur garantit alors les performances d'un noyau incorporant des fonctionnalités générales, tandis qu'intégrateurs et utilisateurs avancés en étendent les fonctionnalités par le développement d'extensions. Cette approche peut également limiter les conflits liés à l'organisation de l'équipe de développement puisque les intégrateurs ne doivent comprendre que les interfaces du logiciel permettant le développement d'extensions. La compréhension des spécificités du noyau n'est pas indispensable. Des conflits peuvent par contre apparaître dès lors que des extensions communautaires entrent en conflit avec des extensions propriétaires vendues par l'éditeur. Le modèle noyau-extension, avec la création d'interfaces de programmation d'applications (API) qu'elle sous-tend, rappelle les « *user toolkits for innovation* » décrites par Von Hippel (2001). Ces boîtes à outils permettent, sur des marchés composés de clients aux besoins hétérogènes, une forme d'externalisation vers les clients des tâches d'innovation nécessitant une connaissance pointue des besoins des clients. Le bénéfice attendu est une meilleure satisfaction des besoins et, dans un projet logiciel libre, un moindre risque de tensions autour des orientations du projet. Franke et Von Hippel (2003) considèrent par défaut les logiciels *open source* comme un « *user toolkit* » du fait de l'accès au code source ; cependant, l'existence d'Interfaces de Programmation d'Applications (APIs) documentées abaisse les barrières à la participation sur le projet *open source*.

3.3.3. Structures d'expérimentation

Samba illustre le coté « tueur d'innovation » de l'obligation de qualité liée à une importante base d'utilisateurs. Cet exemple souligne l'intérêt des incubateurs, tels que celui d'Apache, permettant l'expérimentation en marge du projet principal. D'une certaine manière, le *fork* Samba TNG joue d'ailleurs un rôle d'incubateur. Un

effet similaire peut être obtenu par la création de branches expérimentales au sein du dépôt de sources (cf. Linux par exemple).

3.3.4. Différences de culture

Les différences de culture entre membres d'une même communauté pourraient également constituer un motif de *fork*, dont la prévention mériterait des recherches plus approfondies (l'effet de ce type de *fork* n'est cependant pas obligatoirement négatif, s'il permet aux communautés de sortir des conflits et de recentrer les efforts sur le développement des logiciels). L'exemple de Claroline est parlant de ce point de vue. Dominé par des institutions d'enseignement, ce projet a finalement *forké* à deux reprises, la première avec Dokeos (entreprise), la seconde avec Chamilo (communauté / association).

3.1.5. Forks et logiques de gouvernance

La logique individuelle est généralement portée par une seule personne ; cette dernière joue un rôle de « dictateur bienveillant » et peut conduire à un *fork* du fait de divergences avec la communauté (p.ex. Sodipodi). La logique commerciale peut conduire à une opposition entre les objectifs de la communauté et celle de l'entreprise qui porte le projet, conduisant à la rupture (p.ex. MySQL). La logique industrielle peut se traduire par la mainmise de grands groupes sur le projet et conduire à une fronde des communautés dont les contributions sont handicapées (p.ex. OpenOffice.org). La logique communautaire est naturellement tournée vers les contributeurs extérieurs ; elle n'élimine cependant pas les risques de *forks*, notamment amicaux (p.ex. Samba ou la famille BSD). En pratique, aucune logique de gouvernance ne semble à même d'éliminer le risque de *fork* pour l'organisation qui porte le projet. De plus, les projets adoptent souvent des structures hybrides, qui rendent difficile l'identification des causes et de leurs effets. C'est par exemple le cas pour des éditeurs *open source* encourageant par ailleurs la création d'une communauté active de développeurs (p.ex. OpenERP/Odoo). Dans ce cas, la structuration progressive de la logique commerciale, suite par exemple à l'arrivée d'investisseurs, peut favoriser les tensions et conduire au *fork* (Viseur, 2013c).

La perception négative du *fork* en tant que risque mérite par ailleurs réflexion. Cette dernière provient essentiellement de *forks* inamicaux, fruits de tensions durables, conduisant à une scission de la communauté et à une perte en termes de mutualisation. Cette situation peut cependant constituer un processus salvateur d'auto-régulation en cas de mauvaise gestion du projet ou de divergence irréconciliable. Par ailleurs, certains *forks* apparaissent amicaux, conduisant à des choix techniques différents, parfois avec des échanges de code source (p.ex. famille BSD), ou à des expérimentations en marge du projet officiel (p.ex. Samba TNG).

3.4. Gouvernance au delà du logiciel

L'extension des pratiques *open source* au matériel conduit à des similitudes avec le domaine logiciel, tant pour les licences que pour les modèles d'affaires ou les

modèles de développement (Viseur, 2012). Dans le domaine des véhicules *open source*, par exemple, les licences réciproques (CC-BY-SA) et le modèle de la double licence sont expérimentés. Les logiques individuelles (p.ex. Wikispeed) ou commerciales (p.ex. OSVehicle) y co-existent, tandis que la faible maturité de ces projets semble ne pas encore avoir conduit à la création d'organisations structurant l'activité (Viseur, 2016a).

La problématique du *fork*, bien que moins fréquente, est également connue, comme le montre le conflit récent autour du projet Arduino (Orsini, 2015). En pratique, les cartes de prototypage Arduino sont depuis longtemps concurrencées par des cartes équivalentes : Freeduino, Sanguino, Seeduino,... Depuis 2015, cependant, un conflit entre fondateurs du projet autour du paiement de *royalties* et de l'exploitation de la marque Arduino a conduit à la cohabitation de deux entreprises concurrentes : Arduino SRL (ex-SmartProjects) et Arduino LLC. Cette dernière exploite la marque Genuino (hors USA).

3.6. Gouvernance et concurrence

En 1998 déjà, Eric Raymond soulignait la tendance de certains projets *open source* à occuper la totalité de la niche fonctionnelle qu'ils occupaient : « *Some very successful projects become `category killers'; nobody wants to homestead anywhere near them because competing against the established base for the attention of hackers would be too hard. People who might otherwise found their own distinct efforts end up, instead, adding extensions for these big, successful projects.* ». Or, les composants *open source* sont aujourd'hui massivement utilisés dans les développements logiciels. Ainsi, une étude de Black Duck Software (« *2015 Future of Open Source Survey Results* ») révélait que 78% des personnes interrogées basent leurs activités commerciales sur des logiciels *open source* et que les deux-tiers construisent des logiciels pour leurs clients qui sont eux-mêmes basés sur du logiciel *open source*. Le poids de certains projets tend à inquiéter les autorités de régulation de la concurrence, comme en Europe avec Android (82,8% de part de marché mondiale au Q2 2015 selon IDC), caractérisé par une gouvernance opaque : « *The Commission's in-depth investigation will focus on whether Google has breached EU antitrust rules by hindering the development and market access of rival mobile operating systems, applications and services to the detriment of consumers and developers of innovative services and products* » (EU, 2015).

Une gouvernance *open source* plus fermée représente donc potentiellement une menace pour la concurrence. De grandes entreprises adoptent maintenant le modèle *open source* (p.ex. Google avec Android ou Microsoft avec .Net). Ces entreprises disposent d'un pouvoir économique certain et l'*open source* peut les aider à disséminer davantage leurs technologies et standards. Adatto (2013) montre en effet comment la mise à disposition d'une implémentation de référence peut faciliter la diffusion d'une innovation. La modulation de l'ouverture au cours du temps peut dès lors conduire à des situations de pouvoir économique au travers de technologies et standards largement disséminés. Dans ce contexte, le *fork* apparaît comme un mécanisme d'auto-régulation susceptible de rétablir de la concurrence dans un

marché où elle est menacée par un grand groupe industriel. La disponibilité du code source d'Android permet ainsi l'émergence d'intégrateurs alternatifs (p.ex. Cyanogen) et d'éditions spécifiques à un constructeur (p.ex. Xiaomi). Les entreprises peuvent en effet récupérer et *forker* le code source, pour ensuite l'adapter en toute autonomie aux besoins de leurs propres clients. Les versions *forkées* d'Android représenteraient ainsi environ 20% de l'écosystème Android.

4. Conclusion

Cette recherche aura permis différents apports aux questions de gouvernance *open source*. Premièrement, la question de la transition d'une stratégie propriétaire classique vers une stratégie *open source* préoccupe depuis longtemps les entreprises. La question a déjà été traitée par Muselli pour les aspects juridiques (2007) ; cette recherche propose une exploitation de l'*inner source* comme stratégie progressive de transition vers l'*open source*, permettant à l'entreprise d'acquérir les outils et les expertises nécessaires à la conduite d'un projet *open source*. Deuxièmement, nous avons identifié plusieurs logiques de gouvernance, permettant à l'entreprise de rapidement identifier le degré de structuration et le type de gouvernance nécessaires en fonction de la maturité d'un projet *open source*. Troisièmement, nous avons identifié un ensemble de causes pour les *forks* et avons proposé différents moyens pour en réduire le risque de survenance (p.ex. mise en place d'une architecture modulaire ou de structures d'expérimentation). Quatrièmement, nous avons montré que les problématiques de gouvernance et les risques de *forks* se retrouvaient également au sein de projets d'*open source innovation* (p.ex. *open hardware*). Il est ainsi possible, pour les projets d'*open hardware*, d'exploiter l'expertise accumulée dans le passé avec les projets *open source*. Cinquièmement, nous avons montré que la gouvernance, loin de n'agir que sur les utilisateurs et les développeurs du projet, jouait également un rôle sur la concurrence dans le secteur ICT.

Cette recherche a permis d'identifier trois sujets méritant un approfondissement. Premièrement, l'adoption d'une architecture modulaire, favorisant la participation au projet et la réutilisation au sein d'autres projets, limite les possibilités d'innovation architecturale. L'organisation des changements d'interface (protocole, spécifications,...) passe par la gouvernance inter-projet. Les modalités pratiques de cette dernière mériteraient cependant d'être approfondies. Deuxièmement, l'innovation au sein des projets apparaît comme un problème fondamental, tant pour éviter le risque de *fork* que pour tirer pleinement parti de la mutualisation des développements et de la diversité des contributeurs. Cela peut passer par des mécanismes souples de gestion de sources et le développement de pratiques managériales encourageant une forme d'intraprenariat (comportement d'entreprenariat interne à l'entreprise) sur les projets ou écosystèmes auxquels les entreprises contribuent (Viseur et Pinchart, 2013b). L'encouragement de la prise d'initiative en matière de contribution ou de création de nouveaux projets s'accompagne d'enjeux pour l'entreprise, par exemple en termes de politiques internes de propriété intellectuelle. Troisièmement, le caractère *open source* d'un projet ne doit pas masquer son impact potentiellement problématique sur l'état de la

concurrence dans le secteur ICT. L'utilisation d'un indice de concentration pondéré par le degré d'ouverture de la gouvernance peut aider à identifier les projets majeurs potentiellement problématiques en termes de concurrence (Viseur, 2016c).

5. Références

Adatto, T. (2013), Standards ouverts et implémentations FLOSS (Free Libre Open Source Software): vers un nouveau modèle synergique de standardisation promu par l'industrie du logiciel, in Terminal : Technologie de l'Information, Culture, Société, n°113-114, pp. 137-170, 2013.

Alspaugh, T.A., Asuncion, H.U., Scacchi, W. (2009), Intellectual property rights requirements for heterogeneously-licensed systems, 17th IEEE International Requirements Engineering Conference (RE'09), pp. 24–33, Augustus 31 - September 4, 2009.

Bar, M., Fogel, K. (2003), Open Source Development with CVS, Paraglyph Press.

Dahlander, L., Magnusson, M., How do firms make use of open source communities?, Long Range Planning, vol. 41, n°6, December 2008, pp. 629-649.

De Laat, P. B. (2007). Governance of open source software: state of the art. Journal of Management & Governance, 11(2), pp. 165-177.

Elie, F. (2006), Économie du logiciel libre, Eyrolles.

EU (2015), Antitrust: Commission sends Statement of Objections to Google on comparison shopping service; opens separate formal investigation on Android, 15 April 2015. Online: http://europa.eu/rapid/pressrelease_IP154780_en.htm (lu : 17 janvier 2016).

Fitzgerald, B. (2006), The transformation of open source software. Mis Quarterly, 587-598.

Fogel, K. (2004), How To Run A Successful Free Software Project - Producing Open Source Software, CreateSpace.

Franke, N., Von Hippel, E. (2003), Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software. Research policy, 32(7), pp. 1199-1215.

Gamalielsson, J., Lundell, B. (2012), Long-term sustainability of Open Source software communities beyond a fork: a case study of LibreOffice. In Open Source Systems: Long-Term Sustainability (pp. 29-47). Springer Berlin Heidelberg.

Jensen, C., Scacchi, W. (2010), Governance in open source software development projects: A comparative multi-level analysis. In Open Source Software: New Horizons (pp. 130-142). Springer Berlin Heidelberg.

Laffan, L. (2011), Open governance index-Measuring the true openness of open source projects from Android to WebKit, VisionMobile, London. Online : https://upload.wikimedia.org/wikipedia/commons/5/5f/VisionMobile_Open_Governance_Index_report.pdf (lu le 16 janvier 2016).

Laffan, L. (2012), A new way of measuring openness: The open governance index. Technology Innovation Management Review, 2(1). Online : http://timreview.ca/article/512 (lu : 16 janvier 2016).

Lerner, J., Tirole, J. (2005), The Scope of Open Source Licensing, Journal of Law, Economics, and Organization, vol. 21, issue 1, 2005, pp. 20-56.

MacCormack, A., Rusnak, J., Baldwin, C.Y. (2006), Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code, Management Science, vol. 52 (7), 2006, pp. 1015-1030.

Markus, M. L. (2007), The governance of free/open source software projects: monolithic, multidimensional, or configurational?. Journal of Management & Governance, 11(2), pp. 151-163.

Montero E., Cool Y., de Patoul F., De Roy D., Haouideg H., Laurent P., (2005), Les logiciels libres face au droit, Cahier du CRID, n°25, Bruylant, 2005.

Moody, G. (2009), Who Owns Commercial Open Source – and Can Forks Work?, Linux Journal, April 2, 2009.

Muselli, L. (2007), Les licences informatiques: un outil de modulation du régime d'appropriabilité dans les stratégies d'ouverture. Une interprétation de la licence SCSL de Sun Microsystems. In 12ème conférence de l'AIM, Lausanne (Suisse).

Muselli, L. (2008), Le rôle de licences dans les modèles économiques des éditeurs de logiciels open source, Revue française de gestion, n°181, pp. 199-214.

Nyman, L. Mikkonen, T., Lindman, J., Fougère, M. (2011), Forking: the Invisible Hand of Sustainability in Open Source Software", Proceedings of SOS 2011: Towards Sustainable Open Source.

Nyman, L., Mikkonen, T. (2011), To Fork or Not to Fork: Fork Motivations in SourceForge Projects, IFIP Advances in Information and Communication Technology, Vol. 365, pp. 259-268.

O'Mahony, S. (2007), The governance of open source initiatives: what does it mean to be community managed?. Journal of Management & Governance, 11(2), 139-150.

Orsini, L. (2015), Arduino vs. Arduino: What We Know About The Open-Source Hardware Fork, ReadWrite.com. Online : http://readwrite.com/2015/03/18/arduino-open-source-schism (lu : 07 février 2016).

Pénin J. (2012), Open source innovation: Towards a generalization of the open source model beyond software. Revue d'économie industrielle, (4), 65-88.

Poo-Caamaño, G., German, D. M. (2015), The Right to a Contribution: An Exploratory Survey on How Organizations Address It. In Open Source Systems: Adoption and Impact (pp. 157-167). Springer International Publishing.

Raasch, C., Herstatt, C., Balka, K. (2009), On the open design of tangible goods. R&D Management, 39(4), 382-393.

Raymond, E. S. (1998), Homesteading the noosphere. First Monday, 3(10).

Raymond, E.S. (2001), The Cathedral & the Bazaar (Musings on Linux and Open Source by an Accidental Revolutionary), O'Reilly Media.

Stol, K. J., Babar, M. A., Avgeriou, P., Fitzgerald, B. (2011), A comparative study of challenges in integrating Open Source Software and Inner Source Software. Information and Software Technology, 53(12), pp. 1319-1336.

Välimäki, M. (2003), Dual licensing in open source software industry, Systèmes d'Information et Management, vol. 8, n°1, 2003, pp. 63-75.

Viseur, R. (2011), Associer commerce et logiciel libre : étude du couple Netscape / Mozilla, 16ème conférence de l'AIM, Saint-Denis (France), 25 mai 2011.

Viseur, R. (2012), From Open Source Software to Open Source Hardware. In Open Source Systems: Long-Term Sustainability (pp. 286-291). Springer Berlin Heidelberg.

Viseur, R. (2013a). Identifying success factors for the mozilla project. In Open Source Software: Quality Verification (pp. 45-60). Springer Berlin Heidelberg.

Viseur, R., Pinchart, L. (2013b), Developing Free Software within a Major ICT Company, First International Workshop on Community Experience in Open Software Development (CommEx 2013, June 28th 2013, Koper-Capodistria (Slovenia).

Viseur, R. (2013c), Évolution des stratégies et modèles d'affaires des éditeurs open source face au cloud computing. Terminal. Technologie de l'information, culture & société, (113-114), pp. 173-193.

Viseur, R. (2016a), Open Source Hardware on the Cutting Edge: the Case of Open Source Cars, 2nd International Workshop on the Sharing Economy, 28-29 janvier 2016, Paris.

Viseur, R. (2016b), Etude des facteurs de succès du projet open source Apache Http, Actes du 21ème colloque AIM 2016, Lille, 18-20 mai 2016.

Viseur, R. (2016c), Open Concentration Index: Measure of Market Concentration in Open Source Industry, University of Mons, Working paper.

Von Hippel, E. (2001), User toolkits for innovation, Journal of Product Innovation Management, vol. 18 (4), July 2001, pp. 247-257.

Weber, S. (2004), The success of open source, Harvard University Press, April 30, 2004.

West, J. (2003), How open is open enough?: Melding proprietary and open source platform strategies. Research policy, 32(7), pp. 1259-1285.

Wheeler, D.A. (2007), Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers !, www.dwheeler.com.