HAL
archives-ouvertes.fr

Évaluation de l'innovation des logiciels open source

Nordine Benkeltoum

▶ **To cite this version:**

Nordine Benkeltoum. Évaluation de l'innovation des logiciels open source. Systèmes d'Information et Management, Eska, 2013, 18 (3), pp.1. 10.9876/sim.v18i3.491 . hal-00905917

HAL Id: hal-00905917
https://hal.archives-ouvertes.fr/hal-00905917

Submitted on 2 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ÉVALUATION DE L'INNOVATION DES LOGICIELS OPEN SOURCE

Nordine BENKELTOUM
Université Lille Nord de France, École Centrale de Lille
Cité Scientifique - CS20048
59651 Villeneuve d'Ascq Cedex
FRANCE
nordine.benkeltoum AT ec-lille.fr

RÉSUMÉ

L'appréciation de l'innovation dans le secteur du logiciel est une problématique contemporaine qui a pourtant reçu une attention académique limitée. De ce fait, la littérature présente un manque de critères pour mesurer l'innovation des logiciels. Sur un fond de controverse théorique et pratique sur le terrain de l'open source, cet article évalue l'innovation des logiciels libres. Sur la base de près de 500 études de cas et avec la collaboration de 125 experts de l'industrie, des services et de la recherche, il offre une typologie de l'innovation s'appuyant sur la notion de *valeur ajoutée fonctionnelle*. Il procure également un *cadre* pour la *modélisation* de l'*innovation* qui articule les principales approches de l'évaluation. En montrant les limites des métriques traditionnelles de l'économie et du management de l'innovation, cette recherche ouvre la voie à une nouvelle manière d'aborder la mesure de l'innovation en optant pour une approche spécifique à chaque secteur d'activité.

Mot clés : open source, logiciel libre, innovation, évaluation de l'innovation, Delphi

THE EVALUATION OF OPEN SOURCE SOFTWARE INNOVATIVENESS

ABSTRACT

Product innovation assessment in software sector is a timely topic. Nevertheless, research on that subject is particularly scant. As a result there is a lack of criteria to measure software innovativeness. In a context of theoretical and practical controversy in the open source field, this article assesses open source software innovativeness. Based on almost 500 cases studies and with the collaboration of 125 experts from industry, services and research fields, it suggests an innovation typology supported by the notion of *functional added value*. It provides also an *innovation modelling framework* that combines main evaluation methodologies. By showing the shortcomings of widely used innovation metrics, this research supports a new approach of innovativeness assessment specialized in each sector.

Keywords: open source software, free software, innovation assessment, Delphi

INTRODUCTION

La recherche en innovation s'est longtemps cantonnée à l'étude des capacités des entreprises (ou fabricants) commercialisant des produits. Les travaux de Von Hippel (1988) ont offert une nouvelle perspective à la recherche en innovation, plaçant l'utilisateur au cœur d'une nouvelle approche de l'innovation (Von Hippel 2005). Selon certains auteurs, l'utilisateur constitue même l'unité d'analyse d'un nouveau paradigme (Baldwin et Von Hippel 2011). Ainsi, les capacités des utilisateurs ont attiré une considérable attention académique se traduisant par un volume important de recherches sur le sujet (Bogers et al. 2010 ; Von Hippel 2010). La littérature a démontré que les innovations conçues par des utilisateurs étaient principalement des améliorations de produit (Allen 1983 ; Nuvolari 2004 ; Ulhoi 2004), des adaptations aux besoins (Franke et von Hippel 2003 ; Hicks et Pachamanova 2007 ; Lee et Cole 2003) ou encore offraient des réponses à des problèmes pratiques (Franke et al. 2010 ; Von Hippel 2005). Par ailleurs, internet a donné naissance à des communautés d'utilisateurs complètement autonomes comme les communautés open source (Livari 2010 ; Stuermer 2009). Depuis le début de ce siècle, un grand nombre d'universitaires se sont intéressés à ces communautés et ont mené un volume important de recherches (Von Krogh et Spaeth 2007).

L'appréciation de l'innovation dans le secteur du logiciel est un sujet relativement récent (Heredia Alvaro et Pikkarainen 2011). D'autre part, peu de travaux académiques se sont penchés sur la question de l'évaluation de l'innovation dans le champ de l'open source (Rossi 2009, p. 162). Néanmoins, quelques constats peuvent être dressés. D'abord, ce champ est objet de controverses théoriques et pratiques. Du point de vue théorique, les résultats des deux principales études quantitatives sont complètement contradictoires. D'un côté, Klincewicz suggère que la majorité des logiciels open source sont des clones de produits existants et que seul 1% de ces derniers constituent des innovations radicales (Klincewicz 2005). De l'autre, Rossi soutient que plus d'un tiers des logiciels étudiés sont nouveaux pour le marché et ajoute que *« les solutions open source sont plus innovantes que les solutions propriétaires »* (Rossi 2009, p. 163).

Du point de vue pratique, on retrouve également ce débat. Les citations ci-dessous sont issues d'entretiens menés auprès de praticiens de l'open source.

> *« Mon avis sur les logiciels libres est qu'ils copient en général l'existant, du moins pour ceux que j'utilise le plus, et que je m'en sers plutôt comme boîte à outils*[1]*. »*

> *« Je pense que la plupart des applications open source ne sont pas terriblement innovantes et, même s'il y a des exceptions, il y a tellement de différents champs/catégories de logiciels, et à l'intérieur desquels différentes applications sont innovantes de différentes manières – par conséquent il est difficile de vraiment les comparer et les classer*[2]*. »*

> *« Les logiciels open source sont au mieux un assemblage de fortune pour dépanner. J'ai besoin de trucs qui [..] marchent. Je n'ai pas besoin de passer des semaines à essayer de faire des trucs avec [un logiciel] open source que je peux faire en un jour avec un truc propriétaire. […]Rien ne me vient à l'esprit des 15 ans que j'utilise des logiciels open source*[3]*. »*

Au final, le caractère innovant des logiciels open source manque de clarté (The Economist 2006, p. 65) et il est difficile d'affirmer qu'il existe une relation causale entre

[1] Expert 7, Développeur, Google Inc., entretien en ligne avec l'auteur.
[2] Expert 104, Directeur général, TshwaneDJe Human Language Technology, entretien en ligne avec l'auteur.
[3] Expert 120, Consultant en logiciels libres et étudiant en cinéma, Freelance, entretien en ligne avec l'auteur.

l'efficience, la qualité et la valeur d'un logiciel d'un côté, et le fait qu'il soit open source de l'autre (Fuggetta 2003, p. 85). En outre, les recherches sur l'open source présentent d'importantes limites dont les principales sont les suivantes. Tout d'abord, beaucoup d'universitaires considèrent que l'open source favorise l'innovation (Fuggetta 2003, p. 81) en donnant l'impression d'une confusion entre processus et produit innovant. Ensuite, les méthodes d'évaluation utilisées dans les travaux antérieurs reposent sur des procédures d'une faible robustesse. Ainsi, certains travaux proposent une autoévaluation de l'innovation des logiciels open source (Klincewicz 2005) ou encore offrent une évaluation basée sur le jugement de trois experts seulement (Rossi 2009). Pour finir, la littérature fait état d'un manque de critères pertinents pour évaluation des logiciels en général (Lippoldt et Stryszowski 2009) et des logiciels open source en particulier (Rossi 2009).

Les travaux existants ont montré que les logiciels open source étaient développés par des organisations très variées (Benkeltoum 2011a). En effet, certains logiciels sont produits par des communautés d'utilisateurs (West et O'Mahony 2005), d'autres sont issus du monde de la recherche publique (Lakhani et von Hippel 2003), d'autres étaient initialement des produits fermés (Hamerly et al. 1999), d'autres sont maintenus par des consortiums (Spaeth et al. 2010), des réseaux de services (Feller et al. 2008) ou encore des communautés hybrides (Capra et al. 2011: 145). Dans cette recherche, la grande majorité des logiciels étudiés a été créée par des utilisateurs. Afin d'homogénéiser l'échantillon, les logiciels émanant d'autres organisations ont été exclus. Par conséquent, tous les logiciels analysés dans cet article sont conçus par des *"hobbyists*[4]*"* (Jeppesen et Frederiksen 2006).

Dans ce contexte de controverse théorique et pratique, la question de recherche de cet article est la suivante : *comment évaluer l'innovation des logiciels open source ?* La réponse de celui-ci est double : d'un côté, il propose une typologie de l'innovation basée sur la notion de valeur ajoutée fonctionnelle (VAF) ; d'un autre côté, il offre un cadre pour la modélisation de l'innovation (CMI) qui combine les approches qualitatives et quantitatives de l'évaluation appliquées au domaine des technologies de l'information (TI). Pour ce faire, il se décomposera en cinq parties. La prochaine section dressera un état des connaissances en matière d'évaluation de l'innovation produit et mettra en évidence les spécificités du secteur des logiciels. Elle sera directement suivie d'une description des principales avancées et limites de l'évaluation des logiciels open source et traitera en particulier de la question de la mesure de l'innovation dans ce champ. Une section décrira ensuite la démarche méthodologique de cette étude. Puis, une section présentera les principales contributions et implications de cette recherche. Pour finir, l'article clôturera par une présentation des limites et des perspectives pour la recherche.

[4] Il s'agit d'utilisateurs développant des produits sur leur temps libre.

ÉVALUATION DE L'INNOVATION PRODUIT ET SPÉCIFICITÉS DU LOGICIEL

Les principales approches de la mesure de l'innovation produit

Indicateurs en économie de l'innovation

La problématique de la mesure de l'innovation est aussi ancienne que l'économie classique (Garcia et Calantone 2002, p. 111). On attribue toutefois la paternité de l'une des premières typologies de l'innovation à Schumpeter, qu'il décline en cinq classes : les produits, les méthodes de production, les marchés, les sources d'approvisionnement et les nouvelles niches (OECD/Eurostat 2005, p. 35). Une innovation se définit comme l'introduction d'une nouveauté ou au moins une amélioration significative en matière de produit, service, procédé, commercialisation ou organisationnelle (OECD/Eurostat 2005, p. 54). Cette nouveauté s'analyse alors sous trois angles : l'entreprise, le marché ou le monde (OECD/Eurostat 2005, p. 67). L'innovation est un phénomène complexe jouant un rôle crucial pour les entreprises (OECD/Eurostat 2005, p. 14) et pour l'économie entière.

Depuis le début des années 90, des initiatives ont été menées afin de mesurer et comparer l'innovation sur le plan international. Dans ce domaine, il existe une référence développée conjointement par l'OCDE[5] et la Communauté Européenne (CE) : il s'agit du « Manuel d'Oslo » dont l'évolution en différentes versions témoigne de la nécessité d'adapter les indicateurs aux mutations de l'économie. Toutefois, les métriques proposées dans ce manuel présentent d'importantes limites. D'abord, le champ d'application est celui des entreprises, il exclut de ce fait toute activité non marchande (OECD/Eurostat 2005, p. 20-21) dont les communautés d'utilisateurs font partie intégrante. Cela est d'autant plus marqué dans le secteur des logiciels où la production par les utilisateurs représente une partie substantielle de la production (Lippoldt et Stryszowski 2009, p. 8). Ensuite, en matière d'innovation radicale, on note des différences significatives de performance des entreprises en fonction des secteurs d'activité (OECD/Eurostat 2005, p. 45). Si une partie de cette disparité peut être attribuée à une différence de dynamisme entre secteurs ; une autre partie est forcément imputable à la métrique servant à analyser le caractère radical. Dans certains secteurs, cela a conduit les chercheurs à développer des indicateurs *ad hoc*. On peut notamment citer le secteur des activités traditionnelles, où des indicateurs liés à l'esthétisme ont été développés (Alcaide-Marzal et Tortajada-Esparza 2007), ou encore le secteur du logiciel dont les particularités dénotent la nécessité de métriques spécifiques (Lippoldt et Stryszowski 2009).

Indicateurs en management de l'innovation

Parallèlement à ces indicateurs, la recherche en management de l'innovation propose un ensemble de critères permettant l'appréciation de l'innovation produit[6]. Il existe deux paradigmes dominants en matière d'évaluation de l'innovation : le couple *incrémental/radical* et celui du maintien/rupture. Ainsi, les innovations incrémentales et radicales se distinguent par le type de connaissances qu'elles contiennent. Les innovations radicales contiennent un haut degré de nouvelles connaissances tandis que les innovations incrémentales reposent sur un bas niveau de nouvelles connaissances (Dewar et Dutton 1986, p. 1422). La notion de produit radicalement nouveau fait référence à des produits basés sur une nouvelle technologie (Chandy et Tellis 1998, p. 476).

[5] Organisation de Coopération et de Développement Économiques

[6] Pour une revue relativement complète des métriques génériques permettant d'évaluer l'innovation se référer à (Garcia et Calantone 2002).

Les innovations de maintien et de rupture s'apprécient en fonction de critères marketing (Christensen 1997 ; Christensen et Raynor 2003). D'un côté, les innovations de maintien consistent à améliorer des produits sur la base des critères du marché. D'un autre côté, les innovations de rupture introduisent une proposition de valeur non attendue par le marché (Christensen 1997 ; Christensen et Overdorf 2000). Le concept d'innovation de rupture a fait l'objet d'un profond débat en matière de développement de nouveaux produits. La littérature a souligné qu'il n'y avait pas de critères précis permettant de déterminer si une technologie était de rupture ou non (Danneels 2004 ; 2006).

Pour conclure, l'abondance des terminologies et le débat autour de leur définition rend très difficile leur utilisation. De plus, comme cela est souligné par des chercheurs *« la littérature en développement de nouveaux produits a principalement utilisé des classifications dichotomiques pour identifier le type d'innovation. Nous pensons que ces dichotomies sont trop simplistes »* (Garcia et Calantone 2002, p. 120).

L'évaluation de l'innovation dans le secteur du logiciel

L'innovation dans le secteur du logiciel se définit comme un processus conduisant au développement d'un nouveau design, d'une fonctionnalité ou d'une application nouvelle pour un produit existant ; un nouveau produit, service ou processus ; l'amélioration d'un produit, service ou processus ; l'entrée sur un marché ou la création d'un nouveau marché (Lippoldt et Stryszowski 2009, p. 10). Les caractéristiques de l'innovation dans ce domaine diffèrent des autres secteurs d'activité et ce à plusieurs titres. Le logiciel est singulier car contrairement aux autres produits, il continue d'évoluer après sa diffusion à travers le mécanisme des mises à jour (Codenie et al. 2011, p. 7). En d'autres termes, le logiciel est en constante conception. Ensuite, il est également intangible (Codenie et al. 2011, p. 7; Lippoldt et Stryszowski 2009, p. 8), cela signifie qu'il n'est pas palpable, mais aussi et surtout, qu'il peut être répliqué et diffusé à faible coût. Pour finir, il est aisé de concevoir un logiciel car les coûts d'entrée sur le marché sont faibles. Il suffit d'un ordinateur et de connaissances en programmation (Codenie et al. 2011, p. 8).

Les métriques servant à l'identification de l'innovation des produits tangibles comme les brevets (Lippoldt et Stryszowski 2009, p. 86) ou les marques (Rossi 2009, p. 155) ne sont pas adaptés pour les logiciels. De plus, il n'existe pas de métriques claires et universelles permettant de mesurer l'innovation logicielle (Lippoldt et Stryszowski 2009, p. 8). Dès lors, l'industrie des logiciels requiert une adaptation des métriques existantes à ses singularités (Klincewicz 2005, p. 7) voire la création de métriques spécifiques (Lippoldt et Stryszowski 2009 ; Rossi 2009, p. 155). Plusieurs auteurs ont tenté de proposer des métriques adaptées aux spécificités du logiciel en se focalisant sur les éléments permettant d'identifier les innovations radicales. Soens suggère quatre classes d'innovation : (1) la classe 1 : les innovations incrémentales, (2) la classe 2 : les innovations radicales tirées par le marché, (3) la classe 3 : les innovations radicales poussées par la technologie, (4) la classe 4 : les innovations basées sur la recherche (Soens 2011, p. 41-45).

Quant à Klincewicz, il offre une grille d'analyse basée sur les travaux de Dahlin et Behrens (2005). Ces derniers soutiennent l'idée qu'une technologie radicalement innovante est réussie lorsqu'elle est : *« nouvelle, unique et a un impact sur les technologies futures »* (Dahlin et Behrens 2005: 717). Selon Klincewicz, le critère de nouveauté et le caractère unique sont satisfaits si aucune fonctionnalité comparable n'existait avant la date du lancement du produit. Klincewicz y ajoute la dimension de plateforme (système d'exploitation) et propose la grille reproduite ci-dessous.

Tableau 1 : Grille d'évaluation de l'innovation logicielle (Klincewicz, 2005)

	Nouvelle technologie	**Nouveau pour une plateforme**	**Technologie existante**
Nouveau marché	Radical invention (rupture)		Innovation marketing
Marché existant	Modification de technologie	Modification de plateforme	Pas d'innovation

Rossi propose des critères d'évaluation spécifiques classés en trois dimensions : la firme, le marché et le monde. (1) La première dimension mesure l'innovation pour la firme. Elle repose sur un indicateur unique indiquant si le logiciel est nouveau pour la firme (indicateur 1). (2) La seconde dimension comprend deux indicateurs et mesure si le logiciel est innovant pour le marché. Le premier critère analyse la capacité du logiciel à satisfaire les besoins et demandes des utilisateurs (indicateur 2). Le second critère s'intéresse au degré de nouveauté du logiciel d'un point vue technologique (Indicateur 3). (3) La troisième dimension est la plus générale, elle contient deux critères. Le premier critère interroge sur le fait que le logiciel contient un module inédit (Indicateur 4). Le second critère porte sur la nouveauté de la plateforme logicielle (Indicateur 5) (Rossi 2009: 160).

AVANCÉES ET LIMITES DE L'ÉVALUATION DES LOGICIELS OPEN SOURCE

Généralités sur l'évaluation des logiciels open source

Caractéristiques et controverses

Les technologies open source sont devenues incontournables en systèmes d'informations (SI) aussi bien en théorie (Fitz-Gerald 2010 ; Gary et al. 2011) qu'en pratique (CIGREF 2011 ; Le Texier et Versailles 2009 ; Lindman et al. 2011 ; Lisein et al. 2009). Si les avantages de ces technologies sont nombreux et bien documentés (Thakur 2012), d'autres caractéristiques sont beaucoup plus controversées (Fuggetta 2003). D'abord, en ce qui concerne les avantages, les logiciels libres ont une structure du code source de qualité (Capra et al. 2011 ; Von Krogh et Spaeth 2007) et les organisations assurant le développement garantissent une correction rapide des bugs (Bitzer et Schröder 2005 ; Paulson et al. 2004). De plus, l'open source garantit la transparence (Spinellis et al. 2009), l'interopérabilité ainsi que l'indépendance technologique des entreprises et des États (Benkeltoum 2011a ; 2011b). Ce constat est d'autant plus vrai pour les pays en voie de développement (Chen et al. 2010) et les puissances montantes (Chine, Brésil).

Les points de controverse sont également multiples (Fuggetta 2003) et concernent notamment les pratiques de réutilisation, la modularité et l'innovation[7]. Pour certains universitaires, les logiciels open source présentent un taux de réutilisation de code en interne et externe (entre projets) tout à fait satisfaisant (Haefliger et al. 2008), et pour d'autres ces pratiques peuvent largement être améliorées (Capiluppi et al. 2011). De même, une étude démontre que les logiciels libres n'ont pas une structure plus modulaire que les logiciels fermés

[7] Ce dernier point sera traité spécifiquement à la prochaine section.

(Paulson et al. 2004) alors qu'une autre défend exactement la thèse inverse (MacCormack et al. 2006).

Approches d'évaluation

L'évaluation des TI est toujours un sujet d'actualité (Benlian 2011 ; Lippoldt et Stryszowski 2009), les logiciels open source ne sont pas une exception à la règle (Lundell et al. 2010 ; Miralles et al. 2006). En effet, il y a un corpus théorique montant sur l'évaluation des logiciels open source où deux approches sont identifiables. D'une part, les logiciels open source sont appréciés via des métriques bien connues en évaluation des TI (Miralles et al. 2006). Par exemple, Benlian compare des suites bureautiques ouvertes et fermées sur la base des mêmes métriques : les fonctionnalités, le prix, la facilité d'utilisation et le support technique (Benlian 2011). Du point de vue économique, ces logiciels peuvent être évalués au moyen du modèle du *Total Cost of Ownership* (TCO) (Fitzgerald 2006) qui prend en considération l'ensemble des coûts liés à l'adoption d'une technologie comme : la formation, les frais de migration ou encore la maintenance (Taibi et al. 2007). Alors qu'une comparaison centrée sur le coût d'acquisition (licences) favorise les logiciels open source car la plupart sont distribués gratuitement (Benkeltoum 2011b).

D'autre part, un autre courant de recherches a prouvé que l'appréciation des logiciels open source nécessitait des métriques adéquates[8] et enrichies. Par exemple, du fait que les codes sources de la plupart de ces logiciels sont facilement accessibles, l'évaluation peut reposer sur une analyse de la qualité (Lundell et al. 2010 ; Midha et Palvia 2012) et de la structure du code source (Capra et al. 2011 ; MacCormack et al. 2006). L'évaluation des logiciels fermés ne peut être réalisée sans décompilation, ce que les licences de ces derniers interdisent. De même, bon nombre de logiciels libres ne sont pas distribués par des entreprises. Il est donc impossible de mesurer la réactivité ou d'apprécier les services de l'éditeur. Ces éléments doivent être mesurés via des moyens *ad hoc* comme l'activité des listes de diffusion, la qualité de la documentation du code source ou encore via l'existence de services communautaires (Del Bianco et al. 2009 ; Lee et al. 2009 ; Spinellis et al. 2009 ; Taibi et al. 2007). Le présent article s'inscrit dans ce courant de recherche et bâtit des métriques spécifiques aux logiciels open source.

L'évaluation de l'innovation des logiciels open source

Controverse théorique

Dans la littérature, seules deux études abordent la question de l'évaluation de l'innovation dans l'open source de manière approfondie. Ces dernières offrent des résultats complètement contradictoires. Klincewicz évalue les 500 logiciels les plus actifs de SourceForge via la méthode de l'autoévaluation. Plus particulièrement, Klincewicz *« se focalise sur l'importance des modifications pour le produit entier, perçu par les individus développant le projet »* (Klincewicz 2005: 9). Il mobilise pour cela une grille d'évaluation s'appuyant sur la notion d'innovation radicale adaptée au logiciel (cf. précédente section). Les résultats montrent que 87,2% des logiciels libres étudiés ne sont pas innovants, 10,6% introduisent des modifications de plate-forme, 1% des innovations radicales, 0,8% des modifications de technologie et 0,6% des innovations marketing. Cette recherche prouve qu'il

[8] Deux projets Européens se sont succédé (Qualoss et QualiPSo) sur la problématique de la mesure de la qualité des logiciels open source ce qui témoigne de la complexité du sujet. Pour une description de QualiPSo se référer à Benkeltoum (2011a, p. 123-125).

est aisé d'identifier les logiciels non innovants et les modifications de plateforme mais qu'il est complexe d'identifier les autres catégories de logiciels notamment les innovations radicales.

Rossi compare les solutions offertes par des PME[9] italiennes basées sur des logiciels libres et des solutions *« propriétaires »* (Rossi 2009). L'évaluation porte sur un total de 134 logiciels : 107 sont fermés et 27 sont open source. Elle montre que les logiciels libres sont plus innovants que les logiciels fermés sur l'ensemble des critères (cf. section précédente) : que ce soit du point de vue de la firme, du marché ou du monde.

Les limites de la littérature

Les travaux en évaluation de l'innovation des logiciels open source présentent les limites suivantes : l'apparente confusion entre processus et produit innovant, la faiblesse de la procédure d'évaluation et le manque de critères pertinents permettant d'évaluer l'innovation des logiciels.

- *Limite 1: l'apparente confusion entre processus et produit innovant*

Si certains universitaires considèrent que l'open source ne représente pas un réel changement en matière de développement logiciel (Fitzgerald 2006, p. 594), d'autres suggèrent qu'il en constitue une nouvelle approche (Fuggetta 2003, p. 81). Par ailleurs, il semble que bon nombre de chercheurs considèrent que les produits open source sont innovants car leur processus de conception est innovant, et emploient de manière synonyme les expressions *open source development* et *open source innovation*. Le risque est d'utiliser le mot *innovation* par effet de mode (Klincewicz 2005: 3). Ainsi, pour certains l'open source symbolise en lui-même une *innovation de rupture* (Rossi 2009: 154), un cas extrême d'*innovation ouverte* (Dahlander et Wallin 2006) ou encore une *innovation radicale* (Bonaccorsi et al. 2006: 1086).

Ce mode de développement, aux participants géographiquement dispersés, a tantôt été qualifié de *« E-Innovation »* (Kogut et Metiu 2000 ; 2001), de *« réseaux d'innovation par et pour des utilisateurs »* (Von Hippel 2002) ou encore de *« Private-Collective Model »* où des participants utilisent leurs propres ressources pour constituer un bien collectif (Von Hippel et Von Krogh 2003: 213). Plus récemment, ce modèle a été étendu aux communautés hybrides mettant en relation une firme (Nokia), des partenaires et des développeurs bénévoles (Stuermer et al. 2009). Ce modèle a toutefois une importante limite puisqu'il ne propose pas de distinction entre ce qui est innovant et ce que ne l'est pas. Par exemple, Nokia propose à des entreprises concurrentes d'utiliser sa plateforme. Les auteurs qualifient cette action de diffusion de l'innovation (Stuermer et al. 2009: 180) alors que le phénomène mis en évidence porte sur le partage d'un socle technologique générique sur lequel Nokia ne crée pas de valeur[10]. Nokia préfère garder le contrôle sur les couches applicatives et ne partage pas à ce niveau. Ce cas est typiquement similaire à la stratégie d'intégrateurs comme Thales dans le domaine du *middleware* qui partage des technologies sur des couches basses et conserve des parties stratégiques secrètes (Benkeltoum 2011a). Ceci rejoint la déclaration d'un employé de Nokia pour qui *« notre truc n'est pas si révolutionnaire »* (Stuermer et al. 2009, p. 182).

De nombreux auteurs (Hicks et Pachamanova 2007 ; Lee et Cole 2003 ; Osterloh et Rota 2007 ; Von Hippel et Von Krogh 2003) considèrent toutes les modifications de logiciel comme une innovation. Ainsi, le retour des modifications réalisées par des utilisateurs sur le code d'un logiciel libre est considéré comme un phénomène de propagation d'innovation utilisateurs (Hicks et Pachamanova 2007: 318). Une étude du développement du noyau Linux a montré que les développeurs avaient deux fonctions : une fonction d'assurance qualité et une fonction

[9] Petites et Moyennes Entreprises

[10] Les auteurs parlent eux-mêmes de *« generic frameworks »* (Stuermer, Spaeth et von Krogh, 2009 : 185).

d'innovation. Pour l'innovation, les développeurs suggèrent de nouvelles fonctions ou écrivent des correctifs (Lee et Cole 2003: 637). De même, certains chercheurs confondent les activités de production et d'innovation (Bogers et al. 2010 ; Osterloh et Rota 2007 ; Von Krogh et al. 2003). Enfin d'autres défendent l'idée que le paradigme de l'open source favorise l'innovation (Deek et McHugh 2007 ; Ebert 2007 ; 2008 ; Janamanchi et al. 2009 ; Von Krogh et Spaeth 2007).

- *Limite 2 : une procédure d'évaluation d'une faible robustesse*

Une évaluation qualitative de l'innovation des logiciels libres a souligné que ces logiciels relevaient à la fois de l'innovation et de la copie (Deek et McHugh 2007, p. 7). D'un côté, des logiciels libres copient des logiciels fermés. De l'autre, ils constituent la source technologique qui a permis la création d'internet (Fitz-Gerald 2010). Néanmoins, Deek et McHugh ne détaillent pas les critères utilisés pour distinguer les logiciels open source innovants ou non.

L'étude de Klincewicz porte sur 500 logiciels open source en s'appuyant sur la technique de l'autoévaluation (Klincewicz 2005). Rossi évalue 134 logiciels (107 fermés et 27 open source) basé sur l'évaluation de trois experts (Rossi 2009, p. 159). De plus, la question de recherche de l'étude est la suivante : *« est-ce que les programmes basés sur des solutions FLOSS sont plus innovants que les [logiciels] propriétaires »* (Rossi 2009, p. 153) ? Or, il convient de souligner que Rossi évalue plutôt des solutions logicielles basées sur des logiciels libres que les logiciels libres eux-mêmes. Le produit résultant est plutôt une combinaison de logiciels libres et non libres, c'est-à-dire une solution hybride (Bonaccorsi et al. 2006).

- *Limite 3: le manque de critères pour apprécier l'innovation des logiciels*

La littérature a souligné le manque de critères pertinents pour apprécier l'innovation dans tous les secteurs de l'économie (Le Masson et al. 2010 ; Rossi 2009) et en particulier pour le logiciel (Lippoldt et Stryszowski 2009, p. 8). De même, Deek et McHugh ne révèlent pas les critères utilisés pour distinguer l'innovant de l'imitatif (Deek et McHugh 2007). Dans son étude, Klincewicz utilise des critères non testés pour jauger l'innovation de 500 logiciels (Klincewicz 2005). Rossi propose une évaluation de 27 logiciels libres basée sur des critères non validés empiriquement (Rossi 2009). Dans cette dernière recherche, plus d'un tiers (35%) des produits sont considérés comme *« nouveaux pour le marché »* et plus généralement, un tiers de produit reçoit des scores élevés pour l'innovation (Rossi 2009, p. 162) ce qui est extrêmement rare pour une industrie. Par exemple, la littérature en développement de nouveaux produits estime que seulement 10% des innovations sont radicales (Garcia et Calantone 2002, p. 120). De plus, les critères de la radicalité diffèrent pour les utilisateurs et les fabricants (Afuah 2000 ; Bogers et al. 2010).

MÉTHODOLOGIE DE LA RECHERCHE

Vue générale de la recherche

Positionnement et objectifs

Cet article s'appuiera sur la définition de Lippoldt et Stryszowski qui soulignent qu'une innovation dans le champ du logiciel peut-être purement fonctionnelle (Lippoldt et Stryszowski 2009, p. 10). En revanche, contrairement à Klincewicz, cette recherche ne considérera pas que la proposition d'une fonctionnalité nouvelle et unique constitue une innovation radicale (Klincewicz 2005) étant donné que la notion d'innovation radicale se focalise sur l'utilisation d'une technologie nouvelle (Chandy et Tellis 1998). Pour ces raisons, la notion d'innovation fonctionnelle sera utilisée car elle est davantage en conformité avec la nature de l'innovation dont il s'agit. L'objectif de cette recherche est d'apporter une typologie de l'innovation fonctionnelle associée à des métriques spécifiques à l'open source. Pour ce faire, la notion de *valeur ajoutée* sera utilisée. Elle désigne une plus-value apportée par un logiciel vis-à-vis des produits existants.

La recherche en SI a montré que la combinaison de méthodologies qualitatives et quantitatives est un outil puissant (Agerfalk et Fitzgerald 2008 ; Mingers 2000 ; 2003 ; Pinsonneault et Kraemer 1993) et tout particulièrement dans l'open source (Feller et al. 2008 ; Haefliger et al. 2008 ; Lee et Davis 2003). Cette recherche s'appuie sur le courant de la multiméthodologie. Une approche qui combine différentes méthodologies ou techniques (quantitatives ou qualitatives) afin de tirer profit de chaque méthode (Mingers 2000). Elle se définit comme « *un ensemble de directives ou d'activités pour aider les personnes à mener des recherches ou des interventions* » (Mingers et Brocklesby 1997: 490). Quatre arguments sont en faveur de la multiméthodologie : (1) la complexité des évènements de la réalité, (2) le fait que la recherche soit la plupart du temps divisée en étapes, (3) que l'utilisation de la multiméthodologie est courante en pratique et enfin (4) parce qu'elle constitue une forme contemporaine en pratique (Mingers et Brocklesby 1997: 492).

Mingers et Brocklesby (1997: 491) proposent cinq niveaux allant de ce que les auteurs nomment l'isolationnisme à la multiméthodologie (voir Tableau 2).

Tableau 2 : de l'isolationnisme à la multiméthodologie

Principe	Description
Isolationnisme	Utiliser une seule méthode ou technique d'un paradigme
Amélioration	Améliorer une méthodologie
Sélection	Choisir les méthodologies adéquates en fonction du contexte
Combinaison	Combiner des méthodologies pour un objectif
Partition et combinaison	Partitionner et combiner les méthodologies

Design général de la recherche

La méthodologie de cette recherche se décompose en quatre étapes : (1) la définition qualitative de types d'innovation fonctionnelle sur la base d'études de cas de grande échelle (294 cas), d'interviews avec des experts et d'une revue de la littérature ; (2) une évaluation de 25 logiciels open source par 125 experts ; (3) une comparaison des classifications qualitative et quantitative ; (4) la modélisation de types d'innovation fonctionnelle. Chaque étape est basée sur différentes méthodes qui contribuent à l'objectif général de la recherche (Tableau 3).

Tableau 3 : contribution de chaque étape aux objectifs de la recherche

Étape	Titre	Objectifs
1	Définition qualitative de types d'innovation fonctionnelle	- Exploration du champ - Identification de construits à priori - Catégorisation qualitative de logiciels libres
2	Évaluation de logiciels open source par des experts	- Validation des variables - Validation des hypothèses
3	Comparaison des classifications qualitative et quantitative	- Attribution d'un type sur la base du jugement d'expert - Validation de la classification
4	Modélisation des types d'innovation fonctionnelle	- Modélisation des types d'innovation fonctionnelle dans l'open source

Dans cette recherche, différentes méthodes ont été combinées à chacune des étapes. Elles seront présentées au fur et à mesure du déroulé de cette étude. Le tableau ci-dessous offre une vue générale des principes de ces méthodes, une brève description du rôle joué par celles-ci aux différentes étapes de la recherche.

Tableau 4 : méthodes combinées pour l'étude de l'innovation fonctionnelle

Méthodes	Principes	Parties utilisées et description	Étapes	Littérature
Étude de cas multiple	Comprendre les dynamiques entre les cas	- « échantillonnage théorique » : sélection de 294 logiciels open source - utilisation des principes d'inversement, extension et réplication	1, 2	(Eisenhardt 1989 ; Eisenhardt et Graebner 2007)
Technique Delphi	Évaluation par des experts	- évaluation par des experts : 25 logiciels open source - agrégation statistique : 125 évaluations sur 25 logiciels open source	2	(Danneels 2004 ; Rossi 2009 ; Rowe et Wright 1999)
Théorie enracinée	Découvrir une théorie à partir de données	- codification qualitative des types d'innovation	1	(Glaser et Strauss 1967 ; Sigfridsson et Sheehan 2011)
Statistiques	Utilisation d'outils statistiques	- analyse de la variance : validation des variables - analyse en composantes principales : comparaison entre la classification qualitative et celle des experts - Analyse discriminante : test d'intégrité de la classification qualitative et modélisation de l'innovation	2, 3, 4	(Burns et Burns 2008 ; Capra et al. 2011 ; Lee et al. 2009)
Échantillonnage boule de neige	Identification de répondants grâce à d'anciens répondants	- diffusion du questionnaire parmi les experts	2	(Shah 2006)
Netnographie	*« Ethnographie adaptée pour l'étude de communautés en ligne »* (Kozinets 2002)	- discussion en ligne : interviews avec des développeurs et des *informateurs clés* - étude de fils de discussion	1, 2, 3	(Kozinets 2002 ; 2006 ; Sigfridsson et Sheehan 2011)
Auto-évaluation	Évaluation basée sur des critères sélectionnés	- auto-évaluation de 294 logiciels open source - auto-évaluation de 152 logiciels open source	1, 2	(Chen et al. 2010 ; Del Bianco et al. 2009 ; Fearon et Philip 1998 ; Klincewicz 2005 ; Rowe et Wright 1996)

Détail des étapes de la recherche

Étape 1 : définition qualitative de types d'innovation fonctionnelle

- *Objectifs*

Le but de cette première étape est de proposer une typologie de l'innovation fonctionnelle des logiciels open source sur la base d'une analyse qualitative.

- *Données*

Des critères d'évaluation ont été définis sur la base d'un échantillon de 294 logiciels open source. Ces logiciels ont été sélectionnés en grande partie sur la base d'une liste de 231 logiciels libres à tester sous le système d'exploitation libre ReactOS[11]. Cette liste fut complétée par des logiciels libres utilisés par l'auteur ou mentionnés par des experts. Pour chaque logiciel libre, des informations ont été rassemblées à propos de : ses principales fonctionnalités, ses atouts comparés aux logiciels fermés existants et son origine. Ces données ont été récoltées grâce à l'échange de plus de 524 courriels[12] avec les fondateurs ou des *acteurs clés* du projet (Agerfalk et Fitzgerald 2008, p. 392) mais aussi d'informations disponibles sur le site officiel de chaque logiciel (Stewart et Gosain 2006, p. 299) en triangulant à chaque fois l'information avec d'autres sources (Wikipédia, Freecode, forums, etc.).

Des interviews approfondies avec des professionnels de l'open source ont été menées. Trois experts ont joué un rôle particulièrement fondamental dans la définition des types d'innovation dans le domaine du logiciel libre (**Tableau 5**). 1) Le premier expert est un chercheur industriel d'Alcatel-Lucent qui *« travaille en R&D dans le monde IP (Internet) »* et *« dépose des brevets régulièrement »*. Il est doté d'une expérience de plus de 10 ans en développement et est l'auteur de plusieurs logiciels libres. Environ 67 courriers électroniques ont été échangés avec cet expert. 2) Le deuxième expert est chef de projets au sein de Thales D3S. Il est doté d'une expérience de plus de dix ans en développement. Il occupe des positions de responsabilité dans différents projets européens dédiés aux logiciels libres. Il a une longue expérience en développement notamment pour un distributeur de Linux. Environ 20 courriers électroniques ont été échangés, un entretien d'environ deux heures a été mené et des discussions informelles conduites lors d'une participation à un colloque dédié aux technologies open source (sur quatre jours). 3) Le troisième expert est ingénieur en développement pour une PME éditrice de logiciels libres. Il est également membre du conseil d'administration d'une association de promotion et de défense des logiciels libres. Il a près de dix ans d'expérience en développement de logiciels. Environ 190 courriers électroniques ont été échangés, un entretien d'environ une heure et demie et des discussions informelles dans le cadre d'un salon dédié aux logiciels libres ont été réalisés. Les membres du conseil d'administration (composé de 14 experts de l'open source) de cette association[13] ont largement contribué à l'amélioration de cette recherche en proposant une analyse critique de celle-ci.

Tableau 5 : synthèse des profils des trois experts

[11] ReactOS est un système d'exploitation visant à être pleinement compatible avec les applications Windows 32 bits.

[12] Calcul basé sur un comptage des archives électroniques.

[13] L'association comprend approximativement une cinquantaine de membres institutionnels et 350 membres individuels.

ID	Fonction	Entreprise	Communauté	Expérience	Sources d'informations
1	Chercheur industriel	Alcatel-Lucent	Auteur de plusieurs logiciels libres	- 10 ans en développement - Dépôt régulier de brevets	- 67 courriels
2	Chef de projets	Thales D3S	Membre OW2 Mandriva	- 10 ans en développement - Chef de projets européens	- 20 Courriels - Entretien 2 heures - Discussions informelles (séminaire de 4 jours)
3	Ingénieur en développement	Éditeur logiciel libres	Membre du CA de l'AFUL	- 10 ans en développement	- 190 courriels - Entretien 2 heures - Discussions informelles

- *Analyses*

Les données récoltées pendant la phase qualitative ont été codées suivant le processus itératif suivant :

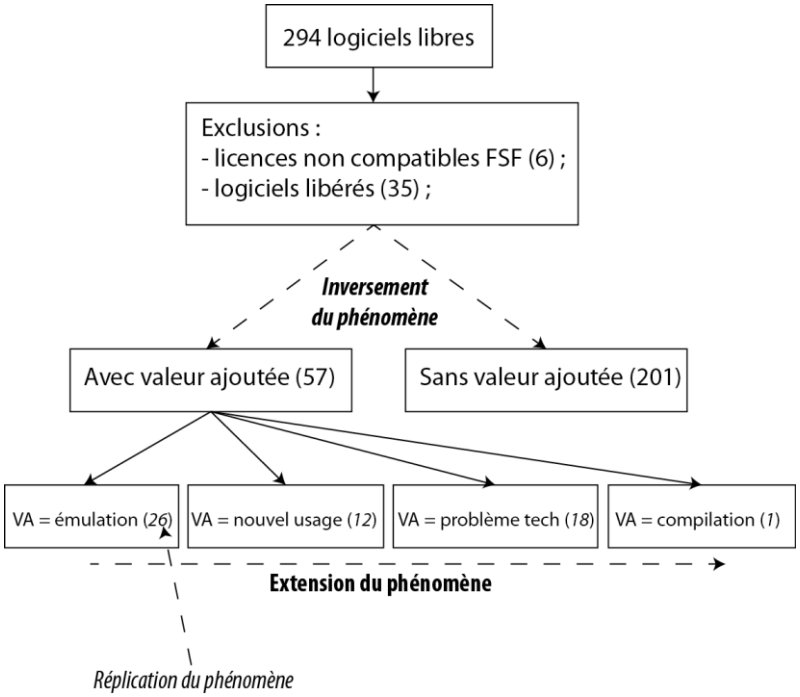
1) étude de la licence afin de s'assurer qu'elle soit bien compatible avec les critères de la Free Software Foundation[14] : 6 logiciels ont été exclus ;
2) vérification que le logiciel a bien été conçu par des utilisateurs : 35 logiciels libérés par des entreprises ont été exclus ;
3) identification du domaine et du sous domaine d'appartenance du logiciel sur la base de sa définition ;
4) recherche d'antériorité par contact avec les fondateurs, développeurs ou données publiques ;
5) identification de la présence ou non de valeur(s) ajoutée(s) introduite(s) par le logiciel du point de vue fonctionnel (sources identiques à 4) ;
6) séparation des logiciels en deux types : avec valeur ajoutée (55) et sans valeur ajoutée (200) fonctionnelle ;
7) classification qualitative des logiciels en fonction du type de valeur ajoutée.

Pour la septième phase, les logiciels ont été classifiés en utilisant l'approche de la théorie enracinée qui vise à faire émerger des théories à partir de données (Glaser et Strauss 1967 ; Sigfridsson et Sheehan 2011). Ainsi, les données ont été analysées manuellement en maximisant les phénomènes d'inversement, d'extension et de réplication (Eisenhardt et Graebner 2007, p. 27). Il s'agissait plus précisément de : distinguer les logiciels apportant une valeur ajoutée (VA) de ceux n'en apportant aucune (inversement de phénomène) ; créer de nouveaux types de VA lorsque la VA apportée par un produit n'entrait pas dans les catégories précédemment identifiées (extension du phénomène) ; identifier la répétition du type de VA (réplication du phénomène). Cette classification s'est aussi appuyée sur l'avis de trois experts (**Tableau 5**) et une revue des innovations fonctionnelles mentionnées dans la littérature. Au final, un processus d'autoévaluation, une méthode où les chercheurs définissent eux-mêmes les critères utilisés pour l'évaluation (Chen et al. 2010 ; Del Bianco et al. 2009 ; Fearon et Philip

[14] Selon la FSF, un logiciel est considéré comme libre si sa licence confère quatre libertés à l'utilisateur : l'exécution, l'étude, la modification et la diffusion.

1998 ; Klincewicz 2005 ; Rowe et Wright 1996), a été suivi. Ce processus peut être schématisé comme ci-dessous.

Figure 1 : classification qualitative (étape 1)



- *Résultats*

Ce processus a permis d'identifier des construits à priori (Eisenhardt 1989 ; Miles et Huberman 1994). Ces construits ont été utilisés ensuite pour créer un questionnaire d'évaluation (cf. étape 2), cinq types d'innovation fonctionnelles ont ainsi émergé des données : (1) L'alternative libre, un logiciel alternatif à un logiciel fermé existant. Cette catégorie regroupe tous les logiciels qui n'apportaient aucune valeur ajoutée du point de vue fonctionnel à part le fait d'être libre. (2) L'émulateur, une version *virtualisée* d'un matériel ou d'un ensemble d'applications logicielles. Il est innovant puisqu'il permet aux utilisateurs de remplacer un matériel qui n'est plus fabriqué car considéré comme obsolète. (3) Le package, un logiciel regroupant plusieurs logiciels libres en un seul bloc, la plupart du temps pour faciliter l'usage, l'installation et/ou la configuration de logiciels complexes. (4) La pièce d'adaptation, un logiciel qui résout un problème technique n'ayant jamais été résolu précédemment. (5) L'orienté nouvel usage, un logiciel introduisant un concept d'usage inédit. Cette typologie s'appuie à la fois sur les données et/ou la littérature sur l'open source. De plus, des exemples permettent d'illustrer concrètement cette typologie (Tableau 6).

Tableau 6 : catégories issues de l'analyse qualitative

Types	Exemple détaillé	Autres exemples	Littérature
Alternative	Le projet GNU est une alternative libre à Unix (Stallman 1983)	Linux Kernel, Gnu Privacy Guard	(Hars et Ou 2002 ; Osterloh et Rota 2007 ; Spiller et Wichmann 2002)
Émulateur	PCSX2 simule le fonctionnement de la Playstation 2	Cassini, PearPC	(Von Krogh et Spaeth 2007)
Package	Le package AMP regroupe Apache, MySQL, Perl et PHP	Debian, Ubuntu	(Deek et McHugh 2007)
Pièce d'adaptation	Mono permet d'exécuter des applications .Net sur des systèmes Linux/Unix	Wine, Samba	(Hicks et Pachamanova 2007 ; Stallman 2002)
Orienté nouvel usage	Mute est un logiciel de P2P (Peer to Peer) permettant à ses utilisateurs de préserver leur anonymat	Nmap, BitTorrent	

Étape 2 : Évaluation de logiciels open source par des experts

- *Objectifs*

Le but de cette étape est de valider la pertinence des variables dérivées de l'étude qualitative. C'est dans cet objectif que des hypothèses ont été dérivées de l'étape 1 :

- *H1 : certains logiciels libres représentent des alternatives à des logiciels fermés existants ;*
- *H2 : certains logiciels libres émulent le fonctionnement de matériel ou d'un ensemble d'applications logicielles ;*
- *H3 : certains logiciels libres combinent un ensemble de logiciels libres ;*
- *H4 : certains logiciels libres résolvent un ou plusieurs problème(s) techniques non résolu(s) préalablement ;*
- *H5 : certains logiciels libres créent un nouveau concept d'usage.*

L'ensemble de ces hypothèses seront testées grâce à une évaluation basée sur la technique Delphi (Rowe et Wright 1999). Cette technique est utilisée lorsqu'un manque de données appropriées est observé et lorsque le jugement humain est nécessaire (Nambisan et al. 1999, p. 374). Il y a quatre conditions pour considérer une technique comme « Delphi » : l'anonymat, l'itération, le retour contrôlé et l'agrégation statistique des réponses. Étant donné que cette recherche a été menée en collaboration avec des experts répartis au niveau mondial (26 pays représentés), seule une variante de la technique Delphi a été adoptée sur la base des recommandations de Rowe et Wright (1999: 354-355).

En somme, l'étape 2 combine les approches qualitatives et quantitatives. Il s'agit plus précisément : d'une part, de sélectionner des logiciels dont le type a préalablement été identifié via le processus qualitatif déjà présenté et d'autre part, de recruter des experts afin de leur soumettre un questionnaire leur permettant d'évaluer des logiciels préalablement sélectionnés.

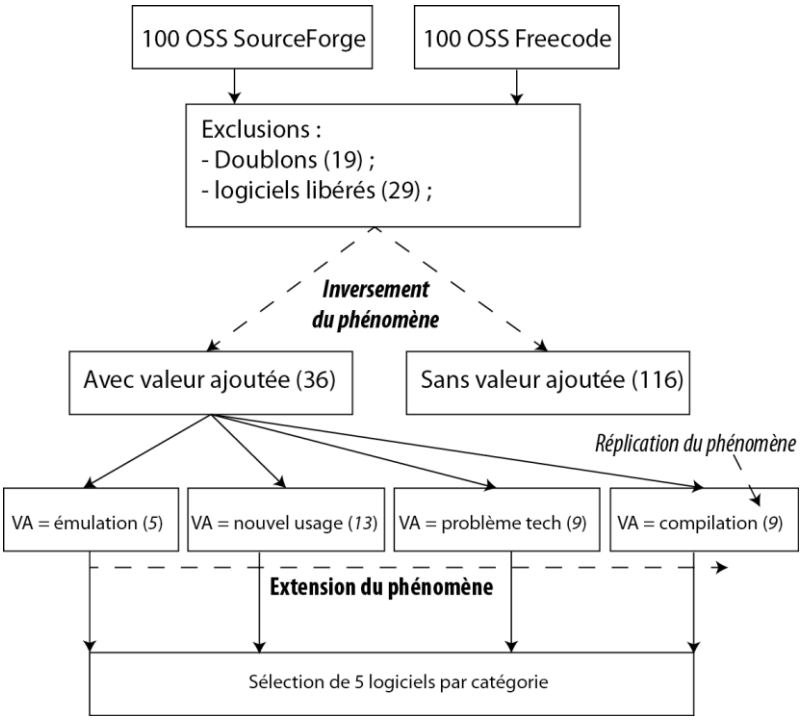
- *Données*

Un questionnaire (**Annexe 2**) a été conçu avec un double objectif : d'une part, tester les hypothèses susmentionnées ; d'autre part, confronter la classification qualitative avec celle émanant d'experts de l'open source (étape 3). Ce questionnaire a été conçu en collaboration avec des experts qui ont commenté les différentes versions par courrier électronique. Une synthèse des apports des experts sur ce questionnaire est fournie en **Annexe 2**.

Pour ce faire, une liste de 152 logiciels libres a été conçue à partir des 100 logiciels les plus populaires des sites SourceForge (100) et Freecode (100) en éliminant les doublons. La sélection de logiciels libres populaires est justifiée par le fait que les logiciels doivent être suffisamment connus par les experts pour être évalués de manière satisfaisante. En effet, une étude pilote a été menée, à l'image de celle de Feller et al. (2008), pendant laquelle cinq experts ont évalué 25 logiciels libres populaires ou non. Toutefois, certains logiciels n'ont pas été évalués car ils étaient inconnus des répondants. Comme l'a souligné un expert lors de cette phase, *« c'est difficile de se prononcer utilement sur des [logiciels] qu'on ne connait pas »* (Open Source Architect, Thales). De plus, la littérature a montré que la popularité est un paramètre important pour évaluer la qualité (Capra et al. 2011: 146) et le succès d'un logiciel open source (Midha et Palvia 2012). Enfin, les précédents travaux ont souligné la nécessité d'analyser l'innovation des produits open source les plus populaires (Rossi 2009, p. 165).

Pour chaque logiciel, des informations ont été rassemblées à propos de : ses principales fonctionnalités, sa/ses valeur(s) ajoutée(s) comparée(s) aux logiciels fermés existants, son origine. Ces données ont été récoltées grâce à des informations publiques, complétées par l'échange de 50 messages avec des acteurs clés des projets. Elles ont été codées en utilisant le même processus que celui de l'étape 1 (7 phases du processus de codage). Excepté la dernière phase (classification qualitative) du processus de codage des données qui a aussi conduit à la sélection de logiciels pour une évaluation par des experts. Cette classification est résumée ci-dessous.

Figure 2 : classification qualitative et sélection (étape 2)



Comme explicité dans la figure ci-dessus, une sélection de cinq logiciels de chaque catégorie préalablement (classifiés qualitativement) a été réalisée, soit 25 logiciels au total (cf. le **Tableau 7**). Cette sélection s'est basée sur les principes de l'échantillonnage théorique (Eisenhardt et Graebner 2007) afin d'obtenir un échantillon maximisant les phénomènes d'inversement (sélection de logiciels avec et sans valeur ajoutée), d'extension (logiciels de chaque type) et de réplication (sélection de 5 logiciels par type).

Tableau 7 : logiciels sélectionnés pour l'évaluation

Nom	Définition	Classification qualitative
7-Zip	Utilitaire de compression de fichiers	Alternative libre
ALSA Driver	Pilote son pour Linux	Pièce d'adaptation
BitTorrent	Protocole de partage de fichiers en P2P	Orienté nouvel usage
DOSBox	Émulateur Dos	Émulateur
FileZilla	Client FTP	Alternative libre
Gimp	Programme d'édition d'images	Alternative libre
KNOPPIX	Distribution Linux en live CD	Package
Linux NTFS	Pilote pour système de fichiers NTFS pour Linux	Pièce d'adaptation
MiKTeX	Distribution de TeX	Package
MinGW	Adaptation d'outils de développement GNU aux systèmes W32	Pièce d'adaptation
Ncurses	Émulation de System V	Émulateur
Nmap	Scanner de sécurité	Orienté nouvel usage
PDFCreator	Outil de création de fichiers PDF	Alternative libre
PeerGuardian	Outil de protection de la vie privée pour le P2P	Orienté nouvel usage
PHP	Langage de Script	Orienté nouvel usage
Pidgin	Client de messagerie instantané multi-protocole	Orienté nouvel usage
PortableApps.com	Package d'applications portables	Package
Samba	Système d'interopérabilité entre les systèmes Linux/Unix et Windows	Pièce d'adaptation
ScummVM	Émulateur du système Scumm	Émulateur
Util-linux	Suite d'utilitaires pour les systèmes Linux	Package
VirtualDub	Logiciel de retouche vidéo	Alternative libre
VisualBoyAdvance	Émulation de la GameboyAdvance	Émulateur
Wine	Implémentation de Windows pour les systèmes Unix/Linux	Pièce d'adaptation
XAMPP	Installeur Apache, MySQL, PHP et Perl	Package
ZNES	Émulateur de la Super Nintendo	Émulateur

Concrètement, un questionnaire a été administré à 125 experts de l'open source (voir **Annexe 2**) recrutés lors de différentes interactions avec le terrain. Un bon nombre provient d'organisations étudiées dans le cadre d'une thèse. Beaucoup d'autres ont été sélectionnés sur le principe de l'*échantillonnage boule de neige* qui consiste à identifier de nouveaux répondants grâce à d'anciens répondants (Shah 2006). Environ 450 courriers électroniques[15] ont été échangés avec les experts. Des associations d'utilisateurs de logiciels libres ont également servi de relais en postant un lien vers le questionnaire mis à disposition en ligne. Du fait que l'évaluation de logiciels est une tâche complexe, les répondants étaient majoritairement des praticiens. Au final, cette étude rassemble des spécialistes du logiciel libre provenant de diverses institutions : 70 % travaillent dans l'industrie, les services IT (Thales, Google, Alcatel-Lucent, Motorola, etc.) ou des centres de recherche (INRIA, NASA, la Marine Danoise, CNRS, etc.) du monde entier (voir Annexe 1). Pour chaque logiciel, les experts se sont exprimés sur cinq dimensions (cf. en **Annexe 2** les items du questionnaire) en utilisant une échelle de Likert à cinq items (*« Je suis tout à fait d'accord »* ; *« Je suis d'accord »* ; *« Je ne suis pas d'accord »* ; *« Je ne suis pas du tout d'accord »* et enfin *« Je ne sais pas »*).

[15] Calcul basé sur un comptage des archives électroniques.

- *Analyses*

Pour chaque question tous les « Je suis tout à fait d'accord » et « je suis d'accord » ont été additionnés comme autant d'experts exprimant leur accord avec l'affirmation proposée. Dans le but d'avoir des données comparables, les données des évaluations ont été agrégées pour chaque logiciel et rapportées à une même échelle (le pourcentage). Pour ce faire, le nombre d'experts ayant répondu à chaque item a été divisé par le nombre de répondants (voir Annexe 4) ce qui représente 3125 évaluations agrégées (125 évaluations d'experts sur 25 logiciels). En somme, les réponses des experts ont été transformées en pourcentage d'accord avec la question posée. Pour chaque logiciel, un type à priori a été codé sous la forme d'une variable nominale. Après cela, une analyse de la variance (ANOVA) et une analyse en composantes principales (ACP) ont été réalisées afin de valider les hypothèses et valider les variables issues de la phase qualitative.

- *Résultats*

L'ANOVA montre des différences significatives pour les moyennes de toutes les variables dépendantes ($P<0.000$) ou ($P<0.05$) (voir **Annexe 5**). Ceci prouve que les variables sont pertinentes et corrélées avec les évaluations des experts validant ainsi l'ensemble des hypothèses de départ (H1 à H5).

L'ACP a calculé 5 axes expliquant l'ensemble de la variance (**Annexe 5**). Néanmoins, en appliquant la règle *Guttman-Kaiser*, on ne retient que les deux premiers facteurs puisque cette règle prévoit que seuls les axes ayant une valeur propre (ou *eigenvalue*) supérieure à 1 doivent être conservés (Rietveld et Van Hout 1993 p. 273). De même, le test du coude (ou *scree plot test*) montre que seuls les deux premiers facteurs ont une valeur propre supérieure à 1 (Annexe 6). En conséquence, les deux facteurs retenus (voir Tableau 8) expliquent respectivement 0,391 et 0,339 de la variance. Combinés, ils représentent près de 0,73 de la variance ce qui est dans la moyenne acceptable (Rietveld et Van Hout 1993 p. 273). Du fait que la première composante se focalise sur la création d'une fonction nouvelle et la résolution de problème technique celle-ci a été nommée degré d'innovation fonctionnelle. Sur le même principe, la seconde composante se concentrant sur la reproduction de produits existants a été appelée degré de reproduction fonctionnelle (**Figure 3**).

En outre, on constate également deux corrélations négatives majeures. La variable *« alternative libre »* est corrélée négativement avec la composante 1. Ce phénomène est cohérent puisque cette composante mesure le degré d'innovation fonctionnelle et étant donné que la variable *« alternative libre »* mesure la capacité d'un logiciel à reproduire les fonctions d'un autre, il en est l'opposé. La corrélation négative de la variable « *package* » avec l'axe 2 s'explique par le fait que cet axe mesure le degré de reproduction fonctionnelle d'un produit existant. Or, la variable « *package* » mesure quant à elle la capacité d'un logiciel, grâce à la combinaison originale de logiciels libres existants, à proposer un nouvel usage.

Figure 3 : analyse en composantes principales

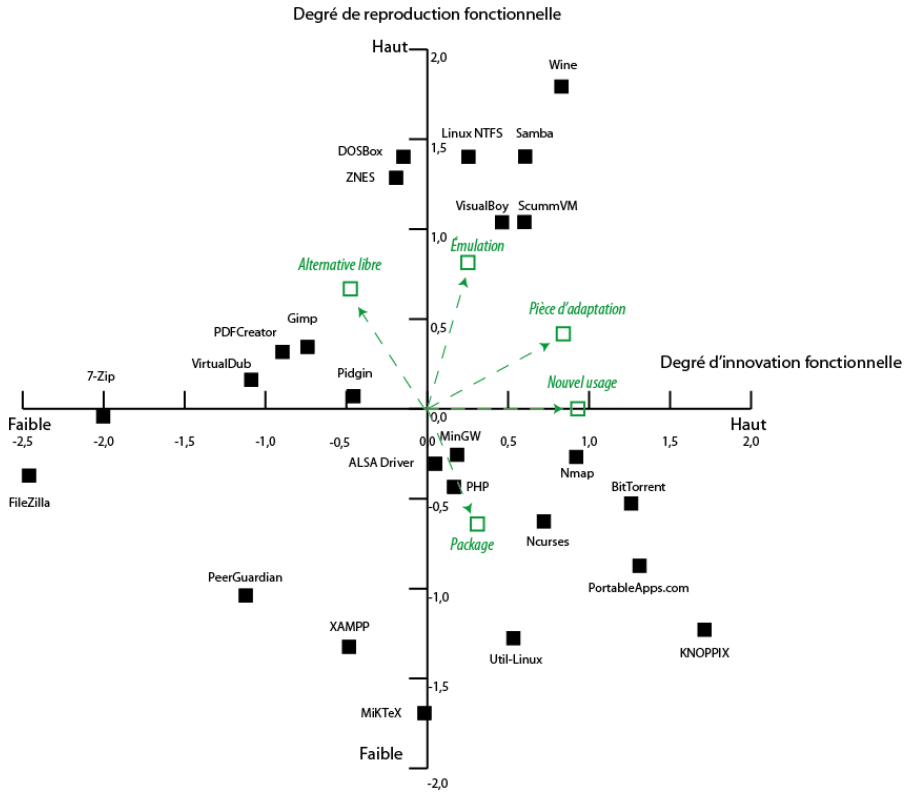


Tableau 8 : composition des axes de l'ACP

	Composantes	
Variables	1 (degré d'innovation fonctionnelle)	2 (degré de reproduction fonctionnelle)
Orienté nouvel usage (variable)	.930	
Pièce d'adaptation (variable)	.841	.418
Émulation (variable)	.251	.814
Alternative libre (variable)	-.476	.668
Package (variable)	.310	-.641

Étape 3 : comparaison des classifications qualitative et quantitative

- *Objectifs*

Dans cette étape, l'objectif est de tester dans quelle mesure la classification qualitative (procédure d'auto-évaluation décrite lors de l'étape 1) et quantitative (classification basée sur les évaluations des experts) convergent ou s'opposent.

- *Données et analyses*

Pour mener à bien cette étape, il convient en premier lieu d'assigner une variable qualitative (le type d'innovation fonctionnelle) à l'aide de variables quantitatives (cinq variables précédemment présentées). Pour ce type de procédure, l'analyse discriminante est une technique pertinente (Droesbeke et al. 2005 p. 137). Ensuite il s'agit de comparer les

classifications qualitative (étape 2) et quantitative (calculée grâce à l'analyse discriminante sur les évaluations des experts).

L'analyse discriminante comprend trois étapes : un test d'égalité des moyennes, une validation de l'étude et une analyse des résultats de la classification (Burns et Burns 2008: chapitre 25).

D'abord, le test d'égalité des moyennes (Lambda de Wilks) est significatif pour toutes les variables ($p < 0.05$) (voir Tableau 9).

Tableau 9 : test d'égalité des moyennes

	Wilks' Lambda	F	df1	df2	Sig.
Pièce d'adaptation (variable)	.592	3.451	4	20	.027
Alternative libre (variable)	.370	8.510	4	20	.000
Émulation (variable)	.318	10.708	4	20	. 000
Orienté nouvel usage (variable)	.615	3.125	4	20	.038
Package (variable)	.175	23.572	4	20	.000

Ensuite, l'analyse discriminante était pertinente puisque le test de Box était également significatif ($p < 0.05$) (voir Tableau 10).

Tableau 10 : Test de Box

Box's M		61,774
F	Approx.	1.681
	df1	24
	df2	1104.425
	Sig.	.021

- *Résultats*

Enfin, les résultats de la classification montrent que 88 % des logiciels (22 sur 25) ont été correctement catégorisés. Ceci confirme que la catégorisation initiale est pertinente (**Tableau 11**).

Tableau 11 : résultats de la classification statistique

		Catégorie	Groupe qualitatif					Total
			Alternative libre	Orienté nouvel usage	Pièce d'adaptation	Package	Émulateur	
Original	**Nb**	Alternative libre	5	0	0	0	0	5
		Orienté nouvel usage	1	4	0	0	0	5
		Pièce d'adaptation	0	0	4	1	0	5
		Package	0	0	0	5	0	5
		Émulateur	0	1	0	0	4	5
	%	Alternative libre	100.0	.0	.0	.0	.0	100.0
		Orienté nouvel usage	20.0	80.0	.0	.0	.0	100.0
		Pièce d'adaptation	.0	.0	80.0	20.0	.0	100.0
		Package	.0	.0	.0	100.0	.0	100.0
		Émulateur	.0	20.0	.0	.0	80.0	100.0
Validation transversale[a]	**Nb**	Alternative libre	5	0	0	0	0	5
		Orienté nouvel usage	1	4	0	0	0	5
		Pièce d'adaptation	1	1	1	1	1	5
		Package	0	0	0	5	0	5
		Émulateur	0	1	1	0	3	5
	%	Alternative libre	100.0	.0	.0	.0	.0	100.0
		Orienté nouvel usage	20.0	80.0	.0	.0	.0	100.0
		Pièce d'adaptation	20.0	20.0	20.0	20.0	20.0	100.0
		Package	.0	.0	.0	100.0	.0	100.0
		Émulateur	.0	20.0	20.0	.0	60.0	100.0

a. La validation transversale (*ou cross validation*) a seulement été réalisée pour ces cas dans l'analyse. Dans la cette validation, chaque cas est classifié par les fonctions dérivées de chaque cas excepté ce cas.

b. 88.0% des cas regroupés originellement ensemble sont correctement classifiés.

c. 72.0% des cas regroupés originellement (validation transversale) ensemble sont correctement classifiés.

Toutefois, l'analyse discriminante laisse apparaître des différences entre la classification qualitative et celle basée sur les évaluations des experts : trois logiciels sur 25 semblent mal classés (voir Tableau 12).

Tableau 12 : classification qualitative et statistique

Logiciel	Catégorie qualitative	Catégorie calculée
7Zip	Alternative libre	Alternative libre
ALSADriver	Pièce d'adaptation	Pièce d'adaptation
BitTorrent	Orienté nouvel usage	Orienté nouvel usage
DOSBox	Émulateur	Émulateur
FileZilla	Alternative libre	Alternative libre
Gimp	Alternative libre	Alternative libre
KNOPPIX	Package	Package
LinuxNTFS	Pièce d'adaptation	Pièce d'adaptation
MiKTeX	Package	Package
MinGW	Pièce d'adaptation	Package
Ncurses	Émulateur	Orienté nouvel usage
Nmap	Orienté nouvel usage	Orienté nouvel usage
PDFCreator	Alternative libre	Alternative libre
PeerGuardian	Orienté nouvel usage	Orienté nouvel usage
PHP	Orienté nouvel usage	Orienté nouvel usage
Pidgin	Orienté nouvel usage	Alternative libre
PortableApps	Package	Package

Samba	Pièce d'adaptation	Pièce d'adaptation
ScummVM	Émulateur	Émulateur
Utillinux	Package	Package
VirtualDub	Alternative libre	Alternative libre
VisualBoyAdvance	Émulateur	Émulateur
Wine	Pièce d'adaptation	Pièce d'adaptation
XAMPP	Package	Package
Znes	Émulateur	Émulateur

D'abord, MinGW fut classifié comme un package (MinGW regroupe des outils GNU) alors qu'il était rangé dans la catégorie des pièces d'adaptation : il s'agit aussi d'une adaptation des outils GNU pour les systèmes Windows. MinGW signifie : Minimalist GNU pour Windows (MinGW Team 2008). Ensuite, Ncurses a été considéré comme orienté nouvel usage alors qu'il était vu comme un émulateur en se basant sur la définition de cette bibliothèque (Free Software Foundation 2007). Enfin, Pidgin fut classifié comme alternative libre alors qu'il était parmi les logiciels orientés nouvel usage. En effet, selon un des experts *« Pidgin [est] innovant dans le sens où c'est un tout protocole en un seul logiciel »* (Expert 2). Pidgin a probablement été considéré comme une alternative libre aux clients de messagerie instantanée existants. En somme, les différences entre la classification qualitative et statistique proviennent d'une raison principale : il existe des logiciels libres relevant de deux catégories. Néanmoins, la classification qualitative est une bonne approximation du jugement moyen des experts.

Étape 4 : modélisation des types d'innovation fonctionnelle

- *Objectifs*

Dans l'étape précédente, l'analyse discriminante a permis d'assigner des variables qualitatives sur la base de données quantitatives (évaluations agrégées). Il s'agit désormais de modéliser les types d'innovation fonctionnelle préalablement identifiés (qualitativement) et validés (quantitativement) afin que ces derniers puissent être généralisés et être réutilisables pour la théorie et la pratique.

- *Données*

Pour chaque groupe, la moyenne des évaluations de tous les logiciels de la classe a été calculée. Le tableau ci-dessous synthétise ces données.

Tableau 13 : caractéristiques des groupes (abrégés « G »)

	Pièce d'adaptation (variable)	Alternative libre (variable)	Émulation (variable)	Orienté nouvel usage (variable)	Package (variable)
G1 : Alternative libre	56,89	74,65	17,46	41,06	26,27
G2 : Pièce d'adaptation	92,18	70,49	60,22	61,44	28,95
G3 : Nouvel usage	74,04	37,71	8,61	68,61	20,82
G4 : Émulateur	79,55	54,29	87,12	55,15	13,53
G5 : Package	73,43	42,10	23,62	62,06	84,49
Moyenne	73,56	55,53	35,16	57,13	37,54
Écart type	17,61	18,12	30,21	15,38	27,48

- *Analyses*

Grâce aux statistiques calculées sur les évaluations d'experts, des types d'innovation dans le champ de l'open source sont proposés. Chaque type a été modélisé sur la base des caractéristiques des groupes de logiciels calculés grâce à l'agrégation des évaluations des experts (étape 2). Par exemple, pour le premier groupe de logiciels (G1), une grande majorité d'experts (près de 75%) a estimé que la seule valeur ajoutée de ces logiciels était le fait d'être des alternatives libres à des produits existants. Ces logiciels ont un score très bas, bas ou moyen sur toutes les autres variables. À contrario, pour le troisième groupe de logiciels (G3), on constate que ces logiciels ont reçu un haut score pour les variables de résolution de problèmes et de création de nouvel usage. Toutefois, ces derniers ont reçu un faible score sur les autres dimensions. Le tableau ci-dessous résume les caractéristiques de chaque type d'innovation fonctionnelle sur la base des données agrégées (voir ci-dessus).

Tableau 14 : résumé des types d'innovation fonctionnelle

	Pièce d'adaptation (variable)	**Alternative libre (variable)**	**Émulation (variable)**	**Nouvel usage (variable)**	**Package (variable)**
G1 : Alternative libre (type)	Moyen	Haut	Très bas	Moyen	Bas
G2 : Pièce d'adaptation (type)	Très Haut	Haut	Haut	Haut	Bas
G3 : Orienté nouvel usage (type)	Haut	Bas	Très Bas	Haut	Bas
G4 : Émulateur (type)	Haut	Moyen	Très Haut	Moyen	Très Bas
G5 : Package (type)	Haut	Moyen	Bas	Haut	Très Haut

Si la valeur était :	
Moins de 20*	Très bas
Entre 20 et 40	Bas
Entre 40 et 60	Moyen
Entre 60 et 80	Haut
Entre 80 et 100	Très haut

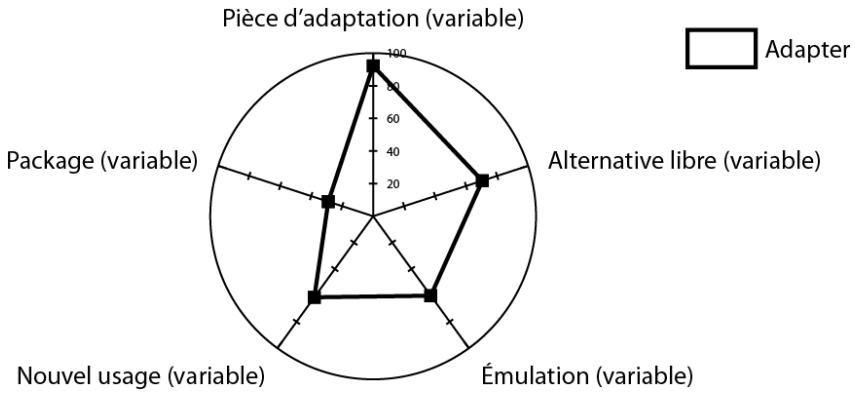
***Pourcentage d'experts.**

Afin de faciliter l'appropriation et la réutilisation de cette typologie, il conviendra de détailler chaque type d'innovation fonctionnelle. D'abord, l'alternative libre (1) affiche un faible niveau sur les variables liées à l'émulation et à la combinaison de produits, un niveau moyen sur les métriques associées à la résolution de problème technique et à l'usage puisque ces logiciels se contentent de répliquer un logiciel existant pour constituer une alternative. La pièce d'adaptation (2) présente un très haut niveau pour la variable mesurant la capacité à résoudre un problème non résolu, un haut niveau sur les variables mesurant l'alternative (ce type de logiciel constitue une alternative à une solution fermée), l'émulation (la plupart du temps ces logiciels simulent le fonctionnement d'un ensemble de logiciels ou d'un matériel), le nouvel usage (ces logiciels permettent de réaliser une ou plusieurs fonctions inédites). À l'inverse, l'orienté nouvel usage (3) créé un usage inédit en résolvant un problème technique mais ne constitue ni une alternative libre (puisqu'aucune fonctionnalité similaire n'existait), ni une émulation ou un regroupement de produit (les valeurs associées à ces variables sont basses

voire très basses). L'émulateur (4) offre une fonctionnalité qui existait mais sous une forme différente. On note un niveau très haut pour la variable relative à l'émulation et haut pour la résolution de problème technique. Enfin, le package (5) présente un très haut niveau pour la variable liée à la combinaison de produits, un niveau élevé pour les variables correspondant à l'usage inédit et à la résolution de problème.

Cette typologie de l'innovation fonctionnelle permet de classifier de manière rationnelle des logiciels open source sur la base de critères validés empiriquement. La **Figure 4** offre un exemple de synthèse graphique d'un type d'innovation (la pièce d'adaptation).

Figure 4 : représentation synthétique de la pièce d'adaptation



CONTRIBUTIONS ET IMPLICATIONS DE LA RECHERCHE

Les trois grandes limites de la littérature ont été détaillées en matière d'appréciation de l'innovation des logiciels open source : l'apparente confusion entre processus et produit innovant, le manque de critères pour l'évaluation de l'innovation des logiciels et la faible robustesse des procédures d'évaluation. La première limite a été dépassée par le *design* même de cette recherche qui a conduit à une délimitation claire du champ d'étude : à savoir l'appréciation de l'innovation produit des logiciels open source. De façon implicite, cette recherche exclut tous les éléments relevant des processus de développement. Les deux autres limites (manque de critères, robustesse de la procédure) reçoivent une réponse détaillée ci-dessous.

Contribution 1 : une typologie de l'innovation fonctionnelle dans l'open source

Cet article contribue au corpus montant en matière de recherche sur les spécificités de l'innovation dans le secteur du logiciel. Les précédentes recherches offraient des résultats mitigés voire complètement contradictoires car notamment ces dernières reposaient sur une définition différente de la notion d'innovation. Et cela en raison de l'inadéquation des métriques traditionnelles de l'économie et du management pour caractériser l'innovation dans le secteur du logiciel. Par exemple, le diptyque incrémental/radical (Dewar et Dutton 1986) présente de sérieuses limites pour analyser l'innovation d'un logiciel. Outre le caractère dualiste de l'analyse, on considérera qu'un logiciel constitue une *innovation radicale* si et seulement si ce dernier se base sur une technologie nouvelle (algorithme, langage, modèle). Pourtant, l'innovation dans le domaine des TI ne se résume pas à la technologie (Carr 2003) et dans le champ du logiciel, elle peut être purement fonctionnelle (Lippoldt et Stryszowski 2009). De même, l'analyse de l'innovation par le biais du couple maintien/rupture (Christensen 2006) ou encore l'utilisation des critères du Manuel d'Oslo (OECD/Eurostat 2005) n'a de sens que lorsqu'il existe un marché pour le produit analysé. Lorsque le produit est distribué gratuitement, comme c'est le cas de la majorité des logiciels open source, ces modèles sont inopérants.

Par ailleurs, une des principales limites des typologies dichotomiques est le fait qu'elles fournissent une vision simpliste (Garcia et Calantone 2002) ou binaire de la réalité alors que celle-ci est complexe et multiple (Mingers 2000 ; Mingers et Brocklesby 1997). Par exemple, un produit ne peut être *à la fois* une innovation de maintien et de rupture (Christensen 2006 ; Christensen et Raynor 2003), incrémentale et radicale (Dewar et Dutton 1986 ; Ettlie et al. 1984) ou encore continue et discontinue (Veryzer Jr. 1998). À l'inverse, une typologie de l'innovation doit permettre la description de formes hybrides combinant plusieurs classes

d'innovation. Par exemple, MinGW (un logiciel analysé dans cette étude) est à la fois rattachable à la catégorie des *packages* (c'est un assemblage de plusieurs logiciels) et à celle des *pièces d'adaptation* (c'est un logiciel qui résout un problème technique).

L'évaluation de l'innovation dans le domaine des logiciels nécessite la construction de métriques *ad hoc*. Cette mesure doit prendre en considération les usages que permettent les technologies et non seulement la technologie (Chandy et Tellis 1998) ou le rapport au marché (Christensen 2006 ; OECD/Eurostat 2005). Dès lors, cette recherche propose une nouvelle manière d'aborder la question de l'innovation logicielle en introduisant le concept de *valeur ajoutée fonctionnelle* (VAF). Ce concept fait référence à une plus-value offerte par un logiciel par rapport aux produits existants. Pour mettre en évidence la VAF d'un logiciel, il faut répertorier les produits concurrents et rechercher en quoi le produit se distingue ou non de ces derniers du point de vue fonctionnel. Ce concept précise la notion d'innovation fonctionnelle proposée dans la littérature (Lippoldt et Stryszowski 2009).

Les précédentes recherches ont montré que l'évaluation des logiciels libres nécessitait des métriques spécifiques (Capra et al. 2011 ; Chen et al. 2010 ; Lundell et al. 2010 ; Taibi et al. 2007) et plus particulièrement en matière d'analyse de l'innovation (Rossi 2009). Dans cette étude, une typologie basée sur la notion de VAF a été proposée. Elle est dédiée à l'open source mais peut être étendue à d'autres champs des TI dans des recherches ultérieures. Cette modélisation offre la première typologie de l'innovation fonctionnelle des logiciels open source s'appuyant sur l'analyse de nombreux produits et avec le concours de plus de cent experts. Avec les connaissances actuelles, il n'est pas raisonnablement possible de mesurer l'innovation de produits hétérogènes comme un logiciel, une machine à café ou un sac à main avec les mêmes métriques. Les typologies génériques se basant sur des notions (ex : radicalité technologique, brevets, marques) largement débattues ou remises en question par la théorie et la pratique (Garcia et Calantone 2002 ; Le Masson et al. 2010). Cet article ouvre la voie à une approche spécialisée (propre à chaque secteur d'activité) de la mesure de l'innovation.

Contribution 2 : un cadre pour la modélisation de l'innovation (CMI)

La littérature a montré que l'évaluation des TI est une tâche complexe (Irani et Love 2001) et constitue un challenge contemporain (Benlian 2011). De même, l'appréciation de l'innovation est une activité complexe (Garcia et Calantone 2002). Les précédentes études sur l'innovation des logiciels libres présentaient d'évidentes limites en matière méthodologique : que ce soit en ce qui concerne l'utilisation de la technique de l'autoévaluation s'appuyant elle-même sur des critères obscurs (Deek et McHugh 2007) voire non empiriquement validés (Klincewicz 2005) ou encore une évaluation Delphi basée sur un nombre très restreint d'experts (Rossi 2009).

À l'inverse, cet article adopte une posture méthodologique combinant des méthodes *dures* et *souples* (Mingers 2000) et démontre que celle-ci est particulièrement pertinente pour évaluer l'innovation. Pour ce faire, cette étude combine deux grandes techniques d'évaluation souvent utilisées séparément dans la littérature à savoir : l'autoévaluation (Benlian 2011 ; Klincewicz 2005) et le jugement d'expert (Rossi 2009). Sur la base du déroulé de cette recherche, un Cadre pour la Modélisation de l'Innovation (CMI) est proposé. Le CMI comprend les étapes suivantes : (1) identifier et sélectionner des produits innovants, (2) découvrir et coder des types d'innovation, (3) tester les types d'innovation, (4) modéliser l'innovation. Afin d'assurer l'appropriation et d'adaptation du CMI, il convient de décrire les éléments essentiels de chaque étape.

Identification et sélection des produits innovants

L'identification de produits semblant innovant peut se baser sur : une ou plusieurs listes de produits existantes et répertoriant des produits connus pour leur performance, une recommandation par des experts, un ou plusieurs article(s) de presse professionnelle comparant des produits. Il s'agit d'utiliser les principes de l'échantillonnage théorique et de l'étude de cas multiple (Eisenhardt 1989 ; Eisenhardt et Graebner 2007) afin de sélectionner des produits qui maximisent les phénomènes de réplication (répétition du même type d'innovation), d'extension (complémentarité des produits pour rendre compte de la pluralité de l'innovation dans le champ étudié) et d'inversement (distinction des produits ayant une valeur ajoutée de ceux qui n'en ont pas, opposition des types de produits identifiés). L'objectif étant de passer d'une réalité complexe et multiple à un nombre de cas limités permettant une étude approfondie de chaque cas.

Découverte et codage des types d'innovation

Grâce aux données récoltées lors de la phase précédente il s'agira d'analyser les données manuellement ou à l'aide d'un outil d'analyse qualitative comme un logiciel d'analyse textuelle par exemple. L'objectif étant de découvrir des catégories d'innovation et d'en dériver des variables et ce en restant très proche des données, en suivant les principes de la théorie enracinée (Glaser et Strauss 1967 ; Miles et Huberman 1994) et de l'étude de cas multiple (Eisenhardt et Graebner 2007). Ce processus devra de préférence être mené avec le concours de praticiens ou d'experts du domaine de référence.

Test des types d'innovation

Il s'agit ici de dériver des hypothèses à valider (ou à invalider) et à bâtir un questionnaire qui sera administré à des experts du domaine. La littérature recommande un minimum de 10 experts pour les analyses de type Delphi (Gallego et al. 2008 ; Okoli et Pawlowski 2004). Ces experts pourront être des praticiens ou des utilisateurs. Le questionnaire devra proposer une liste de produits préalablement sélectionnés et classifiés de façon qualitative (cf. étape 1). Plus précisément, il faudra attribuer un type *à priori* et le coder sous la forme d'une variable nominale dans un logiciel de statistiques. Suite à cela les experts devront positionner les produits dans la ou les catégories pertinentes à l'aide d'une échelle de Likert par exemple. Il est également possible d'utiliser une variable booléenne néanmoins il faudra adapter les tests statistiques en conséquence. L'objectif étant d'obtenir un consensus d'experts. Les données peuvent être analysées grâce à une analyse de la variance (ou un autre test en fonction du type de variable choisi). Les hypothèses seront validées ou non. En cas de non validation, il faudra revoir la classification et retourner à l'étape 2. Une analyse en composantes principales (ou une analyse similaire) pourra être utilisée car elle permettra d'avoir une visualisation graphique des ensembles de produits. L'analyse discriminante (AD) servira essentiellement à comparer le classement à priori et la classification basée sur l'agrégation du jugement d'expert. L'AD permettra de valider ou d'invalider le classement à priori. Plus le nombre de cas correctement classifié est important plus la classification qualitative est pertinente. Les cas mal classifiés (s'il y en a) devront être analysés en profondeur afin de comprendre les raisons de cette mauvaise classification.

Modélisation des types d'innovation

À partir des données recueillies sur les produits testés, il conviendra de modéliser les types d'innovation. Cette modélisation devra s'appuyer sur les caractéristiques des groupes de produits issus de la classification quantitative (évaluation d'experts agrégées). Elle pourra

également s'appuyer sur des exemples de produits afin que celle-ci soit moins abstraite. Le tableau ci-dessous synthétise le CMI et ses étapes.

Tableau 15 : les étapes du CMI

Étapes	Détail	Méthodologies pertinentes
1. Identifier et sélectionner des produits innovants	Identifier des produits semblant innovants. Ensuite sélectionner les produits qui maximisent les réplications, extensions ou différences.	Échantillonnage théorique et étude de cas multiple (Eisenhardt 1989 ; Eisenhardt et Graebner 2007 ; Fitzgerald 1997)
2. Découvrir et coder des types d'innovation	Analyser les données (manuellement ou par le biais d'un outil dédié) pour découvrir des catégories à priori et dériver des variables.	Théorie enracinée (Glaser et Strauss 1967 ; Miles et Huberman 1994)
3. Tester les types d'innovation	Créer un questionnaire pour tester les types d'innovation par le biais de l'évaluation d'experts ou d'utilisateurs. Comparaison des classifications qualitative et quantitative.	Évaluation de type Delphi (Rowe et Wright 1999), évaluation d'utilisateurs (Beaudry et Pinsonneault 2005) ; statistiques descriptives.
4. Modéliser l'innovation	Modéliser les types d'innovation sur la base de données statistiquement agrégées.	Statistiques (Burns et Burns 2008)

Cet article a montré que les métriques traditionnelles de l'économie et du management de l'innovation ne permettaient pas de rendre compte de l'innovation dans le domaine des logiciels. En effet, dans les secteurs comme les TI ou les activités traditionnelles (Alcaide-Marzal et Tortajada-Esparza 2007), les praticiens et académiques doivent adopter une posture originale et novatrice. Le CMI propose un cadre générique pour la création de typologies de l'innovation contingente au secteur d'activité étudié. Toutefois, davantage de recherches sont nécessaires sur l'utilisation de la multiméthodologie pour l'évaluation de l'innovation en SI/TI et dans d'autres champs de recherche.

CONCLUSION

Depuis une dizaine d'années, les utilisateurs ont montré qu'ils étaient capables de créer et maintenir des produits technologiques de façon autonome (Von Hippel 2005). Wikipédia et Linux sont les exemples les plus populaires. Par le biais de la combinaison de méthodologies qualitatives et quantitatives, cette étude contribue à la littérature en matière de typologie et d'approche pour l'évaluation de l'innovation. Elle présente toutefois d'importantes limites qui constituent autant de *challenges* pour la recherche.

La première limite de cette étude concerne le type de produits analysés. En effet, l'utilisation de données provenant de *forges* ou de sites grand public a introduit un biais relatif au type de logiciels open source. Cette typologie s'est appuyée sur des logiciels open source créés par des utilisateurs. Davantage de recherches empiriques sont nécessaires pour comparer les logiciels conçus par les autres types d'organisations impliquées dans l'open source comme les réseaux d'affaires (Feller et al. 2008) ou les communautés hybrides (Capra et al. 2011). Ceci permettra d'améliorer et d'enrichir cette typologie afin de mieux rendre compte de la diversité de ce champ. En outre, cette recherche prouve que la multiméthodologie constitue une approche pertinente pour apprécier l'innovation des produits. L'exercice s'est avéré particulièrement difficile. Le cadre proposé ouvre une première voie dans l'utilisation de la multiméthodologie pour l'évaluation de l'innovation. Cependant, il faut davantage de recherches mobilisant la multiméthodologie pour l'évaluation de l'innovation en SI/TI mais aussi dans d'autres champs de recherche afin de mieux connaître cette approche, en révéler son potentiel.

REMERCIEMENTS

L'auteur souhaite remercier Anthony Di Benedetto, Mathias Béjean, Michel Bigand, Laurent Clévy, Marc-Aurèle Darche, Albert David, Serge Druais, Sophie Gautier, Viet Ha, Armand Hatchuel, Ola Henfridsson, Youness Lemrabet, Grégory Lopez, Frank Piller, Frantz Rowe, François-Xavier de Vaujany, Besoa Rabenasolo, Donald White, Stuart Williams et tous les participants à cette étude.

De très grands remerciements sont adressés aux relecteurs anonymes de *SIM* dont le travail a permis à ce manuscrit d'évoluer de façon radicale. Un grand merci à Yves Pigneur pour sa pédagogie pendant le processus de révision.

Annexe 1 : profils des experts

Tableau 16 : Secteurs d'activité

Champs	Nombre[1]	Pourcentage
Industrie et services IT	62	59.62%
Universités/Écoles	22	21.15%
Services (hors IT)	11	10.58%
Centres de recherché	9	8.65%
Total	104	100%

[1] Les données n'étaient pas disponibles pour 21 répondants

Tableau 17 : Postes occupés

Emplois	Nombre[2]	Pourcentage
Développeurs	29	25,22%
Administrateurs systèmes	19	16,52%
Étudiants	18	15,65%
Expert/Architecte/Ingénieur logiciel	18	15,65%
Autres	11	9,57%
Chercheurs (académiques)	8	6,96%
Directeurs Techniques et R&D	7	6,09%
Managers de Projets	3	2,61%
Dirigeants	2	1,74%
Total	115	100,00%

[2] Les données n'étaient pas disponibles pour 10 répondants

Tableau 18 : Expérience en développement

Expérience	Nombre	Pourcentage
Plus de 10 ans	41	32,80%
Entre 5 et 10 ans	23	18,40%
Entre 3 et 5 ans	14	11,20%
Entre 1 et 3 ans	13	10,40%
Moins d'un an	5	4,00%
Ne développe pas	29	23,20%
Total	125	100,00%

Tableau 19 : Origine géographique

Zones géographiques	Nombre[3]	Pourcentage
Europe	74	62,18%
Asie	14	11,76%
Amérique	26	21,85%
Afrique	1	0,84%
Océanie	4	3,36%
Total	119	100,00%

[3] Les données n'étaient pas disponibles pour 6 répondants

Annexe 2 : questions posées

- Pour les alternatives libres : *Ce logiciel libre est avant tout une alternative libre à un logiciel « propriétaire » (closed source) existant. Exemple : aMSN est un clone libre du client de messagerie MSN Messenger.*
- Pour les émulateurs : *Ce logiciel libre émule le fonctionnement d'un matériel physique ou d'un ensemble d'applications logicielles (ex : système d'exploitation, moteur, etc.) : Exemple : PCSX2 est un logiciel qui émule le fonctionnement de la console Playstation 2.*
- Pour les packages : *Ce logiciel libre est principalement un ensemble de logiciels libres. Exemple : EasyPHP réunit Apache, MySQL et PHP en un seul logiciel.*
- Pour les pièces d'adaptation : *Ce logiciel libre a résolu un problème technique qui jusqu'ici n'avait pas été résolu. Exemple : Mono est le premier logiciel qui a permis d'exécuter des applications .Net sur un système Linux.*
- Pour les orientés nouvel usage : *Ce logiciel libre a créé un nouvel usage. Exemple : Mute est un logiciel de P2P (Peer to Peer) permettant de préserver l'anonymat de ses utilisateurs. Mute a ajouté la notion d'anonymat si on le compare aux logiciels de P2P existants lorsqu'il a été initié.*

Annexe 3 : synthèse des apports des experts sur le questionnaire

ID	Expert	Remarque
1	Open source Architect, Thales	Problème de non connaissance de certains logiciels.
2	Open source Architect, Thales	Faute de frappe dans le nom d'un logiciel : 2SNES au lieu de ZNES.
3	Open source Architect, Thales	Citation de Freecode comme source pour avoir des logiciels populaires.
4	Open source Architect, Thales	Nécessité d'avoir un meilleur équilibre entre logiciels "grand-public" et logiciels "professionnels".
5	Chercheur informatique	Ajout de la possibilité "je ne sais pas"
6	Développeur open source 1	Correction de fautes dans la version anglaise.
7	Expert open source, SSLL[16]	Proposition de s'adresser à des développeurs en priorité (impact sur le choix des contacts à qui le questionnaire a été envoyé).
8	Ingénieur développement, SSLL	Suggestion de mise en ligne du questionnaire en ligne pour qu'il soit téléchargé par les participants.
9	Ingénieur développement, SSLL	Proposition de création d'un formulaire en ligne pour automatiser le traitement des questionnaires.
10	Ingénieur développement, SSLL	Problème de non connaissance de certains logiciels.
11	Ingénieur développement, SSLL	Redondance des questions 3 et 6 (question modifiée en conséquence).
12	Ingénieur développement, SSLL	Problème de présentation.
13	Ingénieur développement, SSLL	Écrire un préambule au questionnaire.
14	Développeur open source 2	Proposition de traduction du questionnaire en espagnol (inachevée).
15	Membre d'un forum	Correction du questionnaire version anglaise.
16	Chercheur, Alcatel-Lucent	Remarque sur la liste de logiciels choisie.
17	Chercheur, Alcatel-Lucent	Proposition d'utiliser un système de questionnaire en ligne.
18	Chercheur, Alcatel-Lucent	Ajout d'une case commentaires.
19	Chercheur, Alcatel-Lucent	Discussion sur le statut des innovations techniques.
20	Chercheur, Alcatel-Lucent	Modulation de la réponse en fonction du répondant.
21	Chercheur, Alcatel-Lucent	Introduction du problème des connaissances du répondant.
22	Chercheur, Alcatel-Lucent	Ambiguïté de la question 4 sur l'émulation.
23	Chercheur, Alcatel-Lucent	Remarque sur la notion de besoin.
24	Chercheur, Alcatel-Lucent	Introduction de "je ne sais pas".
25	Chercheur, Alcatel-Lucent	Problème de compréhension de la question sur les packages.
26	Chercheur, Alcatel-Lucent	Suggestion d'avoir un exemple pour savoir comment répondre à chaque question.
27	Chercheur, Alcatel-Lucent	Problème de connaissance de l'histoire du logiciel.
28	Chercheur, Alcatel-Lucent	Question 10 jugée trop vaste.
29	Chercheur, Alcatel-Lucent	Suggestion de nouveaux items sur le volume de la contribution.
30	Chercheur, Alcatel-Lucent	Suggestion de nouvelle question : activité professionnelle ou hobby.
31	Chercheur, Alcatel-Lucent	Question de compréhension sur le statut de la résolution de problème technique.
32	Chercheur, Alcatel-Lucent	Question sur le statut du portage de logiciels.
33	Chercheur, Alcatel-Lucent	Question de compréhension sur le statut des émulations.
34	Chercheur, Alcatel-Lucent	Remarque sur le lien entre innovation et besoin.
35	Chercheur, Alcatel-Lucent	Proposition de reformulation de la question sur les packages.
36	Chercheur, Alcatel-Lucent	Confirmation de la proposition d'ajout d'exemple par question.
37	Chercheur, Alcatel-Lucent	Remarque générale pour fermer davantage les questions.
38	Chercheur, Alcatel-Lucent	Reformulation de la question sur le Top 5 des logiciels les plus innovants.
39	Chercheur, Alcatel-Lucent	Remarque sur l'émulation (suite).

[16] Société de Services spécialisée en Logiciels Libres.

Annexe 4 : Données agrégées

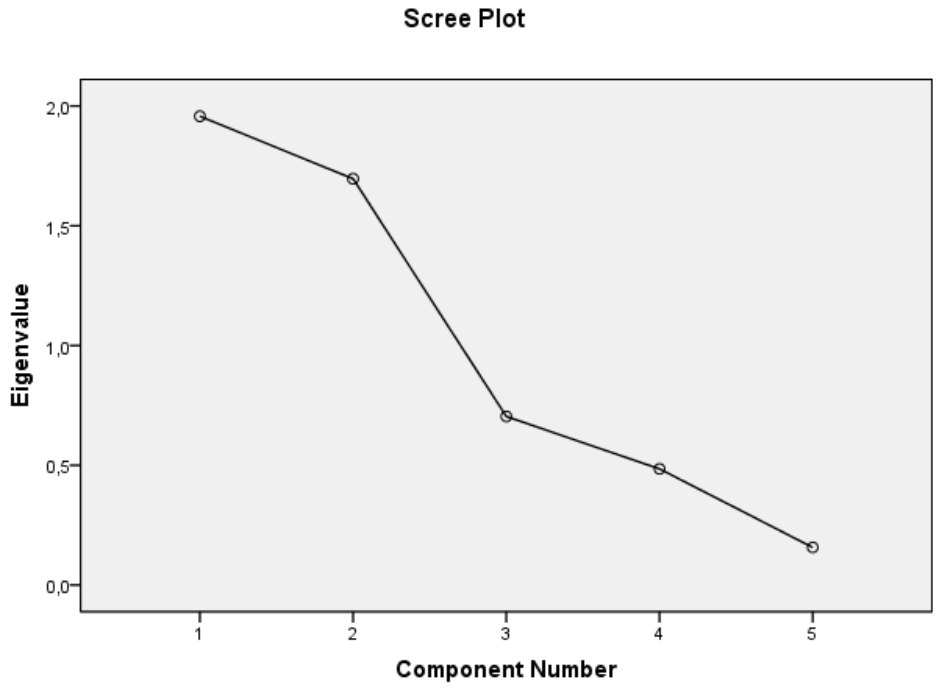
Logiciel	Catégorie qualitative	Pièce d'adaptation (variable)	Alternative (variable)	Émulation (variable)	Nouvel Usage (variable)	Package (variable)
7Zip	Free Alternative	38.6	79.2	17.4	33.8	24.7
ALSADriver	Adapter	78.7	42.3	30.2	49.2	39.7
BitTorrent	Orienté nouvel usage	86.8	28.2	8.6	84.6	13.1
DOSBox	Émulateur	81.5	69.0	82.7	48.8	18.6
FileZilla	Alternative libre	28.0	70.4	11.6	25.0	18.8
Gimp	Alternative libre	75.0	80.0	11.5	47.6	25.8
KNOPPIX	Package	82.6	36.4	25.6	89.3	92.8
LinuxNTFS	Adapter	95.7	81.3	57.0	59.2	21.7
MiKTeX	Package	55.2	27.6	12.5	55.2	74.4
MinGW	Adapter	75.9	60.0	49.1	53.1	82.5
Ncurses	Émulateur	78.4	31.4	12.1	72.5	22.4
Nmap	Orienté nouvel usage	90.9	43.3	9.6	74.5	24.2
PDFCreator	Alternative libre	61.8	77.2	21.2	42.6	33.3
PeerGuardian	Orienté nouvel usage	46.7	38.9	2.4	40.9	25.0
PHP	Orienté nouvel usage	67.4	46.7	10.3	70.5	19.4
Pidgin	Orienté nouvel usage	68.6	66.7	24.4	53.2	32.8
PortableApps	Package	85.7	47.2	30.2	77.5	93.5
Samba	Adapter	98.1	80.0	62.0	66.7	26.9
ScummVM	Émulateur	85.3	51.4	81.1	63.2	18.2
Utillinux	Package	87.0	36.7	10.0	50.0	87.8
VirtualDub	Alternative libre	69.4	74.4	18.8	44.1	22.2
VisualBoyAdvance	Émulateur	80.0	42.3	91.8	58.6	11.4
Wine	Adapter	96.3	78.4	91.7	70.7	27.4
XAMPP	Package	54.3	44.7	14.3	47.4	76.1
ZNES	Émulateur	71.4	54.5	92.9	50.0	5.9

Moyenne	74	56	35	57	38
Écart type	17.6	18.1	30.2	15.4	27.5
Maximum	98.1	81.3	92.9	89.3	93.5
Minimum	28.0	27.6	2.4	25.0	5.9

Annexe 5 : ANOVA des catégories

		Somme des carrés	df	Moyenne des carrés	F	Sig.
Pièce d'adaptation (variable)	Entre groupes	3164.170	4	791.043	3.451	.027
	Dans les groupes	4584.469	20	229.223		
	Total	7748.640	24			
Alternative libre (variable)	Entre groupes	5168.638	4	1292.159	8.510	.000
	Dans les groupes	3036.874	20	151.844		
	Total	8205.511	24			
Émulation (variable)	Entre groupes	15549.727	4	3887.432	10.708	.000
	Dans les groupes	7260.716	20	363.036		
	Total	22810.443	24			
Nouvel usage (variable)	Entre groupes	2275.034	4	568.759	3.125	.038
	Dans les groupes	3640.026	20	182.001		
	Total	5915.060	24			
Package (variable)	Entre groupes	15571.418	4	3892.855	23.572	.000
	Dans les groupes	3303.005	20	165.150		
	Total	18874.423	24			

Annexe 6 : valeurs propres de l'analyse en composantes principales



Afuah, Allan. (2000) "How much do your co-opetitors' capabilities matter in the face of technological change?" Strategic Management Journal, Vol. 21, n°3: p. 397-404.

Agerfalk, Pär J. et Brian Fitzgerald. (2008) "Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy." MIS Quarterly, Vol. 32, n°2: p. 385-409.

Alcaide-Marzal, Jorge et Enrique Tortajada-Esparza. (2007) "Innovation assessment in traditional industries. A proposal of aesthetic innovation indicators." Scientometrics, Vol. 72, n°1: p. 33-57.

Allen, Robert C. (1983) "Collective Invention." Journal of Economic Behavior and Organization, Vol., n°4: p. 1-24.

Baldwin, Carliss et Eric Von Hippel. (2011) "Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation." Organization Science, Vol. 22, n°6: p. 1399-1417.

Beaudry, Anne et Alain Pinsonneault. (2005) "Understanding User Responses to Information Technology: A Coping Model of User Adaptation." MIS Quarterly, Vol. 29, n°3: p. 493-524.

Benkeltoum, Nordine. (2011a) Gérer et comprendre l'open source. Paris: Presses des Mines.

—. (2011b) "Regards sur les stratégies de détournement dans l'industrie open source." Vie et sciences de l'entreprise, Vol., n°187: p. 72-91.

Benlian, Alexander. (2011) "Is traditional, open-source, or on-demand first choice[quest] Developing an AHP-based framework for the comparison of different software models in office suites selection." European Journal of Information Systems, Vol. 20, n°5: p. 542-559.

Bitzer, Jürgen et Philip J. H. Schröder. (2005) "Bug-fixing and code-writing: the private provision of open source software." Information Economics and Policy, Vol. 17, n°3: p. 389-406.

Bogers, Marcel, Allan Afuah, et Bettina Bastian. (2010) "Users as Innovators: A Review, Critique, and Future Research Directions." Journal of Management, Vol. 36, n°4: p. 857-875.

Bonaccorsi, Andrea, Silvia Giannangeli, et Cristina Rossi. (2006) "Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry." Management Science, Vol. 52, n°7: p. 1085-1098.

Burns, Robert et Richard Burns. (2008) Business Research Methods and Statistics Using SPSS. London: Sage Publications Ltd.

Capiluppi, Andrea, Cornelia Boldyreff, et Klaas-Jan Stol. (2011) "Successful Reuse of Software Components: A Report from the Open Source Perspective." Pp. 159-176 in Open Source Systems: Grounding Research, vol. 365, IFIP Advances in Information and Communication Technology, edited by S. Hissam, B. Russo, M. de Mendonça Neto, and F. Kon: Springer Boston.

Capra, Eugenio, Chiara Francalanci, Francesco Merlo, et Cristina Rossi-Lamastra. (2011) "Firms' involvement in Open Source projects: A trade-off between software structural quality and popularity." Journal of Systems and Software, Vol. 84, n°1: p. 144-161.

Carr, Nicholas G. (2003) "IT Doesn't Matter." Harvard Business Review, Vol. 81, n°5: p. 41-49.

Chandy, Rajesh K. et Gerard J. Tellis. (1998) "Organizing for Radical Product Innovation: The Overlooked Role of Willingness to Cannibalize." Journal of Marketing Research, Vol. 35, n°November: p. 474-487.

Chen, Daoyi, Shahriar Shams, César Carmona-Moreno, et Andrea Leone. (2010) "Assessment of open source GIS software for water resources management in developing countries." Journal of Hydro-environment Research, Vol. 4, n°3: p. 253-264.

Christensen, Clayton M. (1997) The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail. Boston, Massachusetts: Harvard Business School Press.

—. (2006) "The Ongoing Process of Building a Theory of Disruption." Journal of Product Innovation Management, Vol. 23: p. 39-55.

Christensen, Clayton M. et Michael Overdorf. (2000) "Meeting the Challenge of Disruptive Change." Harvard Business Review, Vol. March–April: p. 66-76.

Christensen, Clayton M. et Michael E. Raynor. (2003) The Innovator's Solution: Creating and sustaining successful growth. Boston, Massachusetts: Harvard Business School Press.

CIGREF. (2011) "Maturité et gouvernance de l'Open Source - La vision des grandes entreprises." CIGREF, Paris.

Codenie, Wim , Minna Pikkarainen, Nick Boucart, et Jeroen Deleu. (2011) "Introduction." Pp. 1-18 in The Art of Software Innovation - Eight Practice Areas to Inspire your Business, edited by M. Pikkarainen, W. Codenie, N. Boucart, and J. A. Heredia Alvaro. Berlin Heidelberg: Springer.

Dahlander, Linus et Martin W. Wallin. (2006) "A man on the inside: Unlocking Communities as complementary assets." Research Policy, Vol. 35, n°7: p. 1243-1259.

Dahlin, Kristina B. et Dean M. Behrens. (2005) "When is an invention really radical? Defining and measuring technological radicalness." Research Policy, Vol., n°34: p. 717-737.

Danneels, Erwin. (2004) "Disruptive Technology Reconsidered: A Critique and Research Agenda." Journal of Product Innovation Management, Vol. 21: p. 246-258.

—. (2006) "From the Guest Editor Dialogue on the Effects of Disruptive Technology on Firms and Industries." Journal of Product Innovation Management, Vol. 23: p. 2-4.

Deek, Fadi P. et James A. McHugh. (2007) Open Source: Technology and Policy. New York: Cambridge University Press.

Del Bianco, Vieri , Luigi Lavazza, Sandro Morasca, et Davide Taibi. (2009) "Quality of Open Source Software: The QualiPSo Trustworthiness Model." Pp. 199-212 in Open Source Ecosystems: Diverse Communities Interacting, 13th International Conference on Open Source Systems, IFIP, edited by C. Boldyreff, K. Crowston, B. Lundell, and A. I. Wasserman. Skövde, Sweden: Springer.

Dewar, Robert D et Jane E. Dutton. (1986) "The Adoption of Radical and Incremental Innovations: an Empirical Analysis." Management Science, Vol. 32, n°11: p. 1422-1433.

Droesbeke, J.J., M. Lejeune, et G. Saporta. (2005) Méthodes statistiques pour données qualitatives: Editions Technip.

Ebert, Christof. (2007) "Open Source Drives Innovation." IEEE Software, Vol. 24, n°3: p. 105-109.

—. (2008) "Open Source Software in Industry." IEEE Software, Vol. 25, n°3: p. 52-53.

Eisenhardt, Kathleen M. (1989) "Building Theories from Case Study Research." The Academy of Management Review, Vol. 14, n°4: p. 532-550.

Eisenhardt, Kathleen M. et Melissa E. Graebner. (2007) "Theory building from cases: opportunities and challenges." Academy of Management Journal, Vol. 50, n°25-32: p.

Ettlie, John E., William P. Bridges, et Robert D. O'Keefe. (1984) "Organization strategy and structural differences for radical versus incremental innovation." Management Science, Vol. 30, n°6: p. 682-695.

Fearon, Colm et George Philip. (1998) "Self assessment as a means of measuring strategic and operational benefits from EDI: the development of a conceptual framework." European Journal of Information Systems, Vol. 7, n°1: p. 5-16.

Feller, Joseph, Patrick Finnegan, Brian Fitzgerald, et Jeremy Hayes. (2008) "From Peer Production to Productization: A Study of Socially Enabled Business Exchanges in Open Source Service Networks." Information Systems Research, Vol. 19, n°4: p. 475-493.

Fitz-Gerald, Stuart. (2010) "K. Vadera, B. Gandhi, Open Source Technology (2009) University Science Press, Laxmi Publications, New Delhi, pp. 173." International Journal of Information Management, Vol. 30, n°4: p. 374.

Fitzgerald, Brian. (1997) "The use of systems development methodologies in practice: a field study." Information Systems Journal, Vol. 7, n°3: p. 201-212.

—. (2006) "The Transformation of Open Source Software." MIS Quarterly, Vol. 30, n°3: p. 587-598.

Franke, Nikolaus, Martin Schreier, et Ulrike Kaiser. (2010) "The "I Designed It Myself" Effect in Mass Customization." Management Science, Vol. 56, n°1: p. 125-140.

Franke, Nikolaus et Eric von Hippel. (2003) "Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software." Research Policy, Vol. 32: p. 1199-1215.

Free Software Foundation. (2007) "Announcing ncurses 5.6."

http://www.gnu.org/software/ncurses/ncurses.html

Fuggetta, Alfonso. (2003) "Controversy Corner: Open source software an evaluation." The Journal of Systems and Software, Vol., n°66: p. 77-90.

Gallego, M. Dolores, Paula Luna, et Salvador Bueno. (2008) "Designing a forecasting analysis to understand the diffusion of open source software in the year 2010." Technological Forecasting and Social Change, Vol. 75, n°5: p. 672-686.

Garcia, Rosanna et Roger Calantone. (2002) "A critical look at technological innovation typology and innovativeness terminology: a literature review." Journal of Product Innovation Management, Vol. 19: p. 110-132.

Gary, Kevin, Andinet Enquobahrie, Luis Ibanez, Patrick Cheng, Ziv Yaniv, Kevin Cleary, Shylaja Kokoori, Benjamin Muffih, et John Heidenreich. (2011) "Agile methods for open source safety-critical software." Software: Practice and Experience, Vol. 41, n°9: p. 945-962.

Glaser, Barney G. et Anselm L. Strauss. (1967) The discovery of grounded theory: strategies for qualitative research: Transaction Publishers.

Haefliger, Stefan, Georg von Krogh, et Sebastian Spaeth. (2008) "Code Reuse in Open Source Software." Management Science, Vol. 54, n°1: p. 180-193

Hamerly, Jim, Tom Paquin, et Susan Walton. (1999) "Freeing the Source: The Story of Mozilla." Pp. 197-206 in Open Sources: Voices from the Open Source Revolution, edited by C. DiBona, S. Ockman, and M. Stone. Sebastapol, CA: O'Reilly.

Hars, Alexander et Shaosong Ou. (2002) "Working for Free? Motivations for Participating in Open-Source Projects." International Journal of Electronic Commerce, Vol. 6., n°3: p. 25-39.

Heredia Alvaro, Jose Antonio et Minna Pikkarainen. (2011) "The art of innovation incubation." Pp. 104-122 in The Art of Software Innovation - Eight Practice Areas to Inspire your Business, edited by M. Pikkarainen, W. Codenie, N. Boucart, and J. A. Heredia Alvaro. Berlin Heidelberg: Springer.

Hicks, Christian et Dessislava Pachamanova. (2007) "Back-propagation of user innovations: The open source compatibility edge." Business Horizons, Vol. 50: p. 315–324.

Irani, Zahir et Peter E. D. Love. (2001) "Information systems evaluation: past, present and future." European Journal of Information Systems, Vol. 10, n°4: p. 183-188.

Janamanchi, Balaji, Evangelos Katsamakas, Wullianallur Raghupathi, et Wei Gao. (2009) "The State and Profile of Open Source Software Projects in health and medical informatics." International Journal of Medical Informatics, Vol. 78, n°7: p. 457-472.

Jeppesen, Lars Bo et Lars Frederiksen. (2006) "Why Do Users Contribute to Firm-Hosted User Communities? The Case of Computer-Controlled Music Instruments." Organization Science, Vol., n°17: p. 45-63.

Klincewicz, Krzysztof. (2005) "Innovativeness of open source software projects." Pp. 11-31: School of Innovation Management, Tokyo Institute of Technology.

Kogut, Bruce et Anca Metiu. (2000) "The Emergence of E-Innovation: Insights from Open Source Software Development." Philadelphia: Wharton School, University of Pennsylvania.

—. (2001) "Open-Source Software Development and Distributed Innovation." Oxford Review of Economic Policy, Vol. 17, n°2: p. 248-263.

Kozinets, Robert V. (2002) "The Field Behind the Screen: Using Netnography for Marketing Research in Online Communities." Journal of Marketing Research, Vol. 39, n°1: p. 61-72.

—. (2006) "Click to Connect: Netnography and Tribal Advertising." Journal of Advertising Research, Vol. 46, n°3: p. 279-288.

Lakhani, Karim et Eric von Hippel. (2003) "How open source software works: "free" user-to-user assistance." Research Policy, Vol. 32, n°6: p. 923-943.

Le Masson, Pascal , Benoit Weil, et Armand Hatchuel. (2010) Strategic Management of Innovation and Design. New York: Cambridge University Press.

Le Texier, Thomas et David W. Versailles. (2009) "Open Source Software Governance Serving Technological Agility: The Case of Open Source Software within the DoD." International Journal of Open Source Software and Processes (IJOSSP), Vol. 1, n°2: p. 14-27.

Lee, Gwendolyn K. et Robert E. Cole. (2003) "From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development." Organization Science, Vol. 14, n°6: p. 633-649.

Lee, Mae Lyn et Joseph Davis. (2003) "Evolution of Open Source Software: A Study on the Samba Project." Système d'Information et Management, Vol. 8, n°1: p. 43-62.

Lee, Sang-Yong Tom, Hee-Woong Kim, et Sumeet Gupta. (2009) "Measuring open source software success." Omega, Vol. 37, n°2: p. 426-438.

Lindman, Juho, Matti Rossi, et Anna Puustell. (2011) "Matching Open Source Software Licenses with Corresponding Business Models." IEEE Software, Vol. 28, n°4: p. 31-35.

Lippoldt, Douglas et Piotr Stryszowski. (2009) Innovation in the Software Sector: OECD Publications.
Lisein, Olivier, François Pichault, et James Desmecht. (2009) "Les business models des sociétés de services actives dans le secteur Open Source." Systèmes d'Information et Management, Vol. 14, n°2: p. 7-38.
Livari, Netta. (2010) "Discursive construction of 'user innovations' in the open source software development context." Information and Organization, Vol. 20, n°2: p. 111-132.
Lundell, Björn, Brian Lings, et Edvin Lindqvist. (2010) "Open source in Swedish companies: where are we?" Information Systems Journal, Vol. 20, n°6: p. 519-535.
MacCormack, Alan, John Rusnak, et Carliss Y. Baldwin. (2006) "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code." Management Science, Vol. 52, n°7: p. 1015-1030.
Midha, Vishal et Prashant Palvia. (2012) "Factors affecting the success of Open Source Software." Journal of Systems and Software, Vol. 85, n°4: p. 895–905.
Miles, Matthew B. et A. Michael Huberman. (1994) Qualitative Data Analysis: SAGE Publications, Inc.
Mingers, John. (2000) "Variety is the spice of life: combining soft and hard OR/MS methods." International Transactions in Operational Research, Vol. 7, n°6: p. 673-691.
—. (2003) "The paucity of multimethod research: a review of the information systems literature." Information Systems Journal, Vol. 13, n°3: p. 233-249.
Mingers, John et John Brocklesby. (1997) "Multimethodology: Towards a framework for mixing methodologies." Omega, Vol. 25, n°5: p. 489-509.
MinGW Team. (2008) "Minimalist GNU for Windows"

http://www.mingw.org/
Miralles, Francesc, Sandra Sieber, et Josep Valor. (2006) "An Exploratory Framework for Assessing Open Source Software Adoption." Système d'Information et Management, Vol. 11, n°1: p. 85-103.
Nambisan, Satish , Ritu Agarwal, et Mohan Tanniru. (1999) "Organizational mechanisms for enhancing user innovation in information technology." MIS Quarterly, Vol. 23, n°3: p. 365-395.
Nuvolari, Alessandro. (2004) "Collective invention during the British Industrial Revolution: the case of the Cornish pumping engine." Cambridge Journal of Economics, Vol. 28, n°3: p. 347-363.
OECD/Eurostat, Luxembourg. (2005) Oslo Manual: Guidelines for Collecting and Interpreting Innovation Data, 3rd Edition, The Measurement of Scientific and Technological Activities: OECD Publishing.
Okoli, Chitu et Suzanne D. Pawlowski. (2004) "The Delphi method as a research tool: an example, design considerations and applications." Information & Management, Vol. 42, n°1: p. 15-29.
Osterloh, Margit et Sandra Rota. (2007) "Open source software development: Just another case of collective invention?" Research Policy, Vol. 36: p. 157-171.
Paulson, James W., Giancarlo Succi, et Armin Eberlein. (2004) "An Empirical Study of Open-Source and Closed-Source Software Products." IEEE Trans. Softw. Eng., Vol. 30, n°4: p. 246-256.
Pinsonneault, Alain et Kenneth L. Kraemer. (1993) "Survey Research Methodology in Management Information Systems: An Assessment." Journal of Management Information Systems, Vol. 10, n°2: p. 75-105.
Rietveld, Toni et Roeland Van Hout. (1993) Statistical Techniques for the Study of Language and Language Behaviour: De Gruyter.
Rossi, Cristina. (2009) "Software innovativeness. A comparison between proprietary and Free/Open Source solutions offered by Italian SMEs." R&D Management, Vol. 39, n°2: p. 153-169.
Rowe, Gene et George Wright. (1999) "The Delphi technique as a forecasting tool: issues and analysis." International Journal of Forecasting, Vol. 15: p. 353–375.
Rowe, Gene et George Wright. (1996) "The impact of task characteristics on the performance of structured group forecasting techniques." International Journal of Forecasting, Vol. 12, n°1: p. 73-89.
Shah, Sonali K. (2006) "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development." Management Science, Vol. 52, n°7: p. 1000-1014.

Sigfridsson, Anders et Anne Sheehan. (2011) "On qualitative methodologies and dispersed communities: Reflections on the process of investigating an open source community." Information and Software Technology, Vol. 53, n°9: p. 981-993.

Soens, Wim. (2011) "The Art of Idea Harvesting." Pp. 33-48 in The Art of Software Innovation - Eight Practice Areas to Inspire your Business, edited by M. Pikkarainen, W. Codenie, N. Boucart, and J. A. Heredia Alvaro. Berlin Heidelberg: Springer.

Spaeth, Sebastian, Matthias Stuermer, et Georg von Krogh. (2010) " Enabling Knowledge Creation Through Outsiders: Towards a Push Model of Open Innovation." International Journal of Technology Management, Vol. 52, n°3/4: p. 411-431.

Spiller, Dorit et Thorsten Wichmann. (2002) "FLOSS Final Report - Part 3: Basics of Open Source Software Markets and Business Models." Berlecon Research GmbH.

Spinellis, Diomidis, Georgios Gousios, Vassilios Karakoidas, Panagiotis Louridas, Paul J. Adams, Ioannis Samoladas, et Ioannis Stamelos. (2009) "Evaluating the Quality of Open Source Software." Electronic Notes in Theoretical Computer Science, Vol. 233: p. 5-28.

Stallman, Richard. (1983) "Free Unix"

http://www.gnu.org/gnu/initial-announcement.html

—. (2002) "Obstructing Custom Adaptation of Programs." Pp. 124-125 in Free Software, Free Society: Selected Essays of Richard M. Stallman, edited by J. Gay. Boston: Gnu Press.

Stewart, Katherine J. et Sanjay Gosain. (2006) "The impact of ideology on effectiveness in open source software development teams." MIS Quarterly, Vol. 30, n°2: p. 291-314.

Stuermer, Matthias. (2009) "How Firms Make Friends: Communities in Private-Collective Innovation, Doctoral Dissertation." Chair of Strategic Management and Innovation, Department of Management, Technology, and Economics, ETH Zürich, Zürich.

Stuermer, Matthias, Spaeth Sebastian, et Georg von Krogh. (2009) "Extending private-collective innovation: a case study." R&D Management, Vol. 39, n°2: p. 170-191.

Taibi, Davide, Luigi Lavazza, et Sandro Morasca. (2007) "OpenBQR: a framework for the assessment of OSS." Pp. 173-186 in Open Source Development, Adoption and Innovation, vol. 234, Open Source Development, Adoption and Innovation, edited by IFIP. Boston: Springer.

Thakur, Dhanaraj. (2012) "A limited revolution — The distributional consequences of Open Source Software in North America." Technological Forecasting and Social Change, Vol. 79, n°2: p. 244-251.

The Economist. (2006) "Open-source business. Open, but not as usual" March 16th

Ulhoi, John P. (2004) "Open source development: a hybrid in innovation and management theory." Management Decision, Vol. 42, n°9: p. 1095-1114.

Veryzer Jr., Robert W. (1998) "Key factors affecting customer evaluation of discontinuous new products." Journal of Product Innovation Management, Vol. 15, n°2: p. 136-150.

Von Hippel, Eric. (1988) The Sources of Innovation. New York: Oxford University Press.

—. (2002) "Open source projects as horizontal innovation networks by and for users." Pp. 1-26: MIT.

—. (2005) Democratizing Innovation. Cambridge: MIT Press.

—. (2010) "Chapter 9 - Open User Innovation." Pp. 411-427 in Handbook of the Economics of Innovation, vol. 1, edited by B. H. Hall and N. Rosenberg: North-Holland.

Von Hippel, Eric et Georg Von Krogh. (2003) "Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science." Organization Science, Vol. 14, n°2: p. 209-223.

Von Krogh, Georg et Sebastian Spaeth. (2007) "The open source software phenomenon: Characteristics that promote research." Journal of Strategic Information Systems, Vol. 16, n°3: p. 236-253.

Von Krogh, Georg, Sebastian Spaeth, et Karim Lakhani. (2003) "Community, Joining, and Specialization in Open Source Software Innovation: a case study." Research Policy, Vol. 32, n°7: p. 1217-1241.

West, Joel et Siobhán O'Mahony. (2005) "Contrasting Community Building in Sponsored and Community Founded, Open Source Projects." Pp. 1-10 in Proceedings of the 38th Annual Hawaii International Conference on System Sciences. Waikoloa, Hawaii IEEE.