LOUVAIN
School of Management

UNIVERSITE CATHOLIQUE DE LOUVAIN

LOUVAIN SCHOOL OF MANAGEMENT

QUEL INTERET POUR LES ENTREPRISES DE CHOISIR LA SOLUTION OPEN SOURCE ?

Promoteur : Mathieu Zen
Co-promoteur : Jean Vanderdonckt

Mémoire-recherche présenté par Martin Vermeiren
en vue de l'obtention du titre de Master 120 crédits
en sciences de gestion

ANNEE ACADEMIQUE 2015-2016

Tout d'abord, je remercie le promoteur de ce mémoire, Mr. Mathieu Zen, pour son soutien et ses précieux conseils. Il m'a permis de relever ce challenge de la meilleure façon qu'il soit et m'a aiguillé tout au long de celui-ci. Je remercie également le co-promoteur, le professeur Jean Vanderdonckt, sans qui la réalisation de ce mémoire n'aurait pas été possible.

Mes remerciements vont aussi à tous les intervenants qui ont accepté de prendre le temps de répondre à mes questions malgré des agendas très chargés

Mes pensées vont aussi à mes parents qui m'ont toujours soutenu depuis si longtemps et ont fait de leur mieux pour que je puisse m'épanouir et devenir la personne que je suis aujourd'hui. Je les remercie également pour leur soutien et leurs relectures tout au long de ce mémoire.

Je ne peux oublier de dire merci à mes meilleurs amis qui ont fait de ces années universitaires une aventure inoubliable, et surtout certains ayant eu un rôle structurant dans ma vie.

Table des matières

Introduction

Dans un monde toujours plus individualiste où l'enrichissement personnel a toujours fait office de Saint Graal et où seul l'argent semble dicter les lois, les mouvements collaboratifs peuvent apparaître comme des ovnis ou de très belles bouffées d'oxygène

Depuis bientôt vingt ans, l'*open source*, occupe une place de plus en plus importante dans le monde de l'informatique et des entreprises. Bien que ses origines remontent à plus loin, il a commencé à prendre de l'ampleur fin des années 1990 et ne cesse d'augmenter sa présence depuis. Et peu à peu, cela quitte le cadre des logiciels pour toucher de nouvelles technologies et de nouvelles économies. Mais comment expliquer l'essor d'un tel mouvement?

Les valeurs sociales, de partage et d'entre-aide défendues par l'*open source* sont d'autant plus de raison pour s'y intéresser et même y adhérer. Néanmoins, une question plus pragmatique ne cesse de se pointer dans le fond de cette toile: comment font-ils pour générer du revenu? Malgré de très fortes valeurs humaines, il semble improbable que cela soit suffisant pour expliquer le succès toujours grandissant de l'*open source* : à un moment, cela doit être rentable.

D'ailleurs, prenons l'exemple belge de Odoo, anciennement OpenERP, qui affiche d'excellents résultats en proposant des solutions *open source*. Mais est-ce que cela peut s'appliquer à tout le monde ou seulement dans certaines situations ? Comment se fait-il aussi que l'*open source* ne soit pas plus connu du grand public ?

Face à l'inconnue de la pérennité du modèle *open source*, on peut se poser la question cruciale de l'intérêt pour une entreprise d'utiliser des logiciels *open source* que celle-ci les développe ou les utilise. C'est pour répondre à ce point, en se plaçant dans le rôle d'un consultant et non

d'un expert informatique, que ce projet vit le jour. Ce mémoire vise à aider toute personne, qu'elle soit experte ou non en technologie de l'information, à comprendre en quoi le choix d'un logiciel *open source* serait une bonne chose, ou non, pour le développement des activités de leur entreprise.

Pour ce faire, ce mémoire sera divisé en deux parties. La première partie, l'état de l'art, permettra d'avoir une vue théorique sur la question en se basant sur les travaux et articles réalisés à ce jour. Cela permettra dans un premier temps de comprendre l'évolution de l'*open source* en retraçant son histoire. Ensuite, les définitions, les avantages et les désavantages seront abordés de façon à comprendre clairement ce dont on traite. Les licences, qui ont une place particulièrement importante pour comprendre comment il est possible de développer un modèle commercial sur des logiciels libres, seront expliquées juste avant les différents business modèles existants autour de l'*open source*. Cette première partie s'achèvera avec une brève revue des extensions de l'*open source*.

La deuxième partie de ce travail constituera l'analyse empirique. Une série d'interviews a été réalisée auprès d'acteurs du milieu. D'un côté, des avis d'entreprises qui développent des solutions *open source*, utilisent de l'*open source* pour développer leurs services, ou qui développent des solutions propriétaires et de l'autre, des entreprises utilisant des solutions *open source* pour leurs activités, ou des solutions propriétaires. Des consultants et chercheurs IT ont également été contactés. Cette diversité des intervenants permettra d'effectuer une analyse prenant en compte les différents points de vue.

Dans un premier temps, un lien avec la partie théorique sera effectué de manière à relever les similitudes et les différences. Dans un second temps, le point de vue des entreprises développant des logiciels sera abordé. Pour enfin terminer cette partie avec la position des entreprises utilisant des logiciels pour le bon fonctionnement de leurs activités mais dont ce n'est pas le business.

Finalement, la conclusion proposera un avis sur base de l'analyse réalisée.

A titre personnel, ce choix de sujet repose sur l'envie de découvrir une autre manière de développer un business et de fonctionner. En désaccord avec la façon de procéder de notre société où le "je" a souvent plus d'importance que le "nous", la découverte de cette communauté *open source* m'a donné envie d'en apprendre plus. Afin de voir s'il était possible de changer certaines manières de faire en se basant sur le mouvement *open source*, il m'a paru nécessaire de comprendre en quoi celui-ci pouvait être intéressant pour les entreprises.

De plus, savoir que certaines entreprises arrivent à en vivre est encourageant pour le futur et laisse la place à la possibilité de réellement changer les autres. Car si l'argent est surement une condition, pourquoi ne pas créer une société plus collaborative ? Comme vous le verrez dans les prochaines pages, l'*open source* est un beau moyen pour favoriser l'innovation. Si c'est le cas avec les logiciels informatiques, pourquoi pas l'étendre à d'autres choses. Mr. James Whitehurst, CEO de Red Hat, tente de faire peu à peu passer le message dans son livre *The Open Organization* évoquant notamment une raison fort simple : plusieurs cerveaux sont meilleurs qu'un seul cerveau. Et comme le dit si bien notre devise nationale, *l'union fait la force*.

Partie 1. Etat de l'art

Dans cette première partie, la littérature touchant le mouvement *open source* sera traitée. Différents points seront abordés comme une explication de ce qu'est l'*open source*, ainsi que les aspects légaux et économiques. Et nous terminerons par une analyse de ce que le futur réserve à ces logiciels et cette façon de procéder.

1. Qu'est-ce que l'*open source*

Ce chapitre a pour but d'expliquer ce qu'est l'*open source*. Pour ce faire, il est important d'en retracer l'histoire dans ses grandes lignes pour en comprendre ses origines et son évolution. Ensuite, nous aborderons les définitions entourant l'*open source*, avant de s'attaquer aux forces et faiblesses de ces logiciels. Nous terminerons par un regard sur la communauté qui compose ce mouvement.

1.1. Un peu d'histoire

Dans le courant des années 1950-1960, lorsqu'un système d'exploitation d'ordinateur était acheté, le code source permettant de corriger des bogues ou d'ajouter de nouvelles fonctionnalités était généralement inclus dans le *pack*. Ces nouvelles technologies étaient principalement utilisées par les universités qui exigeaient un accès complet à toutes les composantes du logiciel. (Ceruzzi, 2003).

Cependant, suite à l'augmentation des coûts des programmes, l'obligation de céder le code source avec le programme auquel il se rapporte fût décrétée comme concurrence déloyale par le gouvernement américain et bon nombre de logiciels commencèrent à être vendus sous forme de licence. (Fisher, McKie & Mancke, 1983). Il faudra attendre le

début des années 1980 et Richard Stallman pour assister à la naissance des logiciels libres.

1.1.1. Richard Stallman et les logiciels libres

Tout commença de façon anodine, Mr. Stallman, travaillant alors pour le laboratoire d'intelligence artificielle du MIT, voulait réparer une imprimante récalcitrante mais se voit refuser l'accès au code source à cause d'un accord de non divulgation. Ajouté à ça un changement d'ordinateurs au sein du MIT avec un accès limité aux codes, il n'en fallut pas plus pour que Richard Stallman décide de changer la situation et que le mouvement des logiciels libres voie le jour. (Bretthauer, 2002).

En 1984, après avoir démissionné du MIT afin que celui-ci n'interfère pas avec ses projets, Stallman commença à écrire un nouveau système d'exploitation comprenant tous les outils nécessaires à son bon fonctionnement. Il décida de le rendre compatible à *UNIX*, un système d'exploitation déjà disponible à l'époque, pour que les programmeurs ne doivent pas apprendre un nouveau langage. Le projet GNU *(GNU's Not UNIX)* était lancé. Peu après, en octobre 1985, Stallman fonde la *Free Software Foundation (FSF)* afin de soutenir le développement du projet GNU et d'engager de nouveaux programmeurs pour faire face à son succès grandissant (Bretthauer, 2002).

En 1991, tout le système d'exploitation était écrit et il ne manquait plus que le *kernel*, ou le noyeau, la partie permettant de lier tout le système, pour que GNU puisse fonctionner correctement. C'est à ce moment-ci qu'apparurent Linus Torvald et Linux.

1.1.2. Linus Torvalds et Linux

Le 25 août 1991, Linus Torvalds, un jeune étudiant finlandais de 21 ans, dévoile le *Linux kernel*, un nouveau système d'exploitation qu'il décida de monter seul par passion pour l'informatique. En 1992, celui-ci sera licencié sous GNU permettant ainsi

de distribuer un nouvel OS[1] complet qui sera depuis connu sous le nom de *Linux*. A tort ou à raison, Richard Stallman débattra de l'appellation du système d'exploitation qui selon lui devrait se nommer *GNU-Linux*. Ceci marque un point important de l'histoire des logiciels libres. Dorénavant, les programmeurs peuvent utiliser un OS avec un accès total au code source. Celui-ci devient d'ailleurs une alternative sérieuse pour de nombreuses sociétés à partir de 1996.

1998 représente également une année charnière pour *Linux*. De nombreuses sociétés de premier rang comme IBM, Oracle ou Compaq le soutiennent publiquement. C'est également l'année de publication de l'essai "La cathédrale et le Bazar", de Eric Raymond, traitant des logiciels libres, suite à quoi Netscape décida de publier le code source de son navigateur internet. Ceci permit à Linux de jouir d'une visibilité bien plus importante.

1.1.3. L'*open source* Initiative

Suite à la décision de Netscape de rendre son code source accessible, de nombreux acteurs se penchèrent sur la possibilité de rendre les logiciels libres plus attrayants pour des sociétés devant être rentables. Le 3 février 1998, un groupe d'individus discuta la question lors d'une session de stratégie à Palo Alto en Californie. Au cours de cette séance, Christine Peterson suggéra le terme "*open source*" : le but est de mettre en avant le partage des codes sources et non les valeurs sociales et la quête de liberté des utilisateurs cherchée par la FSF ainsi que ses logiciels libres. La suggestion fut acceptée et à la fin du mois, Eric Raymond et Bruce Perens fondèrent l'*Open Source Initiative (OSI)*. Cette organisation vise à promouvoir le partage des codes sources et la collaboration. La nouvelle appellation fut rapidement acceptée par des membres de la communauté comme Linus Torvalds. Elle fut également présentée lors d'un sommet sur les logiciels libres en avril 1998 où elle rencontra un franc succès. (Open Source Initiative, 2012).

[1] Operating System, traduction anglaise de "système d'exploitation".

La mission principale de l'OSI est d'expliquer et de protéger le label *open source*. Outre cela, l'organisation s'occupe également de distribuer les licences aux logiciels. Celles-ci sont accordées si les logiciels respectent les dix critères nécessaires à la qualification d'*open source* (voir 1.2.1.).

1.1.4. Les années 2000

Au cours des années 2000, le nombre de projets *open source* n'a cessé d'augmenter. La création de la plate-forme GitHub en 2009 a eu un impact considérable sur la multiplication de ceux-ci. Il y eut également la société RedHat qui est devenue l'entreprise référence en matière d'*open source*. Le navigateur web Firefox ou l'OS Android pour *smartphones* utilisent tous deux le kernel Linux.

De 2007 à 2015, nous pouvons observer une croissance exponentielle du nombre de projets *open source*. Selon une étude réalisée par *Black Duck Software* (North Bridge & Black Duck Software, 2015), le nombre de projets *open source* serait passé de 126.650 en 2007 à 1.500.000 en 2015.

1.1.5. Microsoft et l'*open source*

Microsoft était perçu comme l'ennemi des logiciels libres et de l'*open source*. Linux fut d'ailleurs qualifié de "cancer" ou "communisme de nouveau genre" par Steve Ballmer, ancien CEO de Microsoft. Pendant tout un temps, cette société s'opposa à toute forme de partage de ses codes sources. Néanmoins, au fur et à mesure que les années passèrent, Microsoft commença à intégrer petit à petit la notion d'*open source* et à l'utiliser. En 2012, MS Open Tech, une section entièrement dédiée à l'*open source*, fut lancée. Sadya Natella, déjà PDG à l'époque, déclara même que "Microsoft aime Linux". Son but étant d'améliorer la présence de l'*open source* dans les produits Microsoft. Ce fut vu comme un grand pas en avant pour l'*open source*. Désormais, cette section indépendante est rattachée à Microsoft et se nomme "*Microsoft Open Technology Programs Office*". Cette division se faisait un devoir de faciliter l'intégration d'autres

logiciels à ses produits et vice versa. Étant désormais actifs sur la plate-forme GitHub, les programmeurs de Microsoft participent à pas moins de 2000 projets *open source* différents. Et ils ne cessent de développer de nouveaux outils, programmes ou applications pour d'autres plates-formes. Comme ce fut le cas avec Office 365 pour la plate-forme éducative Moodle. Ce fut également un bon coup de pub pour l'image même de la société qui se positionnait dès lors comme une entreprise écoutant ses clients et voulant évoluer avec son temps (Clapaud, 2015).

Au jour d'aujourd'hui, il serait même possible de voir apparaître un Windows entièrement *open source* d'après Mark Russinovitch, le directeur technique de Microsoft Azure. Ceci montre l'impact grandissant de l'*open source* dans notre société ainsi que sa présence.

1.2. Définition et notions

1.2.1. *Open source*

D'après OSI (Open Source Initiative, 2007), un programme peut être qualifié d'*open source* s'il respecte dix critères :

1. Redistribution libre :

La licence ne doit pas empêcher quiconque de vendre ou de donner le logiciel en tant que composant d'une distribution de logiciels contenant des programmes provenant de plusieurs sources différentes. La licence ne doit pas exiger une redevance ou d'autres frais pour une telle vente.

2. Code source :

Le programme doit inclure le code source, et doit autoriser la distribution du code source, ainsi que sous forme compilée. Si une certaine forme d'un produit n'est pas distribuée avec le code source, il doit y avoir un moyen très médiatisé

d'obtenir le code source pour un coût raisonnable de reproduction, de préférence le téléchargement via l'Internet sans frais. Le code source doit être la forme préférée dans laquelle un programmeur modifie le programme. Un code source délibérément obscurci n'est pas autorisé. Des formes intermédiaires, comme la sortie d'un préprocesseur ou un traducteur ne sont pas autorisées.

3. Travaux dérivés :

La licence doit permettre les modifications et les travaux dérivés, et doit leur permettre d'être distribués dans les mêmes termes que la licence du logiciel original.

4. Intégrité du code source de l'auteur :

La licence peut restreindre le code source d'être distribué sous forme modifiée que si la licence permet la distribution de fichiers «*patch*» avec le code source dans le but de modifier le programme au moment de la construction. La licence doit explicitement permettre la distribution du logiciel intégré à partir du code source modifié. La licence peut exiger que des travaux dérivés portent un nom ou numéro de version différent du logiciel original.

5. Absence de discrimination envers des personnes ou des groupes :

La licence ne doit pas discriminer contre toute personne ou groupe de personnes.

6. Absence de discrimination envers des domaines d'activités :

La licence ne doit pas restreindre quiconque de faire usage du programme dans un domaine spécifique de l'entreprise.

7. Distribution de licence :

Les droits attachés au programme doivent s'appliquer à tous ceux à qui le programme est redistribué sans qu'il soit nécessaire pour l'exécution d'une licence supplémentaire par ces parties.

8. La licence ne doit pas être spécifique à un produit :

Les droits attachés au programme ne doivent pas dépendre d'une partie du programme d'une distribution de logiciel particulier. Si le programme est extrait de cette distribution et utilisé ou distribué dans les termes de la licence du programme, toutes les parties à qui le programme est redistribué doivent avoir les mêmes droits que ceux qui sont accordés avec la distribution du logiciel original.

9. La licence ne doit pas imposer de restrictions sur d'autres logiciels :

La licence ne doit pas imposer des restrictions sur d'autres logiciels distribués avec le logiciel sous licence. Par exemple, la licence ne doit pas exiger que tous les autres programmes distribués sur le même support doivent être des logiciels *open source*.

10. La licence doit être technologiquement neutre :

Aucune disposition de la licence ne peut être fondée sur une technologie ou d'un style d'interface individuelle.

1.2.2. Logiciel libre

Selon GNU (GNU Operating System, 2016), le terme "logiciel libre" désigne des logiciels qui respectent la liberté des utilisateurs. Ceux-ci ont la liberté d'exécuter, copier, distribuer, étudier, modifier et améliorer ces logiciels. "Logiciel libre" [*free software*] fait référence à la liberté et non au prix.

Un programme est un logiciel libre si vous, en tant qu'utilisateur de ce programme, avez les quatre libertés essentielles :

Liberté 0 :
La liberté d'exécuter le programme comme vous voulez, pour n'importe quel usage.

Liberté 1 :
La liberté d'étudier le fonctionnement du programme et de le modifier pour qu'il effectue vos tâches informatiques comme vous le souhaitez; l'accès au code source est une condition nécessaire.

Liberté 2 :
La liberté de redistribuer des copies, donc d'aider votre voisin.

Liberté 3 :
La liberté de distribuer aux autres des copies de vos versions modifiées; en faisant cela, vous donnez à toute la communauté une possibilité de profiter de vos changements; l'accès au code source est une condition nécessaire.

1.2.3. Débat légitime ?

Tout d'abord, il est important de préciser que les deux mouvements sont nés d'une même volonté, partagent leur passé et une même culture. Mais ils diffèrent,sur les valeurs qu'ils mettent en avant. Si d'un côté les logiciels libres et Richard Stallman mettent une importance capitale sur la liberté des utilisateurs, le mouvement *open source* veut mettre en avant les intérêts de la collaboration.

En effet, comme cité plus haut, le mouvement *open source* est né d'un désir de se dissocier de l'image "free" pour pouvoir être plus attrayant d'un point de vue commercial et stratégique. Cela n'enlève en rien son passé commun avec les logiciels libres. De nombreuses personnes partageaient ce désir de rendre les codes sources publics et utilisables mais n'adhéraient pas forcément avec toutes les valeurs sociales mises en avant par les partisans des logiciels libres. Ils abordaient le problème d'une

façon pragmatique et étaient d'accord que l'*open source* permettait de développer des programmes et logiciels plus performants et mieux construits. Ce qui est une des critiques principales de Richard Stallman à l'égard de l'OSI (Stallman, 2016).

D'un point de vue pratique, les critères de l'*open source* sont moins restrictifs que ceux des logiciels libres. Ainsi, la grande majorité des logiciels *open source* sont des logiciels libres mais l'inverse n'est pas forcément vrai. Etant plus large que les logiciels libres, il se peut que des programmes tivoisés[2] utilisant Linux soit *open source* alors qu'ils sont non libres (Stallman, 2016).

Toutefois, les deux mouvements s'accordent à dire qu'ils ne sont pas en guerre l'un contre l'autre et que leurs objectifs sont semblables. L'un comme l'autre tente de combattre les logiciels propriétaires. Le débat entre *open source* et logiciel libre est pertinent au sein de la communauté informatique et des programmeurs, les experts et connaisseurs aimant à mettre les mots justes sur les différentes choses. Cependant, il n'est pas attendu que tout un chacun soit à même de définir chaque mouvement avec précision et de pouvoir en citer les différences. Au vu du succès grandissant du terme *open source* et de sa popularité, ainsi que de l'objet de ce mémoire qui est tourné vers les intérêts pour les entreprises et non le côté de la question, celui-ci sera utilisé pour la suite de ce mémoire.

1.3. Forces et faiblesses

Lorsque l'on parle des avantages et inconvénients des logiciels *open source*, cela est bien souvent fait en comparaison avec les logiciels propriétaires. Ceci peut se transformer en débat sans fin entre les partisans de chaque mouvement. Néanmoins, il est possible de trouver des points sur lesquels les deux groupes semblent s'accorder. Afin de rester

[2] Un programme ou appareil électronique basé sur des logiciels libres mais qui utilise un système de protection qui empêche l'utilisation de version modifiée (ex : GPS TomTom, Freebox, etc.)

concis et de ne pas trop s'écarter du chemin choisi, les forces et faiblesses des OSS[3] seront abordés dans le cadre des entreprises.

1.3.1. Forces

- Transparence

L'accès au code source étant accordé, il n'y a pas d'inconnue quant à ce que fait le logiciel. L'utilisateur a connaissance de toutes les fonctionnalités, contrairement à un logiciel propriétaire où de nombreuses fonctions sont inconnues de l'utilisateur (Fitzgerald, 2006).

- Sécurité

Grâce à la transparence, l'entreprise a un contrôle total sur ce qui se passe au sein du logiciel. Elle sait également ce qu'il advient de toutes ses données, ce qui n'est peut-être pas le cas avec un logiciel propriétaire où il peut y avoir des fuites d'informations qui remontent jusqu'au développeur du logiciel (Stallman, 2001). De plus, d'après une étude réalisée par Synopsys, pour mille lignes de codes, la densité de défaut des OSS est de 0,61 contre 0,76 pour les logiciels propriétaires (Synopsys, 2015).

- Coût

Il est moins cher pour une entreprise de se procurer un OSS qu'un logiciel propriétaire. Ceux-ci sont parfois gratuits (Lerner & Tirole, 2002).

- Innovation

Grâce à la communauté *open source* et au nombre de cerveaux travaillant sur les mêmes projets, il y a plus de chance de déboucher sur de nouvelles innovations (Goldman & Gabrile, 2005). On peut parler de développement par le bazar : de nouvelles idées sont constamment soumises, les meilleures sont gardées et les moins bonnes rejetées, ce qui crée de nouveaux logiciels ou applications plus performants et innovants. Contrairement

[3] Open Source Software, traduction anglaise de "logiciels open source"

au développement par la cathédrale où les développeurs peuvent être contraints de se focaliser sur une certaine direction sur demande de leur supérieur (Raymond, 1999).

- Indépendance

Comme il est mentionné ci-dessus, l'entreprise a un contrôle total sur le logiciel. L'accès au code source lui permet de réaliser toutes les modifications et ajouts nécessaires à l'usage dont elle a besoin. Elle a la possibilité d'en faire un logiciel sur mesure (Stallman, 2001).

- Communauté

Le nombre de membres permet une meilleure révision des logiciels comparée aux logiciels propriétaires. La correction des bogues est faite plus rapidement, ce qui a un impact positif sur la vitesse d'exécution du programme (Optimus Information, 2015).

1.3.2. Faiblesses

- Licences

Les critères d'attribution et les différentes licences ne sont pas toujours très clairs et souvent restent inconnus ou incompris par les utilisateurs. Cela peut entraîner des problèmes de droits d'auteurs (Stallman, 2001).

- Support

La majorité des programmeurs et développeurs prenant part à des projets OSS font ça de leur plein gré ce qui crée des problèmes au niveau du suivi des logiciels. Cela peut prendre du temps pour trouver les réponses à certaines questions : les personnes ne se sentant plus concernées par le projet, forum noyé de posts, trouver les informations de contacts des personnes qualifiées,... (Optimus Information, 2015).

- Coût

Bien que le coût d'acquisition d'un OSS soit moins important que celui d'un logiciel propriétaire, dans certains cas il est plus avantageux d'opter pour le second. Si l'OSS a

besoin de modifications pour répondre aux besoins de l'entreprise, ça peut devenir coûteux pour celle-ci. En effet, le temps consacré à l'adaptation, la formation des employés ou la recherche de programmeurs qualifiés peuvent au final s'avérer moins intéressant que de faire appel à un logiciel commercial dès le début. (Lerner & Tirole, 2002).

- Complexité

L'utilisation et la modification de certains OSS ne sont pas données aux premiers venus, certaines nécessitent des compétences et connaissances informatiques pointues. L'entreprise peut se retrouver dans une situation où elle ne possède pas les atouts nécessaires, ce qui peut se traduire par des coûts supplémentaires comme cité ci-dessus (Optimus Information, 2015).

1.4. La communauté *open source*

La communauté de l'*open source* est très certainement sa plus grande force. En effet, contrairement à des logiciels propriétaires, les projets *open source* ne peuvent se développer que si des personnes acceptent d'y contribuer leur temps et partager leurs connaissances (Open Source Initiative, 2016). L'un des gages de succès des projets *open source* est d'en augmenter la conscience au sein de la communauté. L'OSI en fait un de ses objectfs (Open Source Initiative, 2016). Le mouvement propose de nombreux moyens pour rentrer en contact avec la communauté comme via la plate-forme GitHub ou le *live chat* Freenode où il est facile de se connecter les uns aux autres aux moyens de forums, profils et messages, et de trouver des personnes qualifiées.

Transparence et vie privée sont deux maîtres mots de la communauté *open source* (Phipps, 2012). Tous les participants sont là de leur bon vouloir, ou celui de leur employeur, et pas forcément pour les mêmes raisons. Certains y sont par philanthropie, d'autres par intérêt personnel. C'est pourquoi il est important de respecter la vie privée de ceux-ci ainsi que d'être transparent dans toutes ses démarches pour le bon fonctionnement d'un projet. Il y a des normes à respecter et celles-ci visent

principalement l'égalité de chaque participant. A partir du moment où le projet est *open source*, que vous soyez le sponsor de celui-ci ou un co-développeur, vous jouissez des mêmes droits et n'avez pas de passe-droit par rapport aux autres. Peu importe si votre contribution fut plus importante (Phipps, 2012). D'ailleurs, si en tant que société vous souhaitez démarrer une communauté *open source* tout en gardant l'exclusivité des projets, les résultats vont probablement être négatifs (West & O'mahony, 2008).

2. Licences

Ce chapitre traite des différents types de licences qui existent pour protéger les logiciels. Le problème des fraudes et du plagiat est également couvert avec deux cas pratiques dont l'issue pourrait avoir un grand impact sur le futur des licences *open source*.

2.1. Copyleft

Le concept voit le jour en 1985, lorsque Richard Stallman présente GNU dans un manifesto. Le nom fait opposition au *copyright* restreignant la liberté des utilisateurs et est utilisé par les logiciels propriétaires.

Théoriquement, le meilleur moyen de rendre un logiciel libre est de le mettre dans le domaine public, sans y attacher de *copyright* donc. Le problème est que cette solution peut aller à l'encontre des libertés des utilisateurs énumérées dans la définition du logiciel libre. En effet, toute personne apportant des modifications au logiciel de base peut décider de protéger sa nouvelle version, enlevant ainsi les libertés octroyées par la version de base. C'est derrière ce raisonnement que le concept de *copyleft* apparait.

D'après GNU, le *copyleft* est une méthode générale pour rendre libre un programme (ou toute autre œuvre) et obliger toutes les versions modifiées ou étendues de ce programme à être libres également (GNU Operating System, 2015).

D'un point de vue pratique, pour protéger un logiciel avec un *copyleft*, il faut d'abord le placer sous *copyright*. Les conditions de distribution sont ajoutées dans un second temps. Elles font office d'outil juridique octroyant les libertés d'utilisation. Il est dès lors illégal de dissocier le code de ses libertés (GNU Operating System, 2015). Concrètement, l'auteur du logiciel remet ses droits sous *copyright*.

Il existe différents types de *copyleft*:

- *Copyleft* fort : il s'applique à tous les programmes et travaux dérivés du logiciel initial.
- *Copyleft* faible : il ne s'applique qu'au logiciel initial.
- *Copyleft* partiel : il empêche de modifier certaines parts.

Le concept *copyleft* a donné naissances aux licences FOSS[4]. GNU General Public licence (GNU-GPL) est la première licence *copyleft* à être utilisée et l'une des plus répandues.

2.2. Licences

Les licences ont pour objectif premier de refuser l'exclusivité d'un logiciel à un individu (Laurent, 2004). Elles peuvent être divisées en trois catégories (Lerner & Tirole, 2005):

- Très contraignante

  Les versions modifiées ne peuvent se joindre à des logiciels ne possédant pas la même licence et doivent être distribuées sous les mêmes termes. Ces licences sont parfois qualifiées de "virales" ou "réciproques".

- Contraignante

  Les versions modifiées doivent mettre le code source à disposition de tous et doivent être distribuées sous les mêmes termes.

- Permissive

  Les modifications apportées et leurs distributions peuvent être rendues propriétaires.

Chacune d'entre elles autorise les usages commerciaux et dérivés des logiciels. Cependant, en fonction des intérêts des développeurs, une licence peut être plus adaptée

[4] *Free and Open Source Software*

qu'une autre. Par exemple, un projet mené par une entreprise commerciale a plus de chance d'être mis sous licence permissive. Contrairement à un projet pour la communauté qui aura une licence restrictive (Lerner & Tirole, 2005).

A noter que la grande majorité des FOSS développés par des sociétés ont désormais une double licence, l'une est gratuite et l'autre, accompagnée d'un support technique, est payante (Deek & McHugh, 2007). Cela rejoint un désir de rester bancable.

2.2.1. Les licences les plus utilisées

Les annexes 1 et 2 montrent les licences les plus populaires ainsi que les catégories dans lesquelles elles se trouvent.

- **MIT License**

La licence MIT est l'une des licences les plus permissives et simples, tous les droits sont accordés et cela sans restrictions. Dans le cas où le travail est redistribué, il est uniquement demandé de placer le copyright et la notice donnant la permission de réutiliser et redistribuer le travail (Open Source Initiative, 2016). Elle permet notamment de réaliser des travaux dérivés ou de passer sous licence commerciale sans la moindre contrainte (Laurent, 2005).

- **BSD License**

Cette licence offre beaucoup de flexibilité, son unique contrainte étant la mention de l'Université de Californie (Lerner & Tirole, 2005). Cependant, il est nécessaire de demander au préalable l'autorisation du créateur pour utiliser son nom afin d'éviter toute mauvaise publicité pour celui-ci si le logiciel est de piètre qualité.

Contrairement à la licence MIT, il est obligatoire d'inclure les termes de la BSD dans les travaux dérivés. Tant que les termes sont respectés, les travaux dérivés peuvent être

réalisés sous une autre licence ou même en tant que programme propriétaire (Laurent, 2005).

- **Apache License**

Tout comme les licences MIT et BSD, la licence Apache v1.1. est permissive. Elle possède également une clause protégeant la réputation du créateur en cas de travail dérivé (Laurent, 2005).

La v2.0 est quant à elle plus complexe. En effet, elle définit de manière précise tous les différents termes de la licence et étend celle-ci à tous les documents dérivés permettant le bon usage et la compréhension du code source (Open Source Initiative, 2004). Il est également précisé que les travaux dérivés peuvent être réalisés sous une autre licence mais que toute contribution sera ajoutée au travail de base conformément aux normes de la v2.0 (Laurent, 2005).

- **GNU General Public License**

Le but de la licence GNU-GPL est d'assurer la libre distribution des logiciels ainsi que le libre accès au code source. Sur cette base, toute personne jouissant de ces droits, doit également les octroyer à son tour (Free Software Foundation, 2007).

Nous sommes donc dans le cas d'une licence restrictive, il n'y a pas la possibilité de réaliser des travaux dérivés sous une autre licence ou en propriétaire. Une fois qu'un code est sous licence GNU-GPL, il le restera et tout futur travail devra respecter les termes de la licence (Laurent, 2005).

- **Lesser GNU General Public License**

Cette licence a également été créée par la Free Software Foundation et fait office d'exception aux sections 3 et 4 de la licence GNU - GPL. En ce sens, les licences LGPL sont

donc autorisées à être développées avec des licences non GPL (Free Software Foundation, 2007).

- **Mozilla Public License**

La MPL est une licence créée par Netscape lors de sa transition vers l'*open source* en 1998. Elle assure le respect des codes sources existants tout en encourageant les futurs développeurs *open source* à y contribuer.

Depuis 2012, Mozilla utilise la v2.0 de la licence. Celle-ci est en quelque sorte une hybride entre la licence GPL et les licences BSD (Laurent, 2005).

2.3. Fraude et plagiat

Au vue de la complexité de toutes ces licences, de la non compréhension de celles-ci par de nombreux utilisateurs et du fait qu'elles n'ont pas encore été proprement testées en justice, des cas litigieux peuvent apparaître (Lerner & Tirole, 2005). Bien que sans conséquence fâcheuse la plupart du temps, il peut arriver que certaines situations prennent des proportions bien plus importantes.

Contrairement aux logiciels libres, les OSS sont généralement moins restrictifs quant à l'appellation. Dès lors, il est possible qu'un logiciel ait une licence *open source* uniquement pour certaines fonctionnalités et que le reste soit sous copyright normal. Cela peut entrainer de nombreuses confusions car ce n'est pas toujours mentionné de façon explicite mais implicite (Laurent, 2004).

Le problème peut aussi se passer dans l'autre sens. Lorsqu'une personne utilise un OSS, elle n'est pas tenue d'en notifier le concédant. Dès lors, celui-ci ne connait pas forcément les utilisateurs de son programme et peut difficilement savoir s'il y a eu certains *copyrights* qui ont été enfreints (Laurent, 2004).

Deux grosses affaires, Versata et Oracle v. Google, tentent d'apporter une première réponse à l'interprétation des licences par la justice.

2.3.1. Versata case

Versata, une entreprise développant des logiciels propriétaires, a décidé d'utiliser un utilitaire *open source* pour l'un des produits qu'elle propose. L'utilitaire développé par la société XimpleWare est licencié sous GPLv2 et autorise donc l'accès au code source et toutes les modifications aux utilisateurs. Il en existe également une licence commerciale mais celle-ci ne fut pas choisie par Versata.

Le problème survint lorsque Versata fournit son logiciel à Ameriprise, une entreprise proposant des services financiers, et que celle-ci autorisa un sous-traitant à décompiler le logiciel. A partir de ce moment, l'histoire se poursuit en deux phases.

1. Versata poursuit Ameriprise en justice.

Versata estime que son client a violé les droits de licences qui lui étaient octroyés. Celui-ci se défend en clamant que Versata est en tort et a violé le *copyleft* GPLv2 sur le droit d'accès au code source et le droit d'y apporter des modifications.

Le tribunal décréta que le *copyleft* GPLv2 avait droit sur le Copyright Act car celui-ci ne protège pas spécifiquement un utilisateur de recevoir le code source d'un logiciel (Jakob, 2015). Etant donné que Versata et Ameriprise ont finalement décidé de s'arranger entre eux, les juges n'ont pas eu à se prononcer quant à un jugement final.

2. XimpleWare poursuit Versata et ses clients en justice.

Selon le développeur *open source*, Versata a enfreint la licence GPL en ne la mentionnant pas dans son programme et Ameriprise l'a enfreint en fournissant le code source à des sous-traitant commerciaux.

Dans ce cas-ci, il n'y a pas d'accord entre les différentes parties. Les juges sont tenus d'adresser trois questions (Jakob, 2015) :

- La distribution du code source aux fournisseurs indépendants (qui vont, eux mêmes, changer, modifier et distribuer le code) est-elle couverte par la licence GPLv2 ?

- Quelles sont les conséquences pour les clients des parties ayant enfreint la licence, qui n'ont pas eux-mêmes distribué le code ?

- Est-ce que GPLv2 contient une licence de brevet ou seulement une licence *copyright*, ce qui voudrait dire que la simple utilisation du logiciel pourrait résulter en violation de brevet ?

Les juges ont répondu positivement à la première question car les politiques FSF et GPL ne différencient pas fournisseurs et tiers. Ils ont également décrété que la violation de la licence par une entité dans la chaîne de distribution ne concerne pas les clients, sous réserve que ceux-ci ne distribuent pas le code. La troisième question n'a pas encore été abordée, ni les implications et conséquences pour une entreprise jugée coupable.

2.3.2. Oracle vs. Google

En 2007, Google et Sun étaient en discussion pour implémenter des API[5] Java dans l'OS Android. Bien qu'il n'y ait pas eu d'accord, Google décida de quand même utiliser certains API de manière à ce que les *copyrights* ne soient pas violés. Cependant, après avoir acquis Sun en 2010, Oracle décida de poursuivre Google en justice pour violation de droits. Selon Oracle, l'utilisation des mêmes noms et titres par Google pour les API irait à l'encontre des *copyrights*.

[5] *Application Programming Interface*

Lors du procès en 2012, l'*U.S. District Court* décréta que même si des lignes de code avaient été copiées, il n'y avait pas eu d'infraction et que dans ce cas-ci, les API ne pouvaient être protégés par des *copyrights*. Cependant, en 2014, des juges de la *U.S. Federal Court of Appeals* renversèrent la décision précédente et jugèrent Google fautif. La société de Palo Alto fit appel auprès de la *U.S. Supreme Court*.

Le procès concernant l'appel de Google eut lieu du 9 au 26 mai 2016 et les jurys tranchèrent en faveur de Google en déclarant que la société n'était pas fautive. Néanmoins, Oracle a décidé de faire appel...

Si l'issue de ce procès révèle que les API peuvent être protégées par un *copyright*, cela aura un impact considérable sur le monde informatique. En effet, la force des API réside dans leur développement par la répétition des meilleures pratiques (Riggins, 2015). Si l'on en venait à les protéger, cela altérera considérablement la croissance de l'industrie des logiciels.

3. Business

Dans ce chapitre, nous aborderons les différentes facettes qui composent le business de la sphère *open source*. Dans un premier temps, nous explorerons les différentes composantes économiques qui accompagnent le développement d'un OSS. Nous toucherons ensuite aux différents modèles existants. Nous terminerons par un point concernant l'importance du marketing pour le succès d'un projet *open source*.

3.1. L'*open source* d'un point de vue économique

La question que tout le monde se pose lorsque l'on aborde l'*open source*, est de savoir pourquoi des personnes qui ne sont pas payées décident de travailler sur le développement de logiciels ? C'est sur base de cette question que l'économie dans l'*open source* sera abordée.

3.1.1. Les programmeurs

Les programmeurs sont les pierres fondatrices des logiciels, que ceux-ci soient *open source* ou commerciaux. C'est avant tout grâce à eux que les logiciels sont développés et il est donc important de savoir ce qui les motive.
Selon Lerner & Tirole (2000), la motivation se traduit par :

Bénéfice net = bénéfice immédiat + bénéfice retardé

Le bénéfice immédiat se traduit par les paiements immédiats moins les coûts immédiats. Les paiements immédiats peuvent être une compensation financière, dans le cas d'une entreprise commerciale, ou bien l'utilité personnelle ou collective de la correction d'un *bug*, dans le cas d'un projet *open source*. Les coûts immédiats sont expliqués par le temps investi dans le développement du logiciel. Le temps accordé à un projet est du temps qui

n'est pas accordé à un autre. Ces coûts sont plus ou moins élevés en fonction du plaisir pris par le programmeur dans le projet auquel il participe.

Le bénéfice retardé comprend deux motivations différentes : l'aspect carrière d'une part, qui est défini par toutes les offres d'emplois et opportunités futures ; et d'autre part la reconnaissance des pairs au sein de la communauté des programmeurs. Bien que ces deux motivations soient différentes et donnent naissance à une multitude de programmeurs différents, d'un point de vue économique elles sont similaires et peuvent dès lors être regroupées sous un ensemble « *signaling incentive* », qui peut se traduire par « l'envie de se faire remarquer ».

Selon Holmström (1999), cette motivation est très forte car

- au plus la performance est visible auprès de l'audience concernée,
- au plus l'impact de l'effort sur la performance sera élevé, et
- au plus informatif est la performance à propos du talent.

Il apparaît donc que les programmeurs vont vouloir travailler sur un projet ayant beaucoup de visibilité et une grande audience (Lerner & Tirole, 2000).

3.1.2. Les avantages de l'*open source*

Maintenant que nous avons eu un regard sur l'équation du programmeur, nous allons voir pourquoi l'*open source* est plus intéressant pour les programmeurs et dès lors l'entreprise.

Commençons par le paiement immédiat.
Pour la majorité des programmeurs *open source*, la compensation est égale à zéro. Néanmoins les utilités personnelle et collective sont très élevées, bien plus que dans un programme propriétaire, et diminuent ainsi les coûts du programmeur (Lerner & Tirole, 2000).

D'un point de vue personnel, le programmeur peut corriger les *bugs* et apporter de nouvelles fonctionnalités au logiciel qu'il utilise.

D'un point de vue collectif, en développant un code accessible à tous, celui-ci peut être utilisé par les écoles et universités pour former les programmeurs de demain. Ceci est appelé l'*effet alumni*.

Intéressons-nous maintenant aux bénéfices retardés.
La *signaling incentive* peut être accentuée de deux façons par l'*open source* (Lerner & Tirole 2000) :

- Meilleure mesure de la performance :
Tout le monde peut voir qui a réalisé quoi, si cela est utile et fonctionne correctement, ainsi que le niveau de difficulté de la tâche et la manière dont elle a été effectuée.

- Main libre :
Le programmeur peut réaliser ce que bon lui semble et prendre les pleines responsabilités de son sous-projet.

De manière générale, il peut être compris que le monde de l'*open source* offre une belle exposition aux programmeurs au sein de leur communauté. Comme l'a dit Eric Raymond (1999), une bonne réputation au sein de sa communauté entraine plus de considération et de coopération de la part de ses pairs, ainsi qu'un statut plus important.

Outre le plaisir de coder (Hars & Ou, 2001), c'est un moyen de présenter ses forces aux yeux de tous. C'est d'ailleurs pourquoi les OSS sont pour la plupart très performants mais pas facilement utilisables par les utilisateurs finaux aux compétences en informatique limitées. En effet, atteindre la finition d'un programme propriétaire, d'un point de vue interface et facilité d'utilisation, requiert beaucoup de travail fastidieux qui n'est pas forcément attractif pour un programmeur (Valloppillil, 1998).

3.2. Business modèles

Bien que l'*open source* soit à la base tourné vers le développement d'un produit plus que du business proprement-dit, de nombreuses entreprises en ont fait leur source de revenu au cours de ces dernières années. En atteste l'exemple de Red Hat dont le chiffre d'affaires a dépassé les 2 milliards de dollars pour l'année 2015-2016 et qui compte plus de 8300 employés.
Nous allons jeter un regard sur les différents business modèles entourant l'*open source*.

3.2.1. Les producteurs communautaires

Ceci est le cas de figure le plus simple : un groupe de développeurs passionnés crée un produit et donne gratuitement l'accès au code source à toute personne intéressée (Krishnamurthy, 2005). De nombreux utilisateurs prennent part au projet pour développer leurs connaissances et jouissent en retour d'un support efficace grâce à la communauté (Lakhani & Von Hippel, 2003).

Il ne faut donc pas voir ce modèle comme orienté business. Tout comme une société ne doit pas voir ces produits comme concurrents mais plutôt comme un support pour développer son business *open source*. En effet, l'entièreté du code source est accessible à tous et par tous.

La communauté a plus ou moins main mise sur l'avenir du code en fonction du choix de la licence, qui est réalisé par la personne lançant le projet. Comme vu au chapitre précédent, qu'elle soit permissive ou restrictive, cela aura un impact conséquent sur les retombées des travaux dérivés. Le choix de licence a dès lors une influence sur les différents modèles qui seront présentés ci-après.

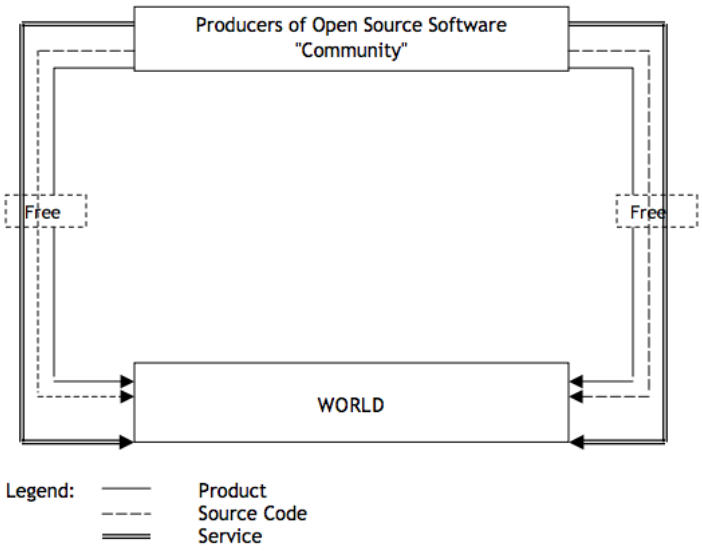


Figure 1: Les producteurs d'OSS communautaires (Krishnamurthy, 2005)

Sur la figure 1, on peut voir schématiquement le trajet effectué par les OSS réalisés par la communauté.

Linux est probablement l'exemple le plus connu pour ce business modèle-ci. En effet, il a pu se développer grâce à la participation de milliers de programmeurs.

3.2.2. Les distributeurs

Les distributeurs fournissent des accès aux codes sources et logiciels développés par la communauté. La plupart du temps, ils proposent des services de support pour générer du revenu. L'un des distributeurs les plus célèbres est Red Hat qui offre de nombreux services Linux.

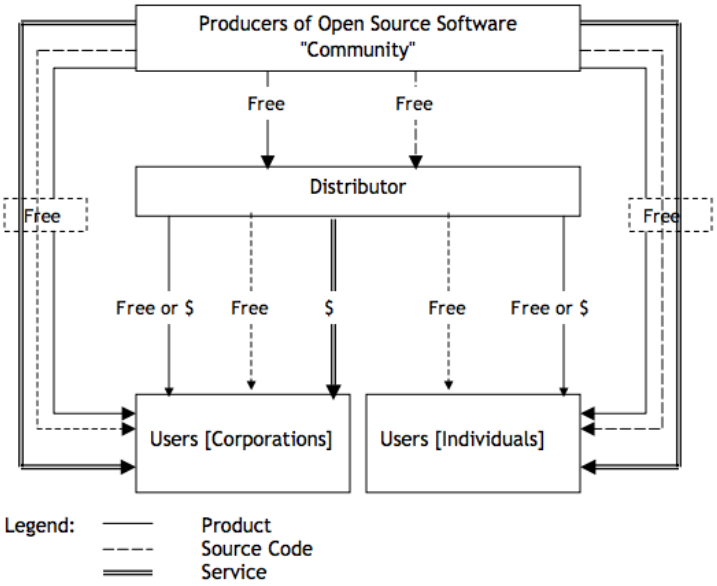


Figure 2: Le modèle des distributeurs (Krishnamurthy, 2005)

Comme le montre la figure 2, le code source est bien entendu distribué gratuitement aux utilisateurs, qu'ils soient des individus ou une société. Le produit en question peut être distribué soit gratuitement ou être facturé. Le prix dépend bien souvent des *packs* déjà associés au produit ou alors s'il est transmis par cd-rom, ce qui arrive plus souvent qu'on ne le pense (Krishnamurthy, 2005).

Intéressons-nous maintenant à la source de revenu des distributeurs : les services.
Ceux-ci concernent uniquement les utilisateurs entreprises et sont payants.
Prenons l'exemple de Red Hat, ils fournissent les produits Linux mais à côté de ça proposent un support clientèle, des formations Linux, de la consultance et encore bien d'autres choses (Red Hat, 2016). Le niveau de service fourni dépend du contrat signé entre l'entreprise et le distributeur.

3.2.3. Les producteurs de logiciels

Une fois de plus, les développeurs communautaires jouent un rôle primordial. En effet, leurs lignes de code sont énormément réutilisées pour le développement de nouveaux logiciels. Parfois appelés travaux dérivés lorsque l'entièreté d'un code est attachée à un produit existant. Cette formule est très avantageuse pour les développeurs

qui peuvent réduire leurs coûts de production et se concentrer sur les services qu'ils proposeront aux clients.

Néanmoins, la licence choisie par l'initiateur du projet communautaire, permissive ou restrictive, aura un impact sur la liberté d'action du développeur de logiciel.

3.2.3.1. Les logiciels sous licence permissive

Grâce à la licence permissive, il n'y a pas d'obligation de redistribuer le code source. La seule contrainte est de notifier d'où il provient. Donc, contrairement aux distributeurs, les développeurs de logiciels peuvent facturer tous les échanges entre eux et les utilisateurs finaux.

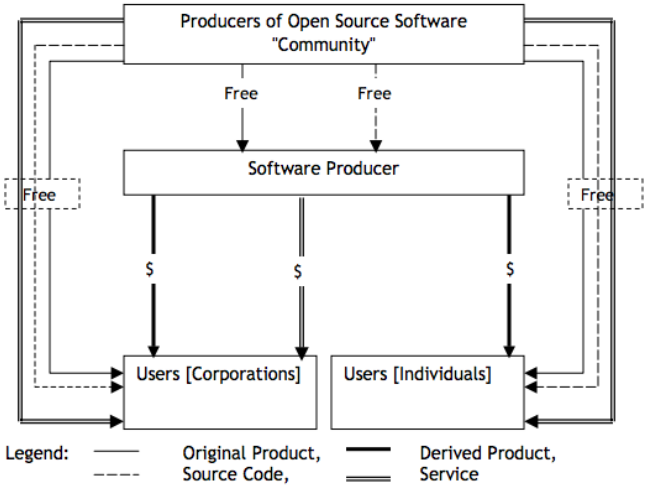


Figure 3: Le modèle des fournisseurs non GPL (Krishnamurthy, 2005)

Comme indiqué sur la figure 3, les utilisateurs finaux ont toujours accès au code source via la communauté. Cela peut être intéressant si le logiciel est constitué en grande partie d'un code communautaire (Krishnamurthy, 2005).

Au jour d'aujourd'hui et avec le succès grandissant du *cloud*, ce modèle est de plus en plus utilisé car il permet de générer des marges plus importantes tout en gardant le contrôle de son produit.

Prenons les exemples de Dokeos et Babelway qui développent leur produit sur une base *open source* pour ensuite le mettre à disposition de leurs clients sans pour autant donner accès au code source. Des interviews de réprésentants de ces sociétés sont disponibles en annexes.

3.2.3.2. Les logiciels sous licence restrictive

La licence restrictive, quant à elle, contraint le développeur à donner un libre accès aux utilisateurs. En observant la figure 4, nous remarquons que nous sommes dans un schéma quasi semblable à celui des distributeurs (figure 1).

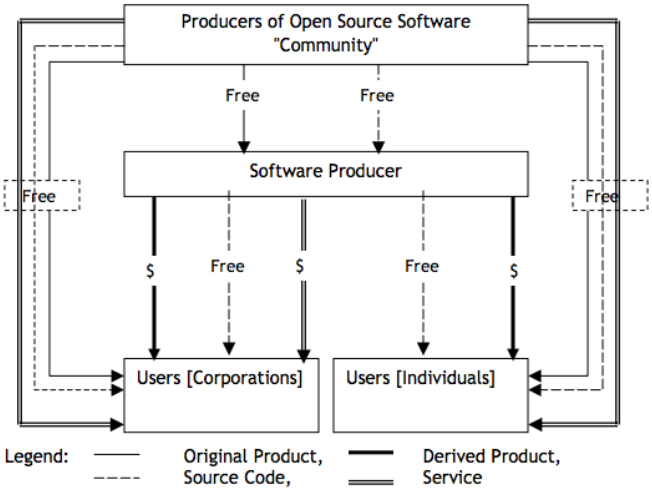


Figure 4: Le modèle des fournisseurs GPL (Krishnamurthy, 2005)

La différence entre les deux est toutefois importante :

- Le distributeur propose des services et du support liés au code source émanant de la communauté.
- Le développeur de logiciel sous licence restrictive propose un nouveau produit basé sur le code source communautaire, il donne accès au code source et propose des services et du support pour son logiciel.

Il est difficile de trouver des exemples d'entreprises commerciales utilisant ce modèle. Elles auront tendance à favoriser les licences non restrictives leur permettant d'assurer plus facilement du revenu. Nous pouvons toutefois à nouveau citer Red Hat.

En effet, le redistributeur Linux propose également des produits dérivés dont il donne l'accès au code source.

3.2.3.3. Différences

L'accès au code source dans le cas du modèle restrictif est propice aux innovations suite à l'implication des utilisateurs et dès lors, encourage également la loyauté envers le produit. Nous avons déjà évoqué ces avantages de l'*open source* dans le chapitre 1.

Cependant, la société permet également l'accès au code source à ses concurrents, qui pourraient ainsi s'en inspirer.

Finalement, la différence se joue également sur la cible recherchée. Un modèle permissif aura des clients ne se souciant uniquement que du bon fonctionnement du logiciel. Tandis que le modèle restrictif attendra un retour de la part de ses clients afin de faire évoluer le produit (Krishnamurthy, 2005).

3.2.4. Fournisseurs de services

Ce modèle-ci est également très simple. Tant que le logiciel a un accès gratuit, un service sera offert.

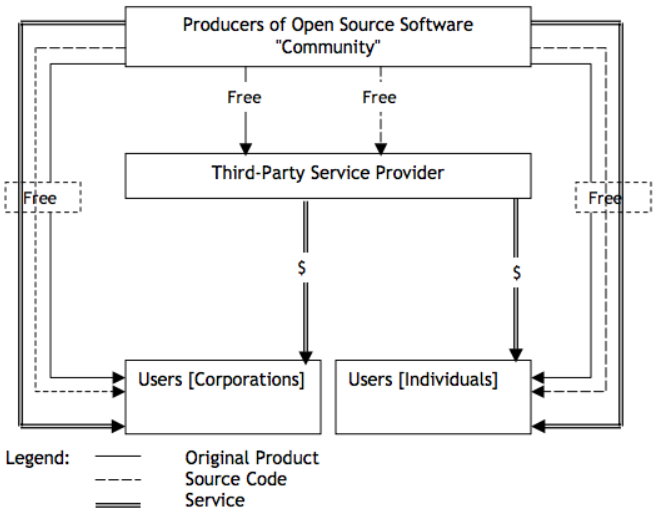


Figure 5: Le modèle des fournisseurs de service tiers (Krishnamurthy, 2005)

Attention, nous ne sommes pas dans le cas d'un distributeur. Comme indiqué sur la figure 5, le parti tiers ne propose que du service, ce qui est d'ailleurs son unique source de revenu.

Pourquoi alors utiliser des tiers partis et non un distributeur ?
Etant plus petits, ceux-ci sont souvent locaux et peuvent dès lors venir sur place pour régler le problème (Kryshnamurthy, 2005). De plus, un service payant est souvent de meilleure qualité qu'un service gratuit. Cet argument peut également être utilisé en faveur du distributeur par rapport à la communauté.

Dans ce cas-ci, nous pouvons prendre l'exemple belge de Kangaroot.
L'entreprise s'est spécialisée dans la consultance et l'offre de service pour les solutions Linux, principalement, et autre programmes *open source*.

3.3. L'*open source* d'un point de vue marketing

Le marketing est un sujet qui pose question au sein du business entourant l'*open source*, il serait vu comme une faiblesse pour de nombreux observateurs. Toutefois, cela n'empêche pas Linux de prendre des parts de marchés à Windows, ou Firefox à Internet Explorer. Dès lors, comment rivaliser avec des concurrents possédant des budgets bien

plus importants pour la promotion de leur produit ? Le marketing sera donc abordé sous cet aspect, ainsi que par une brève étude de cas avec la campagne de Mozilla au début des années 2000.

3.3.1. L'importance du bouche à oreille

Comme il a été dit précédemment, l'une des plus grandes forces de l'*open source* est bien entendu sa communauté, et au plus un projet est prometteur, au plus celui-ci sera attractif pour les programmeurs et au plus sa visibilité sera accrue.

Ceci est très important car cela signifie qu'il n'y a pas besoin d'engager une personne ou une société extérieure pour poster de faux posts sur des blogs. En effet, la promotion du logiciel se fait naturellement. Les personnes y participant vont enclencher un effet de bouche à oreille au moyen de tweets et de posts sur des forums ou blogs (Bertrand, 2009). Le bouche à oreille est d'ailleurs l'un des outils marketing les plus efficaces. Les utilisateurs sont bien souvent plus enclins à choisir un produit ou logiciel qui leur ont été recommandés par des pairs (Trusov, Bucklin & Pauwels, 2009). Cette crédibilité est notamment appuyée par le caractère non-économique du produit et le parrainage des acteurs majeurs du marché (Bertrand, 2009).

Outre le bouche-à-oreille effectué par les pairs, une société peut démarrer elle-même ce processus. En étant active sur les réseaux sociaux, forums et blogs, en prenant part aux discussions et répondant aux questions touchant de près ou de loin le domaine concerné par son logiciel, elle peut rapidement se faire un nom au sein de la communauté (Groganz, 2012).

Toucher le public en dehors de la communauté représente la face cachée de l'iceberg. Ceci est plus coûteux et se rapproche des techniques de marketing traditionnelles. Les points importants à respecter pour un logiciel sont d'expliquer concrètement ce qu'il fait et comment il fonctionne. Il est dès lors conseillé de fournir des fiches techniques détaillées ainsi que des études de cas lorsque l'on communique à la presse ou aux

potentiels nouveaux utilisateurs (Groganz, 2012). Il est important d'établir une communication claire, précise et consistante.

3.3.2. Le branding

Le *branding* est sans aucun doute une conséquence de ce bouche-à-oreille. Les logiciels *open source* bénéficient d'un effet de concentration. Au plus les développeurs et utilisateurs en parlent, au plus du monde va se mettre à travailler dessus et à les utiliser. Cela ayant bien sur une influence positive sur le logiciel en question.

Dès lors, les noms du logiciel et de l'éditeur deviennent une force et un atout marketing important. C'est un gage de qualité qu'on retrouvera sur de nombreux sites d'applications moins connues.
Placer le logo de Red Hat sur sa page d'accueil va rassurer les visiteurs. Cela fonctionne de la même manière que les labels sur les produits vendus en magasin. C'est un moyen facile et utile pour crédibiliser son produit aux yeux des utilisateurs.

Généralement, les personnes profitant de ce *branding* ne sont pas les entreprises éditrices. Celles-ci jouissent déjà d'une côte positive au sein de la communauté et leur marketing se fait de façon naturelle. Ce sont les plus petits développeurs qui vont utiliser ces noms-là. C'est une sorte de win-win, le logiciel moins connu va profiter de la côte de popularité du produit et en retour accroit la visibilité de celui-ci.

3.3.3. L'exemple Firefox

Le cas de Firefox est un très bon exemple pour voir à quel point la communauté peut avoir un impact important sur la campagne marketing d'un produit.

En 1998, Netscape, qui avait du mal à concurrencer Internet Explorer de Microsoft dans le domaine des navigateurs web, décida de démarrer le projet *open source* Mozilla. Ce faisant, il espérait pouvoir tenir le rythme grâce à l'implication de la communauté.

Le projet a réussi à rassembler plusieurs milliers de développeurs qui se sont fait un devoir de répandre Firefox au maximum.
Quatre sites web ont été développés pour l'occasion :

- un site consacré pour le téléchargement du navigateur
- un site pour la campagne marketing.
  Toutes les informations y étaient regroupées et permettaient aux volontaires de connaître les meilleurs moyens pour répandre l'application.
- un site pour la transition depuis Internet Explorer avec tous les arguments et raisons pour utiliser Firefox
- un site affichant la liste des sites non compatibles avec Firefox.

Tous ces sites étaient reliés les uns aux autres et ont eu un impact très conséquent sur la propagation de Firefox. C'était également un moyen facile d'organiser la campagne de façon efficace (Krishnamurthy, 2009).

Néanmoins, le fait d'armes le plus impressionnant et le plus visible de cette campagne reste sans aucun doute la double page publicitaire dans le New York Times du 16 décembre 2004 (voir Figure 6).

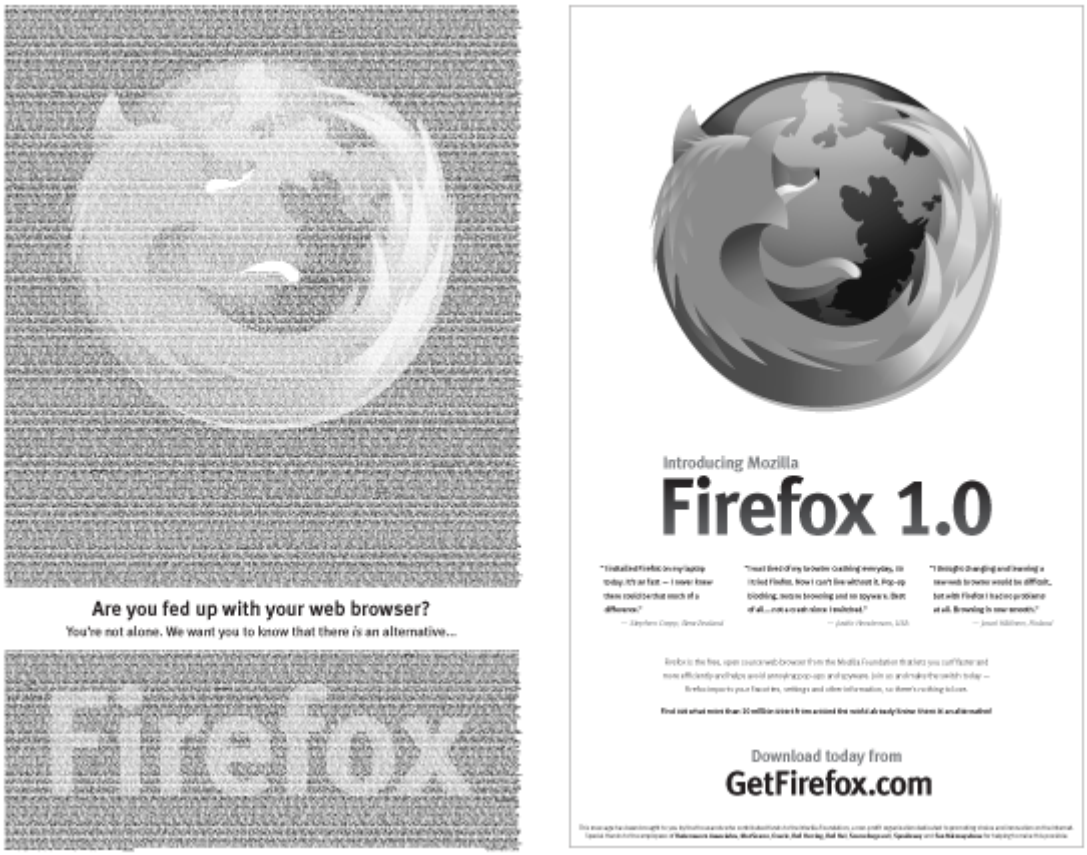


Figure 6: Publicité Mozilla Firefox dans le New York Times (New York Times, 2004)

Plus que la visibilité que cette action a offerte à Firefox, ce sont les conditions dans lesquelles elle a été réalisée qui impressionnent. En effet, le coût d'une double page publicitaire dans le New York Times étant très élevé, la communauté de Firefox a décidé de lancer une campagne de financement et est parvenue à récolter les 250 000 dollars nécessaires grâce à plus de dix mille donneurs pour un don moyen de vingt-deux dollars. Les participants ont pu voir leur nom apparaître sur la publicité (Krishnamurthy, 2005). Cet évènement apparaît comme un précurseur du *crowdfunding* et traduit l'engouement présent autour du projet *open source* Firefox.

Via cette annonce dans un journal comme le New York Times, l'un des objectifs était également de toucher un maximum de personnes et non uniquement des informaticiens. Le plus grand défi de cette campagne était de réunir aussi bien les spécialistes que les utilisateurs lambda sous un même navigateur.

Les conséquences ne se sont pas faites attendre. Une étude menée par *WebSideStory* indique que les parts de marché d'Internet Explorer sont descendues à 90,6% le 12 janvier 2005. Une première en trois ans. (Krishnamurthy, 2009).

La campagne marketing ne s'est pas arrêtée là et a poursuivi ses actions en ligne. Le web est alors devenu couvert de bannières et d'articles mentionnant Firefox. Les leaders informatiques mentionnent le navigateur sur leurs blogs et enclenchent ainsi un effet de bouche à oreille massif.

Les chiffres sont quant à eux éloquents. Firefox est passé de 2% de parts de marché en juillet 2004 à 12% en juin 2007. Le succès est encore plus marqué en Europe où Firefox atteignait 24,1% en mars 2007 (Krishnamurthy, 2009).
Aujourd'hui, d'après *Net Market Share* (2016), Firefox est le troisième navigateur avec 7,98% derrière Chrome (48,65%) et Internet Explorer (31,65%).

4. Exportation de l'*open source*

Ce chapitre va brièvement traiter de l'extension de l'*open source* à d'autres domaines d'activités.

4.1. *Open Data*

Selon Eduscol (2013),
"Une donnée ouverte (en anglais *open data*) est une information publique brute, qui a vocation à être librement accessible et réutilisable. La philosophie pratique des données ouvertes préconise une libre disponibilité pour tous et chacun, sans restriction de *copyright*, brevets ou d'autres mécanismes de contrôle.

En informatique, une donnée ouverte est « une information structurée ou pas, publique ou privée et généralement non utilisable par un humain mais interprétable par une machine. »

Cela s'inscrit également dans les mouvements de *open knowledge* où les individus considèrent que les accès aux informations et aux savoirs devraient être un droit pour chaque personne.

4.2. *Open Hardware*

Selon l'Open Source Hardware Association (OSHWA, 2016) , « Le Matériel Libre (OSHW – OpenSource Hardware) est un terme qui regroupe des produits tangibles — machines, appareils ou tout dispositif physique — dont les plans ont été rendus publics de telle manière que quiconque puisse les fabriquer, modifier, distribuer et les utiliser. »

Ce mouvement s'inspire directement du mouvement *Open Source* et rencontre de plus en plus d'adhérents. Peu à peu cela s'organise et il y a notamment treize conditions à remplir pour pouvoir qualifier un matériel de libre.

4.3. *Open Organization*

The Open Organization est un livre écrit par le CEO de Red Hat, Jim Whitehurst. Dans celui-ci, il explique de quelle manière l'application de certaines pratiques du développement de logiciel *open source* s'adapte parfaitement à l'échelle d'une entreprise et de quelle façon cela permet à l'entreprise de mieux fonctionner.

Le concept de base de ce nouveau type de gouvernance est d'intégrer au développement du produit ou du service aussi bien les acteurs internes à l'entreprise que les externes. Cette lecture est notamment recommandée par John Chambers (président exécutif de Cisco), Michael Dell (CEO de Dell) ou encore Jeanie Daniel Duck (ancienne directrice du Boston Consulting Group et auteur spécialiste en gestion du changement).

4.4. *Open Research*

Rachel Harding, une chercheuse de l'Université de Toronto, effectue des recherches sur la maladie de Hutington. Dans ce cadre, elle a décidé de partager ses avancements au jour le jour en espérant faire avancer ses études plus rapidement.

Il est donc possible de suivre en temps réel ses rapports sur son blog http://labscribbles.com/. En rendant les articles accessibles au grand public, elle espère pouvoir bénéficier de la collaboration de la communauté et ainsi augmenter les chances de mener ses recherches à terme.

Il y a également la *« Open Source Seeds Initiative »* qui veut lutter contre les multinationales agricoles qui s'accaparent toutes les semences. Celle-ci vise à promouvoir le partage de graines entre les semeurs (OSSI, 2016).

Partie 2. Analyse empirique

Cette analyse est basée sur les interviews réalisées auprès de différents acteurs impliqués quotidiennement aux technologies de l'information, qu'elles soient *open source* ou non.
Dans un premier temps, la méthodologie utilisée sera décrite.
Ensuite, les principales ressemblances et différences avec la théorie seront abordées.
Enfin, deux chapitres aborderont la question de recherche sur l'intérêt de l'*open source* pour les entreprises. L'un sera centré sur les entreprises développant des logiciels et proposant des services, l'autre posera le point de vue des entreprises utilisatrices.

5. Méthodologie

5.1. Design des interviews

Les interviews ont toutes été effectuées sur base de discussion ouverte à partir de questions générales sur l'*open source*. Une seule interview fut faite via e-mail, par facilité pour l'intervenant.

La question de départ portait sur « Pourquoi/Pourquoi pas de l'*open source* ? », la discussion pouvant ainsi démarrer et se diriger vers les zones de connaissances des intervenants.

Différents aspects comme les avantages ou désavantages, l'impact financier et l'évolution du marché ont systématiquement été abordées au cours de ces conversations.

De manière générale, les interviews ont duré trente à quarante minutes.

5.2. Intervenants

Pour mener à bien cette analyse, onze intervenants ont été contactés. Ceux-ci viennent d'entreprises et de profession différentes.

Mr. Beauprez est co-fondateur de l'association Internet of Things Belgium. Il est architecte en technologie de l'information et a assisté à l'évolution de l'*open source* depuis son émergence.

Mr. Bostoen est actuellement responsable des technologies de l'information (CIO) chez AG Real Estate. Cela fait maintenant plus de dix ans qu'il exerce cette fonction après un premier passage chez Confinimmo.

Mr. De Praetere est le CEO de Dokeos, un *Learning Management System*, depuis maintenant douze ans. Il publie également de nombreux articles concernant l'*e-learning*.

Mr. Derveau est consultant en architecture informatique pour le Service Public de Wallonie. Il a également pris part à un projet *open source* pour la Commission Européenne.

Mr. Mangen est manager chez CSC, une société de technologie de l'information.

Mr. Pasture est directeur technique chez Babelway, une plateforme d'intégration de données et de communication B2B, depuis maintenant neuf ans.

Mr. Pinckaers est le CEO de Odoo, anciennement OpenERP.

Mr.Schröder est responsable technologique pour Microsoft en Belgique et au Luxembourg depuis dix ans.

Mr. Senterre est responsable des technologies de l'information et de la communication pour Prayon, leader mondial dans le secteur des phosphates. Cela fait vingt ans qu'il travaille dans le domaine des technologies de l'information.

Mr. Viseur est expert en management et technologies *open source*. Il travaille au CETIC et à l'Université de Mons.

Mr. Waroquiers est responsable du développement des systèmes du traitement de volet de la gestion des flux du trafic européen chez EUROCONTROL.

5.3. Analyse des interviews

Une fois les interviews réalisées et retranscrites, un encodage a été réalisé afin de regrouper et classer toutes les informations par thème. De cette manière, il a été possible d'analyser les sujets récurrents, et également faire ressortir les points avec des avis divergents. Cela a également permis de se focaliser sur données pertinentes.

6. Liens avec la théorie

Ce chapitre a pour but de mettre en lien ce qui a été abordé dans l'état de l'art et ce qui a été dit lors des interviews. Il permettra de mettre en avant les points communs et les différences avant d'aborder l'analyse concernant les entreprises.

6.1. Avantages et inconvénients

De manière générale, la théorie et la pratique vont dans la même direction lorsque la question des avantages et inconvénients de l'*open source* est abordée. Il y a tout de même quelques nuances mais qui résultent surtout d'une différence de point de vue. En effet, les intérêts d'une entreprise cherchant à maximiser ses revenus et faire tourner ses activités sont différents de ceux d'un développeur se lançant dans un projet *open source* ou encore des intérêts des théoriciens se basant sur les postulats idéologiques et philosophiques qui ont vu la création de ce mouvement.

6.1.1. Coûts

Bien que les coûts soient moins élevés à l'acquisition, ce n'est pas toujours un critère qui est retenu pour les grandes sociétés. De plus, le *Total Cost of Ownership* de ceux-ci n'est pas forcément moins élevé que celui d'un programme propriétaire.
En effet, une entreprise fera appel aux services proposés dans la plupart des cas de façon à tirer le maximum du logiciel et également être sûr de l'utiliser correctement.
Dès lors, malgré un coût d'acquisition faible, les coûts d'utilisation sur la durée vont augmenter le coût total du logiciel.

Contrairement à un logiciel propriétaire où la licence est payée et pour laquelle on ne sait pas toujours où finit l'argent dépensé: maintenances, mise à jour, marketing,..., dans le cas du logiciel *open source*, c'est beaucoup plus transparent. On sait pour quels services on paie et l'argent est dans la plupart des cas réinvesti dans la recherche et le

développement. En comparaison avec des entreprises plus traditionnelles, les coûts marketing sont moins importants. Il n'y a généralement pas de campagnes coûteuses car la communication se repose principalement sur le bouche-à-oreille au sein de la communauté.

6.1.2. Innovation

De part la structure de l'organisation et les relations plus horizontales, la sphère *open source* est plus apte à proposer des innovations. Cependant, il n'est pas forcément d'avis que cela soit applicable à tous les projets. Un projet de trop grosse taille ne sera pas spécialement plus propice car en fonction du nombre d'intervenants, il peut avoir tendance à s'éparpiller dans toutes les directions.

Des projets de plus petites envergures sont en revanche très propices aux innovations. En effet, les développeurs se servent des lignes de code dont ils ont besoin dans les bibliothèques *open source* et peuvent les façonner à leur façon pour réaliser leur projet. La source de savoir déjà disponible permet aux développeurs de ne pas devoir réinventer la roue, de se concentrer pleinement à la plus-value qu'ils veulent apporter et de développer leurs nouvelles idées.
De plus, il est important de noter que les gros projets ont tendance à être de plus en plus divisés en composants de plus petites tailles, ce qui peut favoriser l'innovation par l'utilisation de l'open source.

6.1.3. Sécurité et qualité

Les personnes développant des logiciels au moyen de l'*open source* s'accordent à dire que c'est nettement plus sûr, et rejoignent ce qui est énoncé dans la théorie. En effet, la transparence du code permet un débogage efficace par la communauté et les utilisateurs.
Un exemple illustre parfaitement ceci : lorsqu'une société passe de bureaux individuels pour ses employés à un *open space*, il n'y a pas lieu d'utiliser un système de pointage

pour s'assurer que tout le monde respecte ses horaires. En effet, comme tout se passe à la vue de tout le monde, les employés seront très vites au courant si quelqu'un ne réalise pas sa part du travail.
C'est le même scénario avec de l'*open source*. Le développeur, sachant que tout le monde peut analyser son apport, a un incitant pour développer ses lignes de code du mieux qu'il peut. Ce qui n'est pas forcément le cas d'un logiciel propriétaire qui, dans le cadre de cet exemple, pourrait apparaître comme un lieu de travail avec des bureaux clos.

Toutefois, lorsqu'on s'adresse aux utilisateurs, ils émettent certaines réticences.
Selon eux, la question mérite d'être posée et avec l'avènement du *cloud*, la cyber sécurité devrait être au coeur des conversations.
Ceci s'explique également par la consonnance du terme « *open* » en opposition avec le terme « fermé » ou « propriétaire ». Selon eux, c'est plus au niveau réputationnel que cela peut poser problème, par exemple dans le cas d'une banque ou de l'armée. Or, les exemples sont légions sur les gouvernements, commissions ou armées utilisant de l'*open source*.

6.1.4. Support

Le support est un argument pour lequel la théorie et la pratique ne s'accordent pas entièrement. Certes la communauté est de manière générale assez réactive face aux problèmes posés mais il est toutefois possible que votre question se perde dans la masse.

Dans le cadre de la pratique, c'est différent. Les utilisateurs faisant également appel aux services proposés par l'entreprise, pourront donc jouir d'un support permanent de la part de leur fournisseur. Et selon eux, ce support est de très bonne qualité, voire meilleur que celui d'un logiciel propriétaire car il représente la source de revenus principale de l'entreprise en question.

Le parallèle avec le point sur le coût peut être fait, dans le cas d'un propriétaire, on ne sait pas forcément où va l'argent investi. Tandis que dans le cas d'un service, c'est exactement ce que l'on paie.

6.1.5. Sur mesure

L'un des avantages de l'*open source* est bien entendu qu'il permet de faire du sur mesure là où le propriétaire ne peut pas toujours entièrement s'adapter aux besoins spécifiques de l'entreprise. Avec l'*open source*, il y a la possibilité d'ajuster le logiciel en interne ou alors avec le fournisseur et les services qu'il propose.

Cependant, du sur mesure n'est pas accessible ou nécessaire à tout le monde.
Prenons l'exemple de Odoo : il propose son outil *open source* pour les plus grosses entreprises en agissant également comme intégrateur pour être sûr que tout soit adapté ; alors que pour les plus petites entreprises, il offre du SaaS (*Software as a Service*, ce terme sera défini plus en détails dans le point 6.2.2.) avec un service plus standardisé. Bien que ce choix de Odoo soit basé sur le sentiment de pouvoir répondre au mieux aux besoins de leurs clients, il est important de noter que cette décision est propre à cette entreprise et n'a pas loi de vérité générale.

6.1.6. Complexité

Bien que la taille d'une entreprise n'est pas un critère sur le choix de l'*open source* ou non, il est plus probable de trouver de l'*open source* dans une grande société et du service ou du propriétaire dans une petite.
En effet, même si le support est disponible, il est important d'avoir des connaissances en informatique afin d'utiliser correctement un programme. Ce qui est bien sur le cas pour un logiciel *open source*. Or une petite entreprise n'aura pas toujours une personne avec les connaissances adéquates en son sein. Dans ces cas-là, il est fort probable qu'elle opte pour un produit totalement fini et facile d'usage.

Par contre, une grande entreprise a de facto plus de chances d'avoir des employés qualifiés pour l'utilisation de ces technologies. L'utilisation d'un programme *open source* demande une certaine formation de ceux-ci, ce que peuvent plus facilement se permettre des sociétés d'une certaine taille

6.1.7. Visibilité

Une fois de plus, la pratique rejoint la théorie. Développer du code *open source* permet de se montrer au sein de la communauté et ses développeurs. C'est l'un des moyens les plus efficaces pour faire parler de son produit dès le début et lancer, sans frais, sa campagne marketing. Cela a donc un impact sur la vitesse de pénétration du marché, qui est le point suivant.

Cela s'applique également pour les entreprises proposant du service en ligne mais utilisant de l'*open source* pour développer leur *software*. Si celles-ci apportent des modifications aux lignes de codes qu'elles utilisent, elles développeront sans nul doute un contact avec d'autres développeurs qui commenceront à en parler autour d'eux et lanceront du coup la machine bouche à oreille.

6.1.8. Vitesse de pénétration

Un point qui n'a pas été abordé dans la théorie mais qui est sorti dans plusieurs conversations est la vitesse de pénétration de marché qu'offre l'*open source*. Cela rejoint l'idée de ne pas réinventer la roue. En utilisant de l'*open source*, il y a une grande partie du travail qui est déjà réalisée par la communauté et on peut dès lors se concentrer sur ce qui fera notre force. Contrairement à un logiciel propriétaire qui devra être monté de bout en bout et demandera, du coup, plus de temps avant de pouvoir être lancé sur le marché.

Comme mentionné au point précédent, la visibilité a une influence positive sur la vitesse de pénétration. Grâce au bouche à oreille déclenché au sein de la communauté, le produit arrive plus rapidement entre les mains des personnes ciblées

6.2. Business modèles

Malgré certaines similitudes, les business modèles mis en avant au cours des interviews diffèrent de ceux énoncés dans la théorie. Ceci peut s'expliquer par l'évolution du mouvement au cours de ces dernières années.
Dans une industrie en perpétuel mouvement, il n'est donc pas étonnant de voir que la situation ait évolué. On peut regrouper les business modèles en trois groupes différents.

6.2.1. Fonctionnalités et avantages

Cette première catégorie est celle qui se rapproche le plus des modèles vus dans la partie théorique. La société développe son logiciel avec de l'*open source* et y donne un accès gratuit. Cependant, au dessus de cette version *open source*, s'ajoute une version payante, parfois même *closed source*, possédant bien plus de services, fonctionnalités et avantages que la version *freemium*. C'est d'ailleurs sur cette version-là qu'est basée le business modèles.

La version *freemium* permet aux utilisateurs de se lancer dans l'utilisation du logiciel et de commencer à le faire tourner mais très vite. C'est également un bon moyen de voir si ce produit répond aux besoins.
Une fois le choix fait, le pas vers la version premium est très vite effectué. Celle-ci permet de recevoir un service sur mesure et plus adapté à nos besoins. Cela évite également de devoir investir du temps et de l'argent en formation.

6.2.2. SaaS

Le *Software as a Service* est de plus en plus présent avec l'avènement du *cloud*. Désormais, l'entreprise ne transmet plus son produit, donc plus le code, elle ne fait que du service. Les applications étant hébergées sur le *cloud*, il n'y a plus lieu d'en fournir une à ses clients. Nous observons donc une rupture avec les modèles vus dans la théorie, il n'y a donc plus de transfert de produit.

Là où l'*open source* est intéressant, c'est dans le développement de l'application. On en revient une fois de plus à l'argument du pourquoi réinventer la roue. La société se sert des bibliothèques pour la base de leur produit, le peaufine en interne et peuvent ensuite mettre leur application sur le *cloud*.

Une fois sur le *cloud*, le moyen de générer du revenu est le même que pour la première catégorie. Les utilisateurs ont accès à une version *freemium* et une version premium. La version *freemium* sera très standard, tandis que la premium offrira toute une gamme de services

Drupal et Odoo sont deux exemples de sociétés utilisant ces deux catégories de business modèles. Toutes les deux ont commencé en proposant de l'*open source*, mais désormais proposent également du SaaS.

6.2.3. Librairies de donnés

Cette catégorie regroupe des entreprises du gabarit de Google, Facebook ou autre Instagram.
L'idée est très simple, ces sociétés lancent un nouveau projet en interne et donnent ensuite l'accès à la première version à la communauté *open source*. Cela va attirer de nombreuses personnes qui vont travailler dessus, le développer et le mettre à jour.

Après, il leur suffit d'utiliser les dernières versions pour leur propre usage. Elles ont donc pu développer un produit de qualité à très faibles coûts. De plus, elles ont entre temps récolté une quantité importante de données ayant énormément de valeur.

En effet, les personnes prenant part à ces projets ne sont pas uniquement des développeurs passionnés faisant ça durant leur temps libre. Il y a énormément de chercheurs et d'universitaires qui y investissent de nombreuses heures de travail. Ces données ont donc une valeur marchande assez élevée.

Nous pouvons également prendre l'exemple de Microsoft qui a lancé de nombreux projets *open source* depuis quelques années et qui a donné accès à de nombreux outils jusqu'alors que disponibles en interne. Nul doute que de nombreux développeurs s'y sont intéressés et cela a permis à Microsoft de récolter énormément de données en retour.

6.3. Marketing

D'un point de vue marketing, nous observons que la théorie se rapproche fortement de la réalité dans le cas de Odoo. Il n'y a pas eu ou très peu de communication; l'entreprise bénéficiant surtout du bouche à oreille et d'une grosse visibilité au sein de la communauté. Le nombre croissant d'utilisateurs au fil des ans se charge de faire passer le mot, Odoo a donc pu investir en recherche et développement plutôt qu'en communication. Les entreprises de services quant à elles recourent également à des techniques de marketing virales classiques comme la publicité ciblée.

Tout comme dans le cas Mozilla Firefox, les sites web des sociétés sont très détaillés et offrent de nombreuses informations concernant le produit ainsi que des études de cas permettant d'avoir un aperçu plus concret de ce qui peut être réalisé. Dans la plupart des cas il y a également la possibilité d'effectuer un essai gratuit pour se faire une meilleure idée de ce que réalise le logiciel.
Cette transparence peut également être un désavantage pour l'entreprise car ainsi, l'utilisateur a la possibilité d'essayer tous les programmes avant de choisir celui qui lui correspond le mieux. Ce qui n'est pas le cas avec du propriétaire.

7. Entreprises développeuses

Ce chapitre se concentre sur les entreprises développeuses. C'est à dire les sociétés dont le revenu est basé sur les services accompagnant un ou des logiciels, que ceux-ci soient totalement ou partiellement *open source* et que les clients aient accès au logiciel ou uniquement au service.

7.1. Mise en garde : l'*open source* comme business modèle

Comme indiqué dans la théorie, de l'*open source* pour de l'*open source* n'est pas viable. Nous nous trouvons alors dans le cas d'un groupe de développeurs passionnés voulant développer un logiciel mais sans perspective d'en retirer du profit. La réalité des entreprises est bien sûr différente. Il y a une obligation de générer du revenu si on veut assurer la pérennité de ses opérations.

La transparence de l'*open source* est une force magnifique mais est également un danger. Il est, en effet, difficile de se différencier de ses concurrents lorsque votre code source est accessible par tous. Cela a pour conséquence de faire de l'*open source* un univers où il n'y a pas ou très peu de place pour la concurrence.
C'est la loi du plus fort, seul le meilleur survit. La raison en est simple, les utilisateurs pouvant tester tous les produits sans réel engagement, finiront par se tourner vers le meilleur et il y aura un effet de consolidation.

C'est pourquoi l'*open source* comme tel n'est pas un business modèle viable. Il est très utile mais ce n'est pas lui qui permettra à votre entreprise de gagner de l'argent. Aussi performant soit votre logiciel, il reste "gratuit". Ce qui fera la force de votre entreprise, c'est d'être capable de proposer le service répondant parfaitement aux besoins de vos clients.

Bien que de plus en plus présent au sein des entreprises, le nombre de celles-ci proposant de l'*open source* est donc très limité. Néanmoins, cela ne veut pas dire que

celui-ci n'est pas un atout utile pour une entreprise informatique. Loin de là, l'évolution de la puissance des machines de calcul a en effet ouvert de nouvelles portes et de nouvelles opportunités.

Dans la mosaïque actuelle, une nouvelle place de choix est donnée à l'*open source* et tout semble indiquer qu'il a encore de belles années devant lui. C'est ce que nous allons aborder dans les points suivants.

7.2. Le cloud

Le *cloud* peut être défini comme *l'accès via un réseau de télécommunications, à la demande et en libre-service, à des ressources informatiques partagées configurables*. En d'autres mots, la délocalisation de la structure informatique (Mell & Grance, 2011).

Avec son arrivée, le *cloud* a chamboulé le paysage de l'informatique et a notamment permis l'envol du modèle SaaS. Cela a un impact considérable sur le développement de logiciel comme désormais il est facile de mettre son application en ligne sans devoir développer une plateforme qui hébergera le logiciel chez le client.

Cette redistribution des cartes à jouer a une conséquence directe sur l'*open source*. En effet, une étude de marché menée par la société North Bridge & Black Duck Software (2016) indique que le cloud est le lieu où l'*open source* sera le plus présent et les SaaS seraient le meilleur moyen de gagner de l'argent en utilisant de l'*open source*. Cela rejoint également le changement de positionnement de la part de Microsoft depuis quelques années.

En effet, outre le changement de direction, le *cloud* est présenté comme l'une des explications à l'ouverture de Microsoft envers l'*open source*. L'investissement pour le développement d'hébergeur n'a plus vraiment de sens et il est devenu plus intéressant de se concentrer sur les applications et services. Nous pouvons notamment observer cette transformation au travers du projet Microsoft Azur.

Ces évènements vont permettre à l'*open source* de se concentrer sur ce qu'il fait de mieux : être un outil de construction.

7.3. L'*open source* comme outil de construction

Grâce à ses bibliothèques et au nombre toujours croissant de ses projets, l'*open source* se positionne comme outil de choix pour développer une application et un service.

Il serait en effet contre productif de vouloir développer en interne toute une partie de son logiciel alors que ces pièces sont disponibles gratuitement au sein de la communauté. Comme nous l'avons déjà indiqué, cela libère énormément de temps et de moyen pour se concentrer sur ce que vous voulez apporter en plus.

Nous revenons une fois de plus à l'argument de ne pas réinventer la roue. Néanmoins, il y a une nuance, ne pas devoir refaire un travail qui a déjà été fait permet de laisser parler son imagination et sa créativité pour amener une solution innovante sur le marché. La roue ne sera pas réinventée certes mais il y a là l'opportunité de révolutionner ce qui se passe autour de celle-ci et en fin de compte faire en sorte que le véhicule avance beaucoup plus facilement et efficacement.

Nous pouvons voir l'*open source* comme les bases de la fondation. Il est d'ailleurs d'avis que ces bases sont actuellement les plus solides qui soient disponibles sur le marché. A cette solidité s'attachent également une grande souplesse et une flexibilité permettant d'ériger avec précision les buildings nécessaires pour répondre au besoin de nos clients.

7.4. Vers les logiciels hybrides

De nombreuses questions sont posées quant à l'avenir des logiciels propriétaires face à la présence toujours grandissante de l'*open source*. Dans bien des situations, la réponse n'est jamais blanche ou noire mais plutôt une certaine teinte de gris. Il est fort probable que ce soit également le cas pour les logiciels.

Il serait illusoire d'imaginer un monde entièrement *open source*, dès lors il est intéressant de regarder ce qui se passe entre ces deux opposées. Nous remarquons que les logiciels hybrides sont en fait le juste chemin. Ceux-ci sont tout simplement une combinaison des tendances observées au sein de l'*open source*.

D'une part, les business modèles tendent à se diriger vers de la prestation de service accompagnant une offre premium en parallèle à l'offre *freemium*.
D'autre part, l'*open source* se positionne comme l'outil de construction parfait. Nous obtenons un logiciel prenant le meilleur de chaque partie avec d'un côté une qualité technique supérieure pour répondre aux besoins des clients et de l'autre une source de revenu stable pour répondre aux exigences du marché et des actionnaires.

8. Les utilisateurs

Ce chapitre abordera la question de l'*open source* du point de vue des utilisateurs. Par utilisateurs, on signifie ici les sociétés utilisant un logiciel. Dans un premier temps nous parlerons de la présence de l'*open source* de façon générale dans les entreprises et les organes publics. Ensuite nous verrons ce que recherchent les utilisateurs et ce que peut leur apporter l'*open source*.

8.1. Ce que recherche l'utilisateur

Le fait de savoir si un programme est *open source* ou non n'est généralement pas un facteur déterminant pour l'utilisateur.
Les deux questions qu'il se pose en premier au moment d'acquérir un logiciel sont :

- Est-ce que ce produit répond à mes besoins ?
- Quel en est le prix ?

Il est impensable que l'utilisateur se procure un produit plutôt qu'un autre sur le simple fait qu'il soit *open source* s'il ne répond pas à ses besoins.

Dans bien des cas, l'entreprise, en fonction de ses moyens, fera un appel d'offres au marché ou une étude de marché. Elle analysera ensuite les différentes solutions disponibles pour enfin choisir celle répondant au plus de critères. Parmi ceux-ci, on retrouve la solidité, la sécurité, la durabilité, le support, l'intégration, l'assistance et encore bien d'autres.

D'une certaine façon, cela appuie l'idée qu'il ne faut pas baser son business modèle sur l'*open source* même. Les sociétés sont pragmatiques, elles vont prendre ce dont elles ont besoin. Comme nous l'avons dit précédemment, l'*open source* peut être un argument en plus pour le logiciel mais en aucun cas le principal. Néanmoins, comme nous le verrons dans le point suivant, le fait d'être *open source* peut faire partie des critères de sélection.

8.2. L'argument *open source*

Les arguments en faveur de l'*open source* ont déjà été listés dans la théorie mais ils étaient principalement axés sur le logiciel en lui-même. Ici, nous aborderons les avantages que représente l'*open source* pour l'organisation dans son ensemble.

8.2.1. Indépendance

Un des arguments phares est sans nul doute de pouvoir garder un contrôle total sur ses opérations et une certaine indépendance vis-à-vis du fournisseur. Grâce à la transparence du programme, on sait exactement ce qu'il se passe. Ce n'est pas toujours le cas avec du propriétaire où on n'est jamais à l'abri d'une porte arrière par laquelle le fournisseur récupère nos informations. Cet argument a été utilisé par Mr. Stallman dès le lancement de son initiative (1985).

C'est d'ailleurs l'une des raisons pour lesquelles l'*open source* a autant de succès avec les organes publics et les armées. Ne pas être informatiquement dépendant de partenaires privés leur permet d'éviter des fuites de dossiers critiques.

En outre, la Commission Européenne fait de l'*open source* un critère pour ses logiciels de communication. De cette façon, elle ne force pas l'installation de programmes couteux à ses Etats Membres et ne privilégie pas certains acteurs privés.

Un autre argument allant dans ce sens est la différence des types de contrats.
Avec un logiciel propriétaire on paie une licence à l'année tandis que pour le logiciel open source, le client va payer les services à l'utilisation et aura dès lors un meilleur contrôle sur ses mouvements et une réactivité accrue par rapport à d'éventuels changement sur le marché.

Toutefois, cet argument est à prendre avec un certain recul étant donné que de des contrats à l'utilisation sont également des plus en plus courant avec des logiciels propriétaires.

8.2.2. Durabilité

La durabilité est un critère important pour une entreprise. En effet, il n'est pas intéressant de démarrer une relation avec un partenaire si celui-ci arrête ses activités au cours des mois qui suivent.

C'est un argument qui divise lorsque l'on compare *open* et *closed source*. Pour certains, un logiciel propriétaire venant d'une grande société est gage de garantie pour l'avenir. Certes, il est vrai que les mastodontes ne s'écroulent généralement pas du jour au lendemain et il est légitime de se dire que c'est une assurance de stabilité et de durabilité.

Néanmoins, la question vaut la peine d'être creusée.
Prenons tout simplement l'exemple de Nokia, géant des télécommunications qui a mis fin à certaines de ses activités lors des dernières années ; ou encore Microsoft qui peut parfois mettre des projets en cours au placard.
En effet, personne n'est à l'abri de mauvaises surprises. Mais le problème dans ces cas-là, comme le code n'est pas accessible, c'est que les utilisateurs se trouvent tout simplement bloqués et n'ont d'autre choix que de se tourner vers un autre produit ou attendre que quelqu'un décide de racheter ou relancer le projet.

C'est à ce niveau que l'*open source* tire son épingle du jeu.
Dans un premier temps, nous supposerons que l'utilisateur ait opté pour une solution sûre auprès d'une entreprise affichant une stabilité suffisante. Ensuite supposons que celle-ci mette subitement fin à ses activités, l'utilisateur se retrouve toujours en possession du code source. Ce qui veut dire qu'il n'est pas bloqué et a toujours la possibilité de poursuivre ses activités en effectuant des mises à jour du programme.

Dans la pratique, il est plus probable que dans un cas comme dans l'autre, l'entreprise se tourne vers une nouvelle solution. En effet, les programmes ont besoin d'évoluer donc continuer avec la même version n'est pas une bonne idée. Et même avec un accès au code source, l'entretenir demanderait un investissement en temps et en argent considérable. Ce qui s'applique également au *closed source*, la majorité des grands contrats entre un éditeur de logiciel et une entreprise contient une clause stipulant que le client se verrait donner l'accès au code source en cas de cession des activités de l'éditeur.

8.2.3. Marché de niche

Les marchés de niche sont souvent des domaines où l'*open source* est très présent car très intéressant pour eux et plus facile d'obtenir le monopole. Les motifs sont simples, pour du marché de niche, l'importance du sur mesure est très importante. Il faut que le logiciel s'adapte parfaitement aux besoins du client. En effet, dans ces conditions, il serait trop coûteux pour une entreprise de développer une solution en propriétaire pour un ou deux clients.

Prenons l'exemple d'Eurocontrol qui s'occupe du trafic aérien en Europe.
Ils sont l'unique acteur sur ce marché et collaborent avec tous les aéroports européens. Ils ont développé leur logiciel de contrôle en interne en utilisant de l'*open source*. Cela leur donne moins de contraintes et leur permet de regarder ce qui se passe au niveau du code lorsqu'il y a un problème. Ou encore de faire appel à un service de support. Leur produit n'est en revanche pas *open* source, il n'est d'ailleurs pas commercialisé et reste au sein de l'entreprise. Il serait très peu probable de voir apparaître une société proposant une solution propriétaire pour le contrôle aérien et permettant ainsi à Eurocontrol d'outsourcer cette partie de leurs activités. Cela ne serait absolument pas intéressant pour l'entreprise. Il lui faudrait énormément d'investissements, aussi bien en temps qu'en argent pour atteindre le niveau d'expertise atteint par Eurocontrol et

développer un programme aussi performant. Et ensuite, il faudrait arriver à convaincre ceux-ci d'abandonner leur logiciel pour oser un pari différent.

8.2.4. *Ceteris paribus*

De manière générale, lorsque tous les critères de sélection de l'entreprise (caractéristiques techniques, prix, durabilité,…) sont rencontrés, celle-ci devrait opter pour le logiciel *open source* plutôt que propriétaire. Dans ce cas-ci, la société peut voir cela comme une qualité en plus au logiciel avec des avantages qui ont été cités ci-avant.

,Vu que dans la majorité des cas, une entreprise fait appel aux services proposés, la courbe d'apprentissage d'un logiciel est identique que celui-ci soit *open source* ou non ; sauf si cette entreprise dispose d'informaticiens qui sont familiers avec les programmes en question ou le langage utilisé.

Dans le cas d'un utilisateur final, celui-ci optera très probablement pour un logiciel propriétaire par facilité d'utilisation. Mais comme nous l'avons expliqué plus haut, ceux-ci ne sont que rarement la cible des développeurs *open source*.

Conclusion

Après ces analyses, il semblerait que le monde des logiciels se dirige vers une certaine tendance avec une redistribution des cartes et le modèle hybride est celui présentant les meilleurs arguments.

En effet, travailler avec de l'*open source* permet de profiter de tous les avantages que celui-ci offre d'un point de vue technique et construction du produit, la couche propriétaire permet quant à elle à l'entreprise d'assurer une base de revenus.

Car si le défaut majeur du logiciel *open source* est qu'il ne garantit pas de rentrées, il se positionne toutefois comme le meilleur outil pour le développement de logiciel et ne devrait pas disparaître du paysage informatique.
Que du contraire, des *database* propriétaires n'ont plus d'intérêt ni de raison d'être avec la présence de toutes les bibliothèques *open source*.

Du point de vue des clients, la situation est légèrement différente.
Ce qui importe le plus, c'est d'être équipé d'un logiciel répondant à leurs besoins et facile d'utilisation. Excepté si l'entreprise possède des informaticiens férus de codes et que ses activités l'obligent à développer également en interne, qu'un logiciel soit propriétaire ou *open source* n'a pas d'importance. La différence notable se fera au niveau du coût payé. Les licences annuelles des logiciels propriétaires étant généralement plus chères que les services "facturés à la minute" des éditeurs *open source*.

En revanche, l'*open source* est considéré, bien souvent à tort, comme étant moins sûr pour la sécurité et moins fiable pour la durabilité que les programmes propriétaires.
Ce qui joue en sa défaveur. Ceci s'explique par sa description.
En effet lorsqu'on le présente, c'est toujours sous la forme "... accès au code source…", ce qui peut encore susciter des réactions négatives si on ne s'y attarde pas plus. Il est vrai que c'est un phénomène encore peu connu de la grande majorité ou qui résonne comme

risqué pour des grosses organisations existant depuis une longue durée. Néanmoins, le *cloud* et le modèle hybride, mélange d'*open source* et de propriétaire, ou encore l'activité de Microsoft au sein du mouvement *open source*, devraient en partie améliorer la situation.

Au vu du changement apparu dans le monde informatique, nous pouvons nous poser la question par rapport à notre société en général.
Serait-il possible de se diriger vers un mode de vie où nous collaborerons tous ensemble et ainsi mettre le "nous" au dessus du "je" ?
La vie ne peut se résumer à vouloir tirer, seul, son épingle du jeu afin de devenir le plus riche possible. Or c'est ce que laisse bien trop souvent apparaître notre système actuel. Il doit être possible de trouver un juste milieu. Pourrions-nous dès lors appliquer ce système hybride à notre société ? Obtenir des résultats et générer de la richesse tout en encourageant le partage et la collaboration.

Sources

Anderson, C. (2012). *Makers: The New Industrial Revolution* : Random House.

Anderson, C. (2013). *Free : How today's smartest businesses profit by giving something for nothing*: Random House.

Anderson, C., & Wolff, M. (2010). The Web is dead. Long live the Internet. *Wired Magazine, 18*.

Bertrand, P. (2009). Le marketing, faiblesse ou force de l'*Open Source*?. En ligne sur le site du Journal du Net, www.journaldunet.com/solutions/expert/43840/le-marketing--faiblesse-ou-force-de-l-open-source.shtml

Bonaccorsi, A., & Rossi, C. (2003). Why open source software can succeed.*Research policy*, *32*(7), 1243-1258.

Bonaccorsi, A., Giannangeli, S., & Rossi, C. (2006). Entry strategies under competing standards : Hybrid business models in the open source software industry. *Management Science*, *52*(7), 1085-1098.

Bretthauer, D. (2002). open source software: A history. Information Technology and Libraries, 21(1), 3

Calof, J. L., Wright, S., & Fleisher, C. S. (2008). Using open source data in developing competitive and marketing intelligence. *European journal of marketing, 42*(7/8), 852-866.

Cazier, J. A., Shao, B. B., & Louis, R. D. S. (2006). E-business differentiation through value-based trust. *Information & Management, 43*(6), 718-727.

Ceruzzi, P. E. (2003). A history of modern computing. MIT press.

Chesbrough, H. (2006). Open innovation : a new paradigm for understanding industrial innovation. *Open innovation : Researching a new paradigm*, 1-12.

Chesbrough, H. (2007). Why companies should have open business models. *MIT Sloan management review*, *48*(2), 22.

Chesbrough, H. (2013). *Open business models: How to thrive in the new innovation landscape*: Harvard Business Press.

Clapaud, A. (2015). Microsoft fait sa révolution *open source*. En ligne www.journaldunet.com/solutions/dsi/microsoft-fait-sa-revolution-open-source/, consulté le 15 mars 2016.

Crowston, K., Wei, K., Howison, J., & Wiggins, A. (2012). Free/Libre open-source software development: What we know and what we do not know. *ACM Computing Surveys (CSUR), 44*(2), 7.

Deek, F. P., & McHugh, J. A. (2007). Open source: Technology and policy. Cambridge University Press.

Doucet, B. (2015). Fabien Pinckaers (Odoo): "transformer un marché de services en un marché de commodity". En ligne sur le site de Régional-IT, www.regional-it.be/2015/05/28/fabien-pinckaers-odoo-transformer-un-marche-de-services-en-un-marche-de-commodity/

Dunlap, I. (2006). *Open source database driven Web development: a guide for information professionals*: Elsevier.

Eduscol. (2013). Définition de l'Open data, en ligne sur le site du Ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche, www.eduscol.education.fr/cdi/culture-de-l-information/opendata/Opendata-defintion.

Feller, J., & Fitzgerald, B. (2002). *Understanding open source software development*: Addison-Wesley London.

Fisher, F. M., McKie, J. W., & Mancke, R. B. (1983). IBM and the US data processing industry: an economic history. Praeger Publishers.

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

Fleisher, C. S. (2008). Using open source data in developing competitive and marketing intelligence. *European journal of marketing*, *42*(7/8), 852-866.

Free Software Foundation. (2007). GNU General Public License. En ligne sur le site de GNU Operating System, http://www.gnu.org/licenses/gpl-3.0.html.

Free Software Foundation. (2007). GNU Lesser General Public License. En ligne sur le site de GNU Operating System, https://www.gnu.org/licenses/lgpl-3.0.html.

GNU Operating System. (2015). What is copyleft. En ligne https://www.gnu.org/copyleft/, consulté le 23 mars 2016.

GNU Operating System. (2016). Licences. En ligne http://www.gnu.org/licenses/licenses.html, consulté le 20 mars 2016.

GNU Operating System. (2016). What Is Free Software. En ligne https://www.gnu.org/philosophy/free-sw.en.html, consulté le 15 mars 2016.

Goldman, R., & Gabriel, R. P. (2005). *Innovation Happens Elsewhere: Open Source as Business Strategy*: Elsevier Science.

Governor, J., Hinchcliffe, D., & Nickull, D. (2009). Web 2.0 Architectures : What entrepreneurs and information architects need to know. " O'Reilly Media, Inc.".

Groganz, S. (2012). Marketing open source is made for geeks. En ligne sur le site *Open Source*, https://opensource.com/business/12/9/a-complete-guide-marketing-project-business

Hars, A., & Ou, S. (2001). *Working for free? Motivations of participating in open source projects.* Paper presented at the System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on.

Hecker, F. (1999). Setting up shop : The business of open-source software. *IEEE software*, *16*(1), 45.

Himanen, P. (2010). *The hacker ethic*: Random House.

Holmström, B. (1999). Managerial incentive problems: A dynamic perspective.*The Review of Economic Studies*, *66*(1), 169-182.

Jakob, S. F., (2015). Versata saga settled with prejudice. En ligne sur le site de l'Institute for

Jalilvand, M. R., Esfahani, S. S., & Samiei, N. (2011). Electronic word-of-mouth: Challenges and opportunities. *Procedia Computer Science, 3*, 42-46.

Jiang, Z., Chan, J., Tan, B. C., & Chua, W. S. (2010). Effects of interactivity on website involvement and purchase intention. *Journal of the Association for Information Systems, 11*(1), 1.

Johnson-Eilola, J. (2002, October). open source basics: definitions, models, and questions. In Proceedings of the 20th annual international conference on Computer documentation (pp. 79-83). ACM.

Kostakis, V., & Bauwens, M. (2014). *Network society and future scenarios for a collaborative economy*: Palgrave Macmillan.

Krishnamurthy, S. (2005). An analysis of open source business models.

Krishnamurthy, S. (2005). The launching of Mozilla Firefox-A case study in community-led marketing. Online verfügbar unter: http://citeseerx. ist. psu. edu/viewdoc/download.

Krishnamurthy, S. (2009). Case : Mozilla vs. Godzilla—the launch of the Mozilla Firefox browser. Journal of Interactive Marketing, 23(3), 259-271.

Lakhani, K. & Von Hippel, E. (2003), "How Open Source Software Works : Free User-toUser Assistance", Forthcoming in Research Policy.

Lakhani, K. R., & Von Hippel, E. (2003). How open source software works :“free” user-to-user assistance. *Research policy, 32*(6), 923-943.

Laurent, A. M. S. (2004). Understanding open source and free software licensing. " O'Reilly Media, Inc.".

Legal Questions on Free and Open Source Software www.ifross.org/en/artikel/versata-saga-settled-prejudice-1.

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *The journal of industrial economics*, *50*(2), 197-234.

Lerner, J., & Tirole, J. (2004). The economics of technology sharing : Open source and beyond: National Bureau of Economic Research.

Lerner, J., & Tirole, J. (2005). The scope of open source licensing. *Journal of Law, Economics, and Organization, 21*(1), 20-56.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *11*(3), 309-346.

Mustonen, M. (2003). Copyleft—the economics of Linux and other open source software. *Information Economics and Policy*, *15*(1), 99-121.

Net Market Share. (2016). Desktop Browser Market Share. En ligne https://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=0, consulté le 15 juillet 2016.

North Bridge & Black Duck Software. (2015). The Future of open source.

North Bridge & Black Duck Software. (2016). The Tenth Annual Future of open source Survey.

Open Source Initiative. (2004). Apache License, Version 2.0. En ligne https://opensource.org/community, consulté le 26 mars 2016.

Open Source Initiative. (2007). GNU General Public License, version 3 (GPL-3.0). En ligne https://opensource.org/community, consulté le 26 mars 2016.

Open Source Initiative. (2007). The GNU Lesser General Public License, version 3.0 (LGPL-3.0). En ligne https://opensource.org/community, consulté le 26 mars 2016.

Open Source Initiative. (2007). The open source Definition. En ligne https://opensource.org/osd, consulté le 15 mars 2016.

Open Source Initiative. (2012). History of the OSI. En ligne https://opensource.org/history, consulté le 13 mars 2016.

Open Source Initiative. (2016). Community & Collaboration. En ligne https://opensource.org/community, consulté le 17 mars 2016.

Open Source Initiative. (2016). Mozilla Public License 2.0 (MPL-2.0). En ligne https://opensource.org/community, consulté le 26 mars 2016.

Open Source Initiative. (2016). The MIT License (MIT). En ligne https://opensource.org/community, consulté le 26 mars 2016.

Optimus Information. (2015). Open-Source vs. Proprietary Software Pros and Cons.

OSHWA. (2016). Open Source Hardware (OSHW) Déclaration de Principes 1.0. En ligne www.oshwa.org/definition/french/, consulté le 15 juillet 2016.

OSSI. (2016). The Open Source Seed Initiative. En ligne osseeds.org/about/, consulté le 24 juillet 2016.

Phipps, S. (2012). open source community colaboration strategies for the enterprise. En ligne radar.oreilly.com/2012/07/open-source-enterprise-strategies.html, consulté le 19 mars 2016.

Piva, E., Rentocchini, F., & Rossi-Lamastra, C. (2012). Is open source software about innovation ? Collaborations with the open source community and innovation performance of software entrepreneurial ventures. *Journal of Small Business Management, 50*(2), 340-364.

Raymond, E. (1999). The cathedral and the bazaar. *Knowledge, Technology & Policy*, *12*(3), 23-49.

Red Hat. (2016). Services & support. En ligne sur https://www.redhat.com/en/services, consulté le 15 avril 2016.

Riehle, D. (2012). The single-vendor commercial open course business model. *Information Systems and e-Business Management, 10*(1), 5-17.

Rifkin, J. (2014). *The Zero Marginal Cost Society : The Internet of Things, the Collaborative Commons, and the Eclipse of Capitalism*: St. Martin's Press.

Riggins, J. (2015). Will Oracle v. Google Mean The Death of the API Economy? En ligne sur le site de Programmableweb, www.programmableweb.com/news/will-oracle-v.-google-mean-death-api-economy/analysis/2015/06/22.

Rosen, L. (2005). Open source licensing (Vol. 692). Prentice Hall.

Rosenzweig, R. (2006). Can history be open source ? Wikipedia and the future of the past. *The Journal of American History*, *93*(1), 117-146.

Rubens, P., (2015). The open source universe may soon be less collaborative and more litigious. Two case now in the courts could open the legal floodgates. En ligne sur le site de CIO, www.cio.com/article/2893738/open-source-development/how-2-legal-cases-may-decide-the-future-of-open-source-software.html.

Stallman, R. (1985). The GNU manifesto.

Stallman, R. M. (2001). Free software : Freedom and cooperation. New York University.

Stallman, R.M. (2016). Why open source Misses the Point. En ligne http://www.gnu.org/philosophy/open-source-misses-the-point.html, consulté le 20 mars 2016.

Stefanou, C. J. (2014). Adoption of Free/Open Source ERP Software by SMEs *Information Systems for Small and Medium-sized Enterprises* (pp. 157-166) : Springer.

Synopsys. (2015). Coverity Scan open source Report 2014.

Thamsen, L., Gulenko, A., Perscheid, M., Krahn, R., Hirschfeld, R., & Thomas, D. A. (2012). *Orca: A single-language web framework for collaborative development.* Paper presented at the Creating, Connecting and Collaborating through Computing (C5), 2012 10th International Conference on.

Torvalds, L., & Read By-Diamond, D. (2001). *Just for fun: The story of an accidental revolutionary*. Harper Audio.

Trusov, M., Bucklin, R. E., & Pauwels, K. (2009). Effects of word-of-mouth versus traditional marketing : findings from an internet social networking site.Journal of marketing, 73(5), 90-102.

Valloppillil, V. (1998). *Open Source Software : A (New?) Development Methodology*. Document non publié, Microsoft Corporation.

Von Hippel, E. (2001). Learning from open-source software. *MIT Sloan management review, 42*(4), 82-86.

Weber, S. (2004). The success of open source (Vol. 368). Cambridge, MA : Harvard University Press.

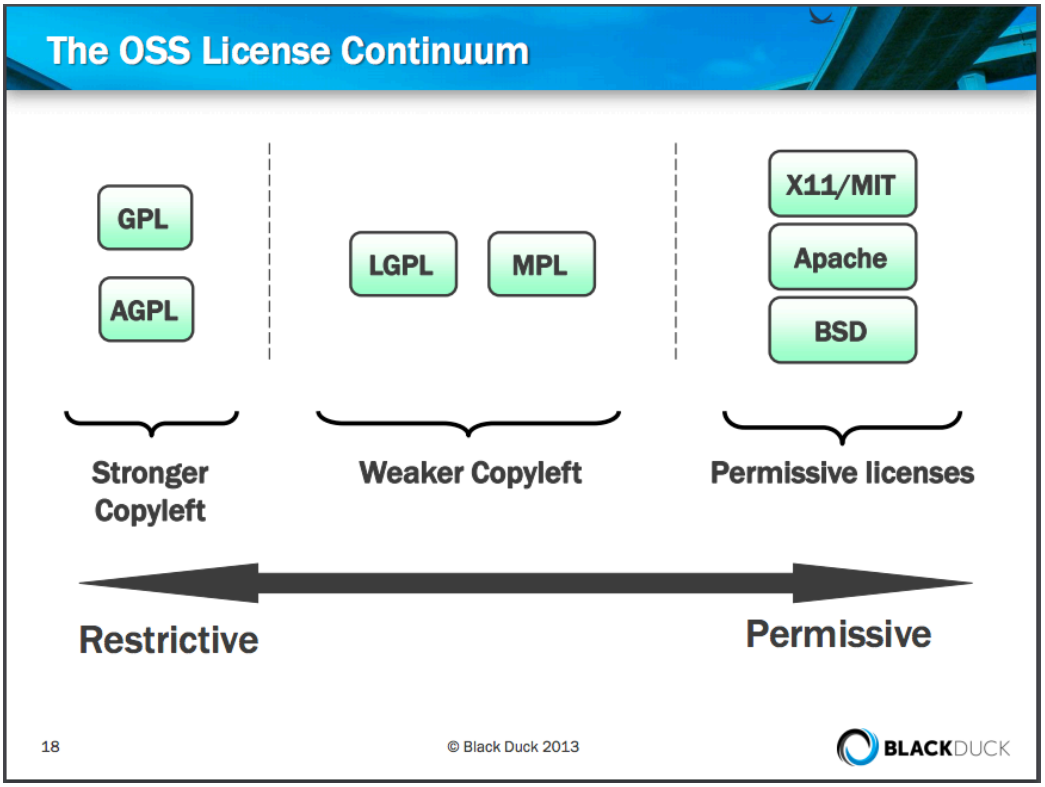
West, J., & Gallagher, S. (2006). Challenges of open innovation : the paradox of firm investment in open-source software. *R&d Management*, *36*(3), 319-331.

West, J., & Gallagher, S. (2006). Challenges of open innovation : the paradox of firm investment in open-source software. *R&d Management, 36*(3), 319-331.

West, J., O'mahony, S. (2008). The role of participation architecture in growing sponsored open source communities. Industry and Innovation, 15(2), 145-168.

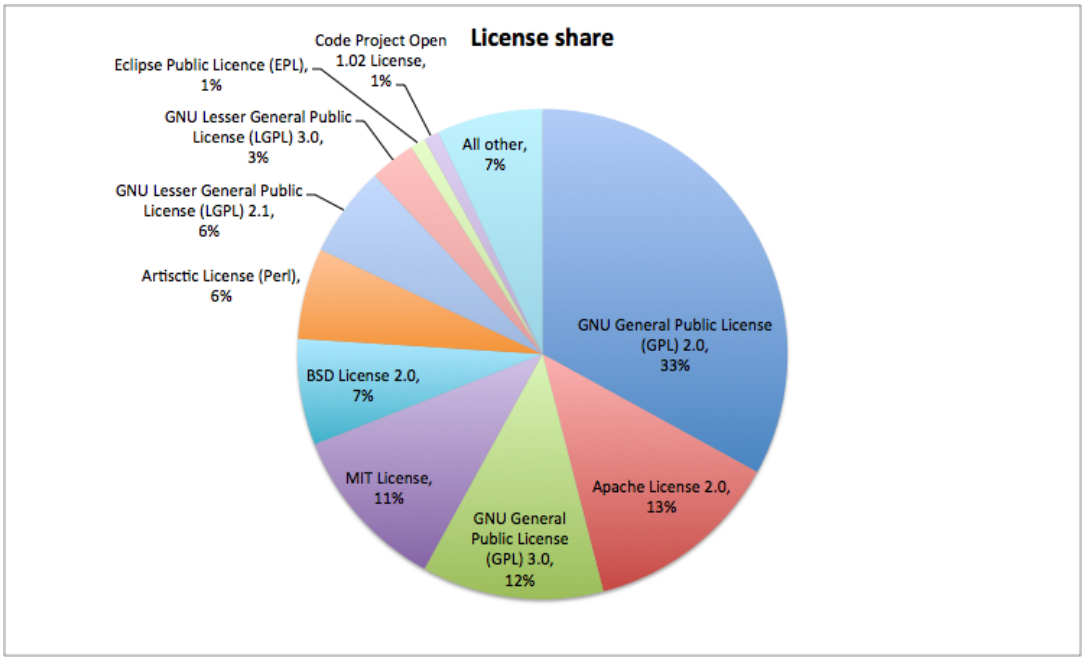
Annexes

1. Classification des licences FOSS les plus répandues



BlackDuck Software. (2013). An Introduction to Open Source Software and Licensing. En ligne https://www.blackducksoftware.com/files/webmedia/_webinars/07-31-13_Intro%20to%20Open%20Source%20Software%20and%20Licenses.pdf

2. Répartition des logiciels par licence



BlackDuck Software. (2013). An Introduction to Open Source Software and Licensing. En ligne https://www.blackducksoftware.com/files/webmedia/_webinars/07-31-13_Intro%20to%20Open%20Source%20Software%20and%20Licenses.pdf

3. Interviews

Les interviews ont été restructurées afin d'éviter les répétitions et ne garder uniquement que les passages pertinents pour l'analyse de ce mémoire. Ceci s'inscrit également dans une démarche écologique afin de ne pas augmenter inutilement le nombre de pages à imprimer

Beauprez, Sven - Co-Founder, Vice-Chairman & Managing Partner, IoTBE (Internet of Things Belgium)

Business models

At the beginning, it was mainly maintenance. Red Hat is a good example. Now a lot of companies have open source at their core but they sell features and more packages to make money. Features could be paying for a complete monitoring of your servers. Red Hat with Linux is again a good example. They sell applications for Linux. A lot of companies go away for traditional maintenance and do closed source on top of their open source core to ensure a revenue stream. It is interesting to keep the core as open source, so it stays free. It keeps the entry barrier very low and give the opportunity to start ups to start their business. At some those companies need more features and applications but it's hard to step away from what they were using, so they will spend a lot of money on the closed packages. This also happening with SaaS and cloud. It's not only with small companies, the same happens with banks for example. They start with open source software and start to buy services and maintenance quite quickly. Open source is a big competition for closed source, especially for database.

SaaS is a new business model. For example, Drupal started as open source but now propose their SaaS Acquia. At first it was hard to make money. You still have a freemium application but as soon as you want to do something extra, you have to pay. They also have a professional SaaS version where they do everything for the company. They maintain everything, code reviews,... through the cloud. It is a second example of how open source is evolving.

There is a third example where Google, Facebook, Apple,... are coming in the picture. Google, Facebook and Twitter are very active in this one. They start to write and try new solutions. Therefore some people will join as it is open source and it will evolve much faster. It is a good leverage since others are writing on top of their work. After, they can use the new versions for their on use. It is very interesting on an artificial intelligence point of view. It is very niche. They collect enormous amount of data. Those data are worked on by so many people and improved so much that they collect excellent version for them and better data. They get the money from selling the data.

Open source v. closed source

In my opinion, you have to look at two things when you go for a software : first it has to do what I want. Second it has to cost as less as possible. After for someone who doesn't know anything about IT, going for oss or closed source would be the same. Closed source might be documented

better but still… For both it will be difficult to learn how to master it and to use it properly. The learning curve is the same in my opinion. So if one is better suited for your use, you should go for it. After if the only difference is one is open and the other is closed. Well, go for the open source, first it will be cheaper to get it. Yet, if you have more knowledge for the closed source, you should go for this one. It is a step by step process.

If you need to start implementing things, you'll need help from outside. It will be far easier and far cheaper to do so with an open source software. It is far more expensive to have closed source software specialist.

On the question about what if companies cease to exist, the answer is quite easy: on the one hand you still have a source code, on the other hand you don't. Open source has a serious advantage here. Some might argue that big company will never stop, there is a good example of very large company going down with Nokia. Microsoft is also a good example, since they've ceased to support many projects many time. On this matter, open source clearly have the upper hand.

Open source is also very useful to solve bugs faster. Since everybody is using it, there are more companies working on it to solve bugs. Hence more secure. The crowd looking at it makes it safer. It is not a few people in an office, it's a whole crowd. For example, CIA wouldn't even be able to change the code without anybody noticing it.

Future

I think we are going more and more towards SaaS and cloud offering. Where you can using for a small fee with extra paying features.

In my opinion, there is no future for fully closed software. But I believe there is a future for hybrid with an open source core and a closed source on top of it. Hybrids will be the future if you are not going towards the cloud.

Security is also a very important matter. People want to know what the code is doing. The banks want to know if it is secure. And it might not sound intuitive for everybody but on that matter open source is better. With the transparency of the code, it is easy to see what's going on and it is way better for the security matter. Security and transparency are very important for a software. To give you an example. When you browse the internet and encounter "http" and "https", the "s" standing for secured. Well, that "s" is made using open source component.

Bostoen, Benjamin - Head of IT & Organisation, AG Real Estate

La partie informatique, que ce soit la partie architecture matérielle ou business, est gérée par AG Insurance. C'est notre pourvoyeur de solution informatique : les machines viennent de chez eux, word, access, mail,... Nous chez AG Real Estate, on s'occupe de ce qui touche à notre métier propre. On a donc besoin d'un logiciel s'occupant d'appel de loyer, un gérant les consommations dans les immeubles,... Ce sont des logiciels tout à fait spécifiques à notre métier. Pour lesquels on fait nos propres choix. C'est donc différent d'un Word commun à toute l'organisation par exemple. Alors, pourquoi on ne prend pas un open office? Aujourd'hui, la réponse est très simple pour nous: la maison mère a décidé d'utiliser microsoft office. Demain, si on se pose la question pourquoi pas de l'*open source*, ce sera principalement basé sur la compatibilité. Est-ce que quelqu'un ayant besoin de fonctions avancées aura ce qu'il lui faut sur un tableur open office? Je pense qu'il y a encore une crainte liée à ça, une incertitude. Ensuite, nous avons les autres logiciels, ceux qu'on développe dans du langage *open source*. Pourquoi prendre un logiciel ayant en base l'aspect *open source*? C'est surtout pour une question de licence. On paie des licences d'utilisation et une maintenance dessus, le fait que ça continue à être développé. Chaque il y a un pourcentage de maintenance à payer même si on ne fait rien. Du coup c'est une charge qui est là pour la société quoiqu'il arrive. Ce qui ne sera pas forcément le cas avec de l'*open source* et qui pourrait du coup être un driver.

On ne fait rien nous-mêmes pour la simple et bonne raison que ne nous sommes pas une grande équipe. On a pas de développeurs en tant que tel et ce n'est pas notre coeur de métier. Par contre, on réalise qu'il y a de plus en plus de sociétés venant présenter leur produit qui se basent sur de l'*open source*. On se rend compte que les solutions de niche - typiquement pour l'immobilier - les solutions sont souvent *open source*, notamment dans la gestion énergétique. Ce sont généralement des solutions tailor made réalisées par de petites équipes qui ont un potentiel énorme mais doivent limiter les coûts de développement. Dès lors l'*open source* s'inscrit parfaitement pour eux.

SaaS et cloud

On commence par une étude en interne, un cahier de charge, qu'on va mettre sur le marché pour voir quels logiciels pourraient répondre à nos besoins. Ensuite, on analyse les solutions disponibles : expérience, viabilité, pérenne,... Donc oui pourquoi pas, à partir du moment où cela répond à nos besoins. Pour nous, comme on a pas le staff derrière, c'est un peu un frein d'utiiliser de l'*open source*. Où sont nos assurances quant au logiciel ? Si c'est une personne avec 100000 clients, il n'y a pas de problèmes, il trouvera un racheteur. Par contre une PME avec cinq clients, c'est un peu plus délicat. Alors oui, on récupère le code mais du coup nous devons peut-être insourcer des développeurs qui auront alors beaucoup de responsabilités et de pouvoir entre les mains. La pérennité de la solution a énormément d'importance dans nos choix. Alors SaaS oui mais avec des assurances.

Le rôle de tout patron est évidemment d'être au courant et up to date. Je pense qu'il faut être capable de réagir rapidement mais de façon sensée. Si demain j'avais carte blanche, je m'interrogerais quand même de savoir si ça a du sens d'avoir les services communs en interne.

Est-ce que j'ai une valeur ajoutée à les avoir chez moi, les gérer moi-même et avoir un contrat me liant avec des compagnies pendant des années ? Typiquement l'email, je pense que ça peut être en dehors. Pour moi les commodités peuvent être externaliser et là oui, le SaaS semble être une très bonne solution. Les coûts de maintenance sont assez importants une fois en interne, dès lors pourquoi ne pas outsourcer ?

Avenir

Il y a une grande problématique autour de la cyber sécurité aujourd'hui. Par exemple, comment sécuriser le cloud ?
Pas toujours évident de faire cohabit de l'hypersécurité avec une collaboration facile, lorsqu'il y a beaucoup de contact avec l'extérieur. Je pense que les gros challenges se passeront dans le domaine de la sécurité. Par rapport aux *open source*, je pense que c'est un outil magnifique pour offrir une base de fondation sans devoir faire appel à de gros investissements. Je pense qu'on se dirige vers une base commune, disons 95%, et puis les 5% supplémentaires se passeraient en interne sans devoir faire appel à une armada de développeurs. Disons que ça ne sert à rien de réinvéter la roue. Celle dont on a besoin existe surement, elle n'a peut-être pas la bonne taille mais elle existe. Ce qui est important c'est du coup pouvoir ajuster la roue à nos besoins. Là, où l'*open source* est très intéressant c'est que c'est une roue qui est vérifiée et retravaillée par énormément de personnes.
Pour en revenir à la sécurité, je ne sais pas si l'*open source* y est vraiment légitime... Alors oui, c'est supposé être plus sûr car vérifié par tout le monde. Mais je ne pense pas, à titre personnel, que je serais rassuré par de l'*open source* pour des transactions boursières. Comment rassurer ses clients par rapport à ça. Est-ce que nous sommes au courant ? Ce ne serait pas surprenant que nos banques en utilisent déjà. La question à se poser en tant que patron, est-ce que ça représente un risque opérationnel ? Surement pas… Réputationnel ? Là c'est plus délicat. Aujourd'hui on ne se pose pas forcément la question, comme il n'y a pas encore eu de précédents. Ou du moins on n'en a pas encore entendu parlé. Par contre les générations suivantes se la poseront peut-être.

De Praetere, Thomas - CEO, Dokeos

En raison d'un problème technique lors de l'enregistrement et par souci d'intégrité, seuls les points notés à la volée seront retranscrits et utilisés dans le cadre de l'analyse.

- L'*open source* n'est plus d'actualité en tant que que business modèle;
- Les arguments philosophiques et communautaires ne sont pas en accord avec la recherche de rentabilité pour une entreprise;
- Direction vers le cloud et le SaaS;
- L'*open source* est élitiste et à tendance à supprimer la concurrence;
- L'*open source* est intéressant pour développer son service, "back office", qui lui sera mis en ligne et permettra de générer du revenu;
- Développement de l'open hardware, open data.

Derveau, Olivier - Consultant en architecture d'entreprise, Service public de Wallonie

Pourquoi de l'*open source*

Pourquoi de l'*open source* dans un organe public et non du propriétaire ? Dans mon cas, on nous a demandé de développer une application d'échange d'informations pour laquelle nous avons reçu de gros subsides. On devait fournir une version pouvant tourner sans aucune contrainte de licences propriétaires. Nous avons donc développé le programme qui était utilisé par la commission et devait pouvoir être utilisé par n'importe quel état membre. Le but était de ne pas imposer de contrainte financière à de plus petits pays ayant des budgets très limités et ne pouvant se permettre d'acheter des licences à IBM, Microsoft ou autre Oracle. Pouvoir fournir quelque chose avec un coût d'entrée nul ou minimum. Outre la raison financière, l'idée était également de ne pas imposer une licence particulière à quelqu'un - également pour des raisons de concurrence déloyale -. Car même si tu fournis le produit gratuitement, tu obliges alors la personne à se procurer une licence avec un certain fournisseur, qu'elle n'apprécie pas forcément. Ou ça également être gênant si on impose une licence Oracle alors que l'état membre travaille actuellement avec IBM, par exemple. L'*open source* offrait donc une solution zéro contrainte.

Les raisons de l'*open source* serait donc selon moi : l'aspect financier et ne pas se lier avec une société privée, garder le contrôle.

Coûts

Sur le long terme, l'*open source* ne coûte pas forcément plus cher. Même s'il faut payer des développeurs pour entretenir le logiciel dans un cas, il faut également payer la maintenance et les licences avec du propriétaire. Je ne saurais pas te donner des chiffres mais je pense sincèrement que les coûts sont plus ou moins semblables. Simplement répartis différemment. De plus, dans ce cas précis, on devait de toute manière développer le logiciel pour l'adapter aux besoins de la commission. Le coût heure de travail aurait été aussi important si on avait utilisé une bibliothèque Oracle plutôt qu'une bibliothèque MySQL.

Continuité

Il y a un budget qui est alloué pour une période de deux fois quatre ans pour assurer le développement et la maintenance du logiciel par une équipe.

Collaboration

Il y avait une volonté de travailler en collaboration avec les états membres pour améliorer le code mais ça ne se passe pas vraiment comme ça. Si on regarde les états membres, ils utilisent le logiciel leur étant fourni sans y prêter plus d'attention. Pour eux, il n'y a pas vraiment de

différence au fait que ce soit *open source*, et il n'y avait pas vraiment d'intérêts de la part des utilisateurs de s'intéresser d'un peu plus près au code. Parfois, notamment en Belgique avec le gouvernement Philippe, l'état fait appel à des experts externes pour modifier et customiser un peu le logiciel pour qu'il réponde parfaitement à leurs besoins.

Présence de l'*open source* dans les organes publics

A la commission c'était généralement du commercial, venant d'Oracle. Pour le gouvernement wallon, il y a une forte volonté d'aller vers de l'*open source*. Il n'y a pas de serveurs payants. Bien sur les supports sont payés. Il y a le décret Open Data qui vient d'être adopté visant à donner accès aux informations publiques.

Chaque gouvernement gère comme bon lui semble sa structure informatique. Toutefois, il y a une volonté de la part des gouvernements flamand et wallon de suivre le fédéral au niveau des outils utilisés. C'est surtout une bonne manière de justifier facilement les coûts.

Carte blanche

Je pense que dans la majorité des domaines, l'*open source* est une solution tout à fait adaptée. Pour moi de l'*open source* par défaut, et éventuellement du propriétaire si cela s'avérait moins cher au final. Je pense notamment aux domaines de niche où il n'y a pas beaucoup d'acteurs. Car même si l'*open source* est "gratuit" à la base, il ne faut pas négliger tous les coûts qui se pointent au fil du temps. Le développement et la maintenance eux coûtent assez chers.

Cloud et SaaS pour les organes publics

Je ne sais pas… Je pense que oui mais il y a la problématique de sécurité et de propriété des données. Par exemple dans le cas de données extrêmement confidentielles, la commission, le gouvernement va essayer d'avoir les données chez eux. Là où elle peut les maitriser. Même la location est importante. Par exemple, je vois mal la commission utiliser un programme hoster par Google ou Amazon si ça n'est pas sur le sol européen. Car ça voudrait dire que le flux de données quitterait l'Europe à un moment donné. Ce qui peut poser questions.

Mangen, Alain – Manager, CSC

CSC utilise depuis longtemps des technologies innovantes pour résoudre les besoins critiques de nos clients. Dans de nombreux cas, l'innovation provient de la communauté *open source*. Dans la dernière décennie, de nombreux projets CSC ont intégré des produits *open source* dans le cadre de nos solutions ou en tant qu'outils utilisés dans des opérations quotidiennes. Avec l'acquisition de plusieurs start-up fondées sur des produits *open source*, CSC a remis l'accent sur l'*open source*. Sous la direction du CTO Dan Hushon, nous positionnons l'*open source* dans le cadre de notre stratégie technologique, en lui donnant l'attention qu'elle mérite et en apportant les avantages de l'*open source* à nos clients.

http://www.csc.com/innovation/insights/112737-csc_and_open_sourcehttp://www.csc.com/innovation/insights/112737-csc_and_open_source
http://www.csc.com/innovation/insights/121305-why_open_sourcehttp://www.csc.com/innovation/insights/121305-why_open_source
http://www.csc.com/innovation/publications/91590/96221-open_source_it_s_not_just_for_software_anymorehttp://www.csc.com/innovation/publications/91590/96221-open_source_it_s_not_just_for_software_anymore
https://leadingedgeforum.com/publication/is-the-open-meme-starting-to-spread-2269/https://leadingedgeforum.com/publication/is-the-open-meme-starting-to-spread-2269/
https://leadingedgeforum.com/publication/is-the-open-meme-starting-to-spread-2269/

Il existe à mon sens 4 niveaux d'utilisation et d'implication dans l'OpenSource, avec une valeur ajoutée croissante :

1. Le premier correspond à un niveau « utilisateur », il est similaire à celui que pourrait avoir un utilisateur particulier lorsqu'il décide de faire usage de l'utilisation libre d'un logiciel, comme par exemple un OpenOffice ou un LibreOffice. Dans le cadre de l'IT, il y a 20 ans, cette utilisation était très marginale et on ne la trouvait souvent que chez quelques rares spécialistes sous la forme d'une librairie GNU ou Cygwin ; aujourd'hui même si cela n'est pas toujours identifié comme de l'OpenSource, cette utilisation est devenue un standard de facto incontournable et au-delà des outils de productivité, la quasi-totalité des projets reposent à un point ou un autre de leur architecture sur de l'OpenSource, qu'il s'agisse de l'infrastructure (serveurs Linux, solutions Cloud, etc), des couches systèmes (Apache, JBoss et autres), applicatives (stacks de développement Java etc) ou de développement (DevOps, IDEs, protocoles, librairies …)
2. Le second est un niveau « intégrateur », qui permet, grâce au caractère « libre » du logiciel, de l'intégrer dans une solution complète, en le modifiant le cas échéant, et en le couplant à d'autres, de réaliser une solution ou une architecture à la taille de l'Entreprise. Cette approche basée sur des composants permet de combiner un Time to Market beaucoup plus rapide qu'une solution complètement sur mesure, tout en conservant l'adaptabilité aux besoins réels du client que l'on ne pourrait avoir avec un produit Off The Shelf.

3. Le troisième est un niveau de « collaborateur » avec la communauté, qui peut être l'évolution logique du deuxième niveau. Lors de l'intégration d'un composant aux besoins réels d'un client, on peut se trouver naturellement confronté à une série de besoins additionnels qui ne sont pas initialement couverts, et que l'on peut soit suggérer à la communauté des développeurs, soit développer soi-même et réinjecter ensuite en proposition d'amélioration dans la communauté. Il est clair que c'est à ce moment que l'effet levier et la collaboration du modèle ouvert prend tout son sens.
4. Le quatrième niveau d' «initiateur » est constitué des sociétés qui, telles RedHat ou d'autres, sont entièrement basées sur le modèle économique de l'OpenSource.

Au niveau de CSC Belgique, non seulement l'utilisation de composants et d'outils OpenSource est devenue monnaie courante dans le cadre des développements DevOps (Niveau 1, avec toutes les stacks de développement Java, par exemple) mais nous avons également eu l'occasion dans le Secteur Public de construire des solutions de bout en bout et de réaliser un reengineering complet d'architectures Entreprise en intégrant des briques OpenSource (Niveau 2). Dans un nombre limité de cas, les adaptations réalisées aux composants utilisés ont pu être réinjectées dans la communauté (Niveau 3). Enfin, au niveau mondial, bien que CSC ne soit pas une société commerciale basée un modèle économique OpenSource, plusieurs initiatives ont été prises pour proposer des projets dans ce mode (Niveau 4).

Motifs

- De manière générale, quelles sont les raisons et motifs invoqués par une entreprise pour effectuer une transition depuis des logiciels propriétaires vers des logiciels *open source* ?

La première raison qui est souvent invoquée en pratique est la réduction des coûts, et la volonté d'utiliser un logiciel gratuit pour éviter de payer des licences.
Il s'agit cependant non seulement :

- d'une mauvaise raison,
- car même si la réduction des coûts est souvent la conséquence de l'utilisation du modèle OpenSource, ce n'en est pas la principale valeur ajoutée
- mais aussi d'une raison « stricto censu » inexacte,
- car cela résulte d'une confusion entre « logiciel gratuit » (« free beer ») et « logiciel libre » (« free speech ») ; même si certaines versions des logiciels OpenSource peuvent être utilisés de manière libre, l'essentiel de l'OpenSource repose sur l'idée de laisser la liberté d'utilisation aux utilisateurs et sur le modèle économique correspondant. Au lieu de faire payer des licences qui ne sont qu'un « droit de passage », parfois à payer plusieurs fois (à l'image des licences « client », « serveur » et « licence d'accès client » pratiquées par certains constructeurs) , le modèle consiste à laisser le logiciel libre d'accès et d'utilisation mais de proposer un réel support aux organisations qui en ont le besoin.

Au jour d'aujourd'hui, l'OpenSource n'est plus « seulement » une alternative « bon marché » que les sociétés et les clients recherchent pour diminuer leurs coûts, et de nombreuses organisations tant privées que publiques (gouvernements français, anglais etc) placent au contraire l'OpenSource comme le premier type de solution à envisager, pour de nombreuses autres raisons et motifs plus réalistes qui sont également évoqués ci-dessous en réponse à la question « Performance ».

- Est-ce que cette transition nait de consultations internes ou de consultants extérieurs ?

Les deux sont possibles et se produisent en pratique : CSC Belgique a par exemple un Centre de Compétences OpenSource et promeut activement ce type de solutions chez plusieurs clients, mais nous recevons aussi parfois la demande directe de clients.

- N'est-il pas plus simple de rester sur des logiciels propriétaires ?

Les seuls éléments qui pourraient paraître plus simples, sont
1/ que le recours aux logiciels propriétaires permet en apparence tout au moins, de reporter une partie de la responsabilité du projet sur d'autres fournisseurs. Il s'agit cependant d'un faux débat, car même dans ce cas la qualité globale de la solution fournie au client repose sur le fournisseur « prime contractor». A l'inverse, on constate en pratique que le recours à un logiciel propriétaire rajoute une dépendance par rapport à un fournisseur externe, et l'arrêt du support, des mises à jour ou même de la fourniture du logiciel propriétaire par le fabricant peut réellement mettre tant le client que le fournisseur principal dans un embarras parfois extrêmement coûteux. Une situation similaire avec un logiciel « OpenSource » permet alors de s'en sortir en effectuant soi-même la maintenance du logiciel – nous avons déjà rencontré ce cas de figures chez l'un de nos principaux clients Secteur Public.
2/ que par le passé le logiciel libre était assimilé aux outils « shareware » et « freeware » par opposition aux grosses solutions commerciales. On avait communément coutume de dire il y a 20 ans,que quiconque choisissait certains grands constructeurs, ne prenait peut-être pas la meilleure solution mais qu'en tous cas on ne pourrait pas le lui reprocher ; c'est peut-être encore le cas aujourd'hui avec certains poids lourds du monde commercial, mais aujourd'hui la qualité réelle de la solution apportée prime de plus en plus, et l'apparition de sociétés mondiales réalisant la totalité de leur chiffre d'affaires sur le modèle OpenSource, comme RedHat, ont changé cette perception du monde. Les plus importantes sociétés comme Google, Facebook ou Amazon n'existeraient pas sans l'OpenSource, et sans les contributions de Linux, Apache, Java, Perl, Python, MySQL et d'autres, l'écosystème IT ne serait pas celui que nous connaissons aujourd'hui. Même des sociétés traditionnellement orientées vers les logiciels propriétaires comme Microsoft, se tournent de plus en plus vers l'OpenSource, et Microsoft Azure tourne à 30% sur Linux (http://www.zdnet.com/article/microsoft-nearly-one-in-three-azure-virtual-machines-now-are-running-linux/). A l'heure actuelle, certains domaines entiers comme le « Big Data », sont même entièrement construits et développés autours de solutions initiales OpenSource telles qu'Hadoop. Les études des analystes tels que Gartner mettent de plus en plus en avance des solutions OpenSource dont la maturité est donc éprouvée.
En revanche (mais ceci est vrai également en ce qui concerne les logiciels propriétaires) il est important de comprendre les différences entre les types de licenses proposés.

Performance

- Quel est l'impact d'une transition *open source* sur les performances de l'organisation ? Il y a-t-il une réelle différence par rapport aux logiciels propriétaires ?

L'aspect Performance est évidemment étroitement lié au logiciel considéré, qu'il soit commercial ou OpenSource ; mais globalement l'utilisation du modèle OpenSource permet :
- une agilité supérieure (portabilité, adaptabilité liée à la disponibilité du code source, à l'absence de dépendance etc),
- une intégrabilité et une interopérabilité meilleure (par le respect de standards ouverts),
- une rapidité vers le marché plus élevée (intégration de composants modifiables, « building blocks » ou composants remplaçables dans une architecture « loosely coupled »),
- une meilleure compétitivité, une absence de monopole et une indépendance vis-à-vis des fournisseurs (alternatives plus nombreuses, marché libre, diversité technologique),
- une meilleure réutilisabilité bi-directionnelle entre les projets et la communauté,
- une meilleure qualité (effet levier de la communauté),
- une meilleure sécurité du code (auditabilité).

Un autre élément important est l'évolutivité des solutions, le fait de pouvoir disposer de composants « à l'essai » et « gratuits » pour une utilisation ponctuelle permettant aux intégrateurs qu'aux clients de réaliser des études de faisabilité et des « Proofs of Concept » à moindre coût, avec une prise de décision et l'acquisition de versions supérieures et/ou d'un contrat de support ultérieurement et en toute connaissance de cause. De ce point de vue, l'OpenSource peut donc aussi jouer un rôle d'initiateur de solutions ou de projets.

Coût

- Bien que le coût d'acquisition du logiciel soit inférieur, est-ce que cela ne revient pas à plus cher dans le long terme ? Les coûts de formations ne sont-ils pas plus élevés ou l'appel à des experts externes ?

Comme expliqué plus haut, le coût d'acquisition n'est pas le premier avantage et est souvent une mauvaise ou une fausse raison de recourir au modèle libre. Le réel coût à considérer doit être global, horizontalement à travers tous les départements ou métiers du client et verticalement à travers tous les niveaux impliqués depuis l'application jusqu'aux couches d'infrastructure (cfr « Total Cost of Owernship »), et ceci pendant tout le cycle de vie de la solution, depuis la conception et la réalisation jusqu'à l'exploitation et la maintenance.
En ce qui concerne la formation, à l'inverse les supports de formation (documentation collaborative, traductions dans la communauté, vidéos Youtube, sites wiki etc) sont souvent aujourd'hui de qualité équivalente si pas supérieure à ceux fournis avec les logiciels propriétaires ; et au contraire le coût d'utilisation de spécialistes pointus formés à des solutions propriétaires et parfois devenues entretemps obsolètes, peut quant à lui devenir très vite prohibitif. Au niveau académique, la démarche est d'ailleurs largement connue et active depuis de nombreuses années, et les secteurs commerciaux (public ou privé) suivent de plus en plus.

- Est-il dès lors financièrement intéressant d'effectuer une telle transition ? Quelle est la durée d'amortissement des investissements ?

Il est bien entendu essentiel d'effectuer une analyse de rendement au cas par cas, et prétendre qu'une catégorie de logiciels constitue la panacée serait une erreur. Il s'agit essentiellement de

modèles économiques différents dont les avantages (et dépendances éventuelles) doivent être analysées en fonction du contexte particulier de chaque client. Néanmoins, au niveau de CSC Belgique, nous avons par exemple eu l'occasion de réaliser un projet de reengineering pour un client important du Secteur Public, qui nous a permis de réduire le coût d'une infrastructure nationale de 60 à 30 M€ (investissement sur 5 ans), pour un projet qui a duré moins de 2 ans et coûté 2 M€. Il est clair qu'il s'agit là d'un bel exemple de retour sur investissement.

- L'amélioration des performances compense-t-elle les investissements ?

A nouveau cela doit être analysé au cas par cas, mais il n'est pas irréaliste de penser qu'il est possible à la fois de diminuer les investissements et d'augmenter les fonctionnalités et la performance. Le cas cité ci-dessus en est un bon exemple, non seulement les coûts récurrents du client ont été notablement réduits, mais il a également été possible d'offrir certaines nouvelles fonctionnalités majeures « gratuitement ».

Pasture, Mathieu - Directeur technique, Babelway

Traditionnellement, soit on vend un logiciel. Qu'il soit en *open source* ou pas. Soit on fournit un service, l'entièreté du service aux clients. Les deux modèles existent et ont des désavantages. Le modèle du software est très cher. Très flexible car on peut faire exactement ce qu'on veut. Mais très cher car il demande des spécialistes en interne, une infrastructure et il faut le maintenir. Il faut donc toute une équipe qui est capable de s'en occuper. Donc très cher mais très flexible. A l'opposée, le système où on outsource tout est beaucoup moins cher car il n'y a pas d'investissement. Par contre on a perdu la flexibilité. On n'a donc plus aucun contrôle car celui-ci est pris par la société intermédiaire. Face à cette situation, quand on a décidé de créer notre société, on a décidé de créer un produit dans lequel on a le meilleur des deux mondes. La flexibilité d'un côté, comme une plateforme partagée. Et en même temps on garde le contrôle comme un software. C'est pour ça qu'on s'est très rapidement dirigé vers un modèle en SaaS. Software as a service. Donc à la place d'acquérir un software, qu'il soit *open source* ou non. Tu loues un service sur l'outil internet, donc dans le cloud. Par exemple, l'utilisation de Gmail pour tes emails. C'est géré pour toi et tu ne dois pas t'en occuper. Pour nous, notre positionnement a été naturellement d'offrir le service. Juste offrir le software c'est pas assez. C'est trop difficile pour les gens. Ils n'ont pas la capacité de mettre en oeuvre ce software. Du moins la plupart des gens. Ce n'est pas leur vocation et n'ont pas d'intérêt à le faire. C'est donc pour ça qu'on est parti sur le modèle SaaS. On offre un mix entre un logiciel, du code que nous avons écrit en partie qui tourne sur une infrastructure. Et on offre du service, qui non seulement permet au code de tourner mais également à l'utilisateur de l'utiliser. Du support sur cette plateforme. Et on peut même offrir du full service. Voilà pour Babelway en quelques mots.

Open source et propriétaires

Donc le software n'est pas en *open source* parce qu'on pense que ça n'a aucun avantage pour nous. Aucun avantage d'un point de vue viabilité de l'entreprise. Je pense que ça n'aurait pas beaucoup de désavantages mais ça n'a pas d'avantages. On y a réfléchi au début : veut-on se poser comme *open source* ou non? Notre produit n'est pas un logiciel, c'est un service. Dès lors les clients ne sont pas intéressés à avoir le produit. Certains seront intéressés par certaines questions auxquelles l'*open source* répond, par exemple la continuité. Le fait que si l'entreprise fait faillite, ils n'ont pas accès au code et ne peuvent le faire tourner eux-mêmes. On a prévu un système leur permettant de récupérer le code en cas de problèmes. Ce n'est pas une licence *open source* mais contractuelle.

Par contre on utilise beaucoup de librairies *open source* dans notre software. Plutôt pour des couches d'infrastructure. Pas dans le service qu'on rend mais dans les briques de base. Il me semble que mis à part deux exceptions anecdotiques, on n'utilise pas de software qui ne soient pas *open source*. En fait tout le software que nous n'avons pas écrit est *open source*. On utile un OS *open source* pour nos serveurs (linux), une base de données *open source*, apache,... On utilise beaucoup d'*open source* mais on écrit beaucoup de codes aussi.

Je pense que le propriétaire n'existe même pas pour les librairies et je ne vois pas d'alternatives à Linux… Windows ? Mais ça ne me viendrait à l'idée d'utiliser windows… Je crois que l'offre

open source est la mieux adaptée pour nos besoins, elle est meilleure. Sans parler du prix, elle est juste meilleure. Elle est de très bonne qualité, existe depuis longtemps. Pour nous qui sommes des développeurs, on a besoin de comprendre ce qu'il se passe. Du coup avoir accès au code est très important. C'est très difficile d'utiliser une librairie sans avoir accès à son code.
On contribue aussi à l'*open source*. On respecte les licences, ça veut dire qu'on va plutôt favoriser des projets avec une licence flexible. Toutes nos modifications et corrections sont republiées.

Il y a différentes catégories de logiciels :
- le communautaire : linux, apache, et beaucoup de librairies de très bas niveau. Qui sont géniales. Elles sont trop couteuses pour être faites par une seule personne. Avec une bonne libraire *open source*, les concurrents disparaissent. Il n'y a pas de raison d'avoir un autre développeur http qu'apache par exemplee Postgres.
- Les OSS qui appartiennent à une société, mono société. MySQL.
- Complètement closed source. Oracle.
Donc nous avons trois bases de données qui ont la même fonction. Certaines vont être plus fortes dans certains domaines.
Le problème de l'*open source* monosociété, ça veut dire qu'il n'y a qu'une seule entité qui finance cet *open source*. Ce qui peut poser un risque. Par exemple, MySQL a été racheté par Oracle. Qui du coup n'est plus devenu *open source*. Les monosociétés ont souvent un service beaucoup plus pointu, contrairement aux librairies qui ont un but plus général. Elles ont également un avantage concurrentiel par rapport à la distribution de leur logiciel. C'est important que ce soit *open source* pour des dévelopeurs. Comme je te l'ai dit, pour quelqu'un qui utilise le code, il faut y avoir accès. Par contre, tu ne peux pas vendre ton logiciel mais tu peux vendre les services. C'est un modèle qui est difficile pour générer du revenu. On peut prendre l'exemple d'Odoo qui est passé en SaaS.

Choix du SaaS

Pour fournir un service, un logiciel pointu, je pense que le modèle SaaS est le meilleur actuellement. Ca prend beaucoup d'énergie et de temps pour avoir un bon logiciel et le faire tourner. Les gens veulent quelque chose qui tourne et est fiable. Le SaaS offre une économie d'échelle bien plus grande qu'un OSS. A la base on voyait également les OSS comme une économie d'échelles avec le partage de développement entre plusieurs sociétés. Par contre, c'est un avantage que tu perds une fois que tu es en *open source* mono société. Donc au final, que les sources soient ouvertes ou fermées, ça ne change pas beaucoup de choses. Par contre le SaaS offre un avantage, car on a des serveurs sur lesquels les logiciels tournent avec des gens qui y regardent et vérifient que ça tourne bien. Si problème, on intervient tout de suite et fait l'update. Les maintenances sont automatiquement gérées par la société distribuant le software. L'utilisateur a donc un package qui est prêt à l'emploi. Pour les applications, je pense que SaaS est mieux. Pour les librairies, des morceaux d'applications, je pense que l'*open source* est ce qu'il y a de mieux.

Avenir

On remarque que dans tous nos devices, on est dans des milieux fermés et on remarque de plus en plus que les gens qui nous fournissent des logiciels essaient de nous locker. Par exemple android, Google controle ce qu'il se passe via le playstore. Tu ne peux pas installer toi-même des applications. Apple fait très bien ça aussi. Là, je pense que l'*open source* à un rôle à jouer et une place à prendre. Mais selon moi, de plus en plus on se dirige vers un bundle software + service payant.
Je pense que le logiciel *open source* monosociété va disparaître car économiquement il est moins bien que le logiciel en Saas ou closed source. L'*open source* communautaire est ce qu'il y a de mieux et de plus efficace mais n'est pas forcément celui qui innove le plus. Car pour innover rapidement, il faut des ressources financières et une direction stricte. Par contre pour fournir un produit de qualité et stable, c'est ce qu'il y a de mieux. Par contre, il y a une grande compétition dans ces logiciels pour les ressources. Pour intéresser quelqu'un, que ce soit une société ou un privé.
Je ne pense pas que le closed source soit voué à disparaître.

Un des problèmes du SaaS, c'est que tu vends un abonnements. Ca coute très cher de développer un système, surtout en capital humain. Par contre les revenus sont très lents à arriver. SaaS est un système difficile à démarrer car il faut investir beaucoup. Contrairement à de la consultance informatique par exemple.

Pinckaers, Fabien - CEO, Odoo

Pourquoi l'*open source*

Premièrement, le choix de l'*open source* était basé sur des choix idéologiques. A l'heure actuelle je trouve que c'est un super business modèle. Mais je ne le conseillerais à personne. Ca ne fait qu'un an et demi que nous avons découvert que c'était un super business modèle. Nous avons énormément souffert pour y arriver. L'*open source* est difficile et peut être très dangereux car c'est un marché hyper transparent. Les acheteurs peuvent tester le produit avant de l'acheter et il est dès lors très dur d'avoir une technique de différenciation. Contrairement à des logiciels propriétaires. Ici les gens peuvent tout tester et dès lors utiliser le meilleur, comme c'est moins cher.

Donc l'*open source* est hyper dur car si vous n'êtes pas le premier, vous n'êtes rien du tout. C'est un milieu très élitiste ne laissant pas place à la concurrence. Premier n'est peut-être pas le bon terme car par exemple dans notre cas, on est arrivé un peu tard. Mais à un moment donné il n'en reste plus qu'un car il y a un effet de consolidation où tous les utilisateurs vont sur le meilleur. Comme plus on a d'utilisateurs, plus on devient grand, il y a une espèce de cercle vertueux, ce qui fait qu'on grandit très bien. Par contre, tous les autres sont au bord de la faillite. Par exemple dans notre secteur, il y a en a encore qui tiennent mais il ne devrait plus tenir longtemps. On peut rentrer dans l'*open source* si on va être le meilleur de son secteur. Sinon c'est vraiment dangereux. Toutefois l'*open source* offre une entrée rapide sur le marché et une très grande visibilité. C'est aussi très utile lorsqu'il faut être très innovant. L'*open source*, selon moi, est l'équivalent d'une très bonne offre freemium. Maintenant, tout le monde fait du SaaS, on pourrait voir l'*open source* comme l'offre freemium. C'est une manière pratique et facile d'attirer des utilisateurs gratuits. Après, le tout est d'arriver à retenir ces personnes-là pour leur proposer les services.

Il y a bien évidemment moyen de s'en sortir avec du closed source, il y en a qui y arrive très bien. Cependant, pour notre modèle et la manière dont on a voulu le faire, je ne pense pas qu'on aurait pu y arriver ou être meilleur avec du closed source, loin de là. Mais on a galéré. Il a fallu devenir le meilleur avant que ça commence à marcher et que les engrenages se mettent bien ensemble. Pour nous, ce qui a vraiment fonctionné, c'est d'utiliser l'*open source* comme un modèle de développement et pas comme un business modèle. On est sur un business modèle traditionnel qui est même plutôt propriétaire. On a deux produits, Odoo Community et Odoo Enterprise. La majorité est sur Community et 20% sur Enterprise. Community est *open source* et Enterprise propriétaire. On vend les fonctionnalités.

Marketing

On n'a pas beaucoup investi en communication mais en développement produit pour satisfaire les utilisateurs. Tout le reste à suivi. On a énormément profité du bouche à oreille. Les utilisateurs qui sont contents en parlent directement autour d'eux et ça va assez vite.

Avenir

Dans le cas d'Odoo, il ne devrait pas trop y avoir de problème. Sauf erreur de notre part, notre produit et notre offre nous permettent de voir le futur assez sereinement. Le marché est énorme est et en forte croissance. Il n'y a aucune cause externe qui devrait nous poser des difficultés. La diffculté c'est au niveau de l'exécution. Si on ne devient pas beaucoup plus grand dans les années à venir, ce sera de notre faute.

On remarque que le SaaS remarque de plus en plus l'*open source*. De plus en plus les gens se retournent vers le SaaS par simplicité et du coup, on observe de plus en plus d'acteurs, comme nous, faisant de l'*open source* et du SaaS. Par exemple, nous faisons 50% de chaque et dans notre cas ça devrait rester comme ça. C'est propre à nous mais nous proposons le SaaS aux petites entreprises et l'*open source* aux plus grandes. La principale raison est que les plus grandes ont besoin de services et d'intégrateurs pour avoir du sur mesure. Tandis que les plus petites utilisent le produit plutôt de façon standard et là c'est du SaaS.

Schröder, Bruno - Technology Officer BeLux, Microsoft

Position de Microsoft

Le gros désaccord qu'il y a eu entre microsoft et l'*open source* est que les partisans de l'*open source* voulaient faire passer des lois interdisant microsoft. C'est comme ça que ça a commencé. Il y a eu des propositions de lois qui ont été votées interdisant au secteur public d'interdire l'achat de logiciel d'origine commerciale. Ce à quoi Microsoft a forcément mal réagi. Ca a très vite dégénéré en guerre d'idées. Cependant ça n'avait pas grand chose à voir avec la technologie *open source* elle-même. Il y a eu une série d'affirmations du côté *open source* qui n'étaient pas forcément vraies. On a vu par exemple, qu'un modèle n'était pas plus sûr que l'autre.

Ce qui a changé, c'est qu'en matière d'informatique on a un avantage assez intéressant par rapport aux autres secteurs, la puissance de calcul augmente de façon considérable avec le temps. La loi de Moore. On est passé il y a environ cinq ans d'une situation où l'on devait optimiser tous les composants à une situation où on a assez de puissance de calcul pour intégrer n'importe quoi. Avant ce point de basculement là, la stratégie qu'on voyait chez les fournisseurs de logiciels était une stratégie de plateforme intégrée. Tous les produits viennent alors du même environnement. C'était également le cas dans le monde de l'*open source* où il y avait une sorte de guerre de distributions entre Debian, Ubuntu, Red Hat et quelques autres. On n'avait pas assez de capacité pour pouvoir faire quelque chose d'assez générique. Le cloud a complètement changé tout ça. Le cloud a été possible avec le changement de puissance de calcul. Maintenant, on peut se permettre d'intégrer des technologies extremement disparate en arrivant à les faire fonctionner ensemble de manière correcte.

Du coup à partir de ce moment là, en ce qui concerne Microsoft, on s'en fout complètement de la machine qu'il y a derrière. A partir du moment où on peut intégrer et faire fonctionner les différentes composantes de façon à ce qu'un outil informatique standard puisse l'utiliser, il ne faut plus se préoccuper de ce qui se cache derrière. Je dirais que ça a été le grand déterminant technologique qui a permis la transition. On avait donc mis en un interne toute une série d'actions. On a mis directement sur Azur la possibilité de pouvoir faire tourner des machines virtuelles *open source*.

L'autre évènement déterminant à été, chez nous, le changement de patron. On est passé de Steve Ballmer à Satya Nadella. On est en gros passé d'un vendeur à un ingénieur. Le vendeur était concentré sur la puissance de la marque et tout ses composants. Il était là depuis un certain temps et venait de ce monde plateforme. On est passé un patron ingénieur qui avait une manière assez différente de voir le développement de Microsoft et l'innovation. Lui n'a absolument aucun problème à utiliser des technologies pour ce qu'elles font. A côté de l'ouverture de la plateforme Microsoft à l'utilisation de technologie *open source*, on a eu le deuxième changement en interne qui a été l'utilisation en interne par Microsoft de technologies *open source*. On avait donc toute une série de projets chez nous visant à faire plus ou moins ce que faisaient certains produits *open source*, c'est projets ont été arretés. Cela ne servait à rien de mettre du travail dans quelque chose qui avait déjà été fait, si on savait intégré ces produits dans nos plateforme autant les utilisés pour ce qu'ils font.

Le fait d'être à même de pouvoir faire tourner des technologies dans un environnement, sans que celui soit intégré, a vraiment été le déclencher. Avec le changement de patron et de perspectives. Le fait que les approches n'ont plus beaucoup de sens maintenant explique cela aussi. On se retrouve maintenant avec ce cloud où tout le monde se connecte avec n'importe quoi et il faut s'adapter à cet environnement hétérogène. Je pense que ces environnements hétérogènes seront là pour au moin les 20-25 années à venir.

Open source v. closed source

On se dirige également vers une nouvelle philosophie d'applications. Avant c'était un peu comme une grosse boîte à outils dans laquelle l'utilisateur prenait les choses dont il avait besoin. De nouveau c'était lié à un problème d'optimisation des différents outils car il était assez difficile d'intégrer des choses avec des philosophies technologiques différentes. Maintenant, dans le monde du smartphone, l'application permet de réaliser une chose mais du début jusqu'à la fin. On est passé d'une philosophie globale à quelque chose de plus précis. Dans ce monde là, la capacité à mettre rapidement des produits et services sur le marché est beaucoup plus critique que le fait de proposer quelque chose qui va tenir la route quelque soit l'usage que l'utilisateur en fait. Par exemple, Word et Excel c'est de la boite à outils. On peut y faire des milliers et des milliers de choses différentes. Comme pouvoir être sur le marché rapidement est bien plus important aujourd'hui, on prend ce qui est déjà disponible et fonctionne comme ça il ne faut pas le retravailler. On intègre ce qu'il faut et on est beaucoup plus rapidement sur le marché avec un produit qui fonctionne. Alors que ce soit de l'*open source* ou du closed source n'a pas beaucoup d'importance. Ce qu'on voit dans le monde de l'*open source*, c'est qu'il y a une bonne base de bibliothèques de culture générale en matière de technologies. Les choses que tout le monde comprend et qui sont les briques de base, on les trouve facilement dans l'*open source*. Quand on veut faire quelque chose qui est beaucoup plus particulier, en général il faut le développer sois-même. Alors là, c'est la question du business modèle. Est-ce que je mets mon code en *open source* pour bénéficier de l'effet de communauté qui va m'aider à travailler dessus ou bien est-ce que je le garde en modèle propriétaire car il y a quand même des choses dans l'algorithme qui font que je dois quand même le protéger car c'est ça qui va me permettre de tenir la distance face aux concurrents. Exemple, on sait que Google, Facebook, Instagram,.. Utilisent de l'*open source* de tout part mais pourtant le code final n'est pas *open source*. Même si Google a mis Android en *open source*. On retrouve donc un mélange des deux en fonction de ce qu'on veut faire et du type de produit que l'on veut mettre dans la nature. Les logiques ont un basculé par rapport au monde d'avant.

Microsoft a mis à disposition de pleins de gens des outils qui n'étaient disponibles que pour ses chercheurs. De nombreuses librairies ont été mises en ligne.

On est passé d'un modèle commercial assez formaté à un modèle où on est constamment en contact avec les clients, où on suit leur projet, on a des mises à jour journalières. On est basé d'un modèle de vente très transactionnel à une vente un peu plus de proximité. C'est sans nul doute une conséquence du passage sur le cloud. Ce qui a surement un impact très positif sur le résultat.

Une des raisons pour lesquelles l'*open source* est aussi bien intégré maintenant est que le modèle financier qui est derrière est un modèle de consommation. On va payer à la minute d'utilisation. On ne vend plus des licences, là où on avait fait une grande différence entre le monde *open source* et le monde commercial. Maintenant on se retrouve dans des modèles assez équivalents.

Senterre, Marc - ICT executive, Prayon

Open source v. closed source

Nous utilisons des programmes propriétaires. Pour une société industrielle avec des objectifs de résultats et qui doit tourner tout le temps, le risque avec l'*open source* c'est le support. Il n'y a aucun engagement pour corriger les bugs. Ca se fait de manière spontanée et au mieux possible par la communauté mais il n'y a pas de garantie. Avec les logiciels propriétaires, nous signons à chaque fois un contrat de maintenance où l'éditeur s'engage à faire du support. C'est la raison principale. L'*open source* en temps que tel est intéressant car il permet des solutions rapides avec un écosystème très performant car tout le monde développe et met à jour. Tout existe facilement en *open source* excepté l'engagement sur le résultat. C'est un problème car lorsque tout va bien tout va bien mais lorsqu'il y a un bug il faut une solution immédiate. Alors oui ça coûte moins cher mais est-ce que c'est risque que nous sommes prêts à prendre, ça c'est autre chose. Je pense que c'est une problématique qui est présente dans beaucoup de société comme la notre.

Alors oui, de l'*open source* accompagné d'un service là c'est différent. Généralement ils ne fournissent pas la dernière version mais la dernière qui a été vérifiée et stabilisée Dans ce cas ci il y a un contrat de support et de maintenance. Ce genre de solutions, je n'aurais aucun problème à les utilisées. Mais dans des domaines précis. Dans des domaines d'innovation ou de niche, l'*open source* avec cette couche de gestion est acceptable. Je ne ferai pas de l'*open source* pour remplacer du business critical, pour remplacer SAP, les mails,... Les applications qui sont critiques pour le métier. Si je prends de l'*open source* et qu'il se plante, c'est délicat. Tandis qu'avec SAP, j'ai un contrat de maintenance qui interviendra très rapidement. Pour les domaines de niches et l'innovation, des solutions très spécifiques, là je pense que l'*open source* avec les services sont meilleurs. Mais pour le business critical, je ne pense pas. Je pense que l'*open source* peut être très adapté aux start-ups. Personnellement, je pense que le support pose problème avec l'*open source*.

Il est vrai que nous sommes peut-être un peu conservateur et ne prenons peut-être pas assez de risques mais je pense que c'est une certaine vérité des grosses entreprises. De plus, à supposer que SAP vienne à cesser ses activités, il y a une clause dans le contrat indiquant que nous pouvons récupérer le code source. C'est généralement le cas avec tous les gros éditeurs. Egalement avec Microsoft par exemple.

Lorsqu'on l'on opte pour un logiciel, pour nous, il faut qu'il soit standard et qu'il réponde aux besoins de nos clients. S'en suit une série de critères techniques. On évite un maximum quelque chose qui est non standard. On se dit que pour nos besoins, il doit y avoir des solutions sur le marché car nous ne sommes pas différents des autres. Ce qui convient le mieux à notre environnement.

Saas & Cloud

Nous sommes en train d'effectuer notre transformation digitale avec un shift métodique, application par application, vers le cloud. Nous utilisons aussi bien du cloud public que du cloud propriétaire. Pour le SaaS, nous avons des applications marketing et en customer relationships.

Coûts

Alors il est sûr que la différence entre notre SAP et de l'*open source* est gigantissime Idéalement il nous faudrait des développeurs internes mais ce ne serait pas le cas, du coup on ferait tout via la communauté et ça nous reviendrait à prix minimum. Mais la différence est énorme, les prix de maintenance d'une société comme SAP sont exorbitants. Même la différence avec l'*open source* / service serait importante. L'*open source* est toujours moins cher.

Futur

Il y aura toujours un gap entre open et closed source. Comme je l'ai dit, je pense que pour les solutions innovantes, l'*open source* sera toujours utilisé. Après je pense que le propriétaire aura toujours sa place. A partir du moment où vous avez des engagements à tenir envers les actionnaires et créanciers, vous prenez moins en moins de risques.

Viseur, Robert - Expert en Management et Technologies *Open Source*, CETIC & Faculté Polytechnique de Mons

MODELES

Première chose, l'*open source* on peut en produire et en vivre. Il y a toute une série d'entreprise qui produisent de l'*open source* et après développe un business autour des composantes de celui-ci. Par exemple, Alfresco qui fait de l'édition de logiciel *open source* et puis on a un modèle double licence avec une version communautaire et une version commerciale payante avec des licences qui contient des fonctionnalités en plus. Ils ont également un réseau de partenaires qui assure du service sur les installations et qui rétribuent Alfresco pour des statuts de partenaires privilégiés. Premier modèle donc : on produit de l'*open source* pour soit des licences, soit des services sur le produit.

Ensuite, il y a les modèles d'usage. On utilise de l'*open source* dans des développements informatiques. Aujourd'hui, pratiquement 3/4 des entreprises utilisent de l'*open source* dans leur développement. Ca ne veut pas dire qu'elles en produisent. Mais dans les développements, elles en utilisent.

Il y a un modèle qui est plus de grosses boites. Les modèles de mutualisation, des gros acteurs vont s'associer pour développer des logiciels qui ne sont pas différenciants. Ce que j'appelle des commodités. L'idée est de standardiser un maximum des produits. C'est la production d'une base technologique commune de façon à baisser les coûts. On y retrouve plus des boites de statures internationales concurrentes sur différents marchés mais qui s'associent pour des produits technologiques qui seront intégrés dans des produits finaux.

TENDANCES

Les environnements de développements propriétaires, globalement, sont morts. Si on regarde aujourd'hui, on a un usage très large des outils de développements *open source*. Même Microsoft passe sur des outils *open source* malgré plusieurs années de défenses du modèle commercial.

Un autre point, est la vision de l'*open source* décrite avant 2006: un mouvement essentiellement communautaire et idéologique. Cette philosophie existe toujours. Mais depuis, on a des grosses boites qui investissent massivement dans l'*open source*. En plus du tissu de PME. Ces personnes là sont plus tournées vers la rentabilité. On tombe alors sur un milieu assez hétérogène avec d'un côté les communautés avec les valeurs d'origines, et d'un autre une vision plus industrialisée. Il y a depuis 10 ans, un fonctionnement commercial de l'*open source*. Par contre, en Wallonie, on est plutôt en retard par rapport à l'*open source*. Pour exemple, les parts de marché de Microsoft sont plus élevées ici que dans d'autres pays.

PME v. GROSSES ENTREPRISES

Il n'y a pas vraiment de règles en fonction des tailles des entreprises. On retrouve de l'*open source* dans toutes sortes de sociétés différentes. Les montants investis sont différents par contre. En 2001, IBM met 1 milliard USD pour développer Linux. SUN a racheté MySQL pour 1 milliard USD quelques années après. Microsoft sponsorise la fondation Apache. Oracle a également toute une série de projets, notamment dans le monde Java. A côté de ça, il y a tout un tissu de PME qui se sont créés soit pour faire du service. Par exemple de l'installation dans les entreprises. Ou alors de l'édition *open source*. Odoo en est un bon exemple. L'avantage par rapport aux grosses entreprises est que les processus de développement sont très ouverts. Les relations de travail sont plus horizontales. Ce n'est pas toujours efficace dès lors d'utiliser de l'*open source* car il y a trop de pions entre les deux parties. Par exemple, une collaboration entre Nokia et Samsung sur de l'*open source* n'a pas fonctionné car il y n'y avait pas de contacts direct entre les développeurs mais beaucoup d'intermédiaires. Du coup c'est trop lent. Les grosses boites ne sont pas toujours assez réactives. L'évolution de Microsoft est intéressante : sur 15 ans on est passé de "pas question de toucher à de l'*open source*" à "on essaierait bien sur des petits projets" à "on développe de l'*open source*". On voit que ça a pris 15 ans pour cette structure. Lorsqu'on est une petite structure, ça va potentiellement beaucoup plus vite pour rentrer dans ces logiques de collaboration avec les communautés. Par contre, si vous avez une PME sans grosses compétences en informatique, ça ne va pas aller. Il y a une courbe d'apprentissage non négligeable car ce sont globalement des nouveaux outils et des nouvelles manières de travailler. Autre facteur, si vous travaillez dans les applications mobiles, vous pouvez utiliser de l'*open source* pour le développement. Par contre le modèle de valorisation est plutôt payant et on se retrouve dans un modèle sous vente de licences ou d'abonnements. Je ne pense pas qu'on puisse lier l'utilisation de l'*open source* avec la taille d'une entreprise. On se trouve bien souvent dans du cas par cas.

MARKETING

Il y a d'une part le côté bouche-à-oreille. On met le logiciel en libre téléchargement, les gens vont en parler autour d'eux. Dans beaucoup de cas, ils vont revenir vers vous car ils n'arriveront pas à l'installer et vous pourrez prester des services. Un deuxième aspect intéressant, le modèle des doubles licences. Cela permet de jouer sur la marque, qui devient alors une propriété. Si par exemple 1 milliard a été investi dans MySQL, c'est parce que 95% des développeurs connaissent MySQL. C'est peut-être l'aspect un peu plus traditionnel du marketing utilisé en *open source*.

Il y a également les cas où d'un côté on a une version communautaire, relativement autonome, et d'un autre une entreprise qui vend une version commerciale basée sur la version communautaire. Prenons l'exemple de Red Hat et CentOS. CentOs reprend ce que fait Red Hat en supprimant la marque et ne gardant que les codes qui sont libres. Il y a également la version communautaire de Red Hat qui est Fedora. Ils font ce que fait Red Hat mais sans garantie et sans service de mise à jour. D'un point de vue marketing, c'est assez intéressant à analyser ce qui se passe dans un cas comme celui-ci. Un autre cas intéressant est celui de Netscape et Mozilla.

Mozilla est plutôt tourné développeur tandis que Netscape est tourné utilisateurs finaux qui ne connaissent rien. Netscape a été dissout en 2003 et Mozilla a dû développer une vitrine pour séduire l'utilisateur final. Mozilla est d'ailleurs surement l'un des seules qui arrivent à toucher autant l'utilisateur final. La majorité des logiciels *open source* est utilisée par des intermédiaires. Ou des adopteurs précoces. Les autre seront intéressés par des produits finis auxquels il ne faut pas chipoter et directement prêt à l'usage. C'est très difficile de leur vendre du vrai *open source*. Comment les faire payer en plus? C'est un véritable problème et jusqu'à présent ça n'a jamais vraiment marché.

AVENIR

L'arrivée du cloud amène un gros changement pour l'*open source*. Il a cependant deux effets antagonistes. Le combat de base de l'*open source* qui est de libérer des postes de travail devient un combat d'arrière-garde car ce que vous aviez sur le poste de travail passe dans le navigateur et se retrouve hébergé. Si on regarde aujourd'hui, vous avez un Google Documents permettant de faire de la bureautique en ligne. Peut-être pas aussi performant qu'open office ou microsoft office mais il permet de faire les fonctions de base. Le combat pour le poste de travail libre est un combat perdu car le poste de travail, qui est microsoft actuellement, va se déplacer dans le navigateur progressivement. Ce ne sera ni Microsoft ou Linux mais probablement Android. Qui pour le coup, est plutôt fermé d'un point de vue applicatif. Deuxième chose, un cloud est principalement construit avec de l'*open source*. Vous en trouvez partout dans le cloud. Prenez Microsfot Azur, il y a une très grosse partie *open source*. Ensuite, il y a un modèle d'innovation et de développement qui se passe en *open source*. Prenez l'open data ou l'open hardware. On passe dans du matériel également, même si une bonne partie de l'innovation passe par du logiciel. Par exemple, dans les voitures. Une partie des technologies du domaine *open source* vont se retrouver dans des biens manufacturés. Si on prend Tesla, on télécharge une mise à jour sur leur site, on la met sur clé usb et dans la voiture. Celle-ci se met à jour. Ca change beaucoup la dynamique d'innovation.

Waroquiers, Philippe - Responsable du developpement des systemes de traitement de plan de vol et de gestion de flux, Eurocontrol

Pourquoi de l'*open source*

On a commencé avec de petites activités car on trouvait des oss de bonne qualité qui nous permettaient de réaliser nos besoins de façon efficace. On pouvait essayer ça relativement facilement comme c'était disponible librement. Du coup, pas besoin de faire d'appel d'offres ou de discuter de budgets. Comme c'était au début des logiciels, on utilisait uniquement dans le développement. Pas de contraintes niveau procédures et autorisations.
Cela ne partait pas d'une question derrière des convictions philosophiques, etc. Cela marchait bien, du coup on l'a utilisé.

On ne réfléchit pas au fait que ce soit *open source* ou non. Ici, l'oss répondait à nos besoins et était de qualité. C'est uniquement pour cela qu'on a opté pour.

On fait une étude de marché pour voir ce qu'il y a de disponible et on prend ce qui nous plait.
Par contre le fait que ce soit *open source* apporte un plus car si jamais on a une question sur le fonctionnement ou un problème technique, on peut investiguer plus facilement. C'est une qualité d'être *open source* mais on ne choisira pas un logiciel sur cet unique critère.

Coûts

Le coût n'était pas un critère. C'est un plus bien sûr, mais le choix de l'*open source* n'était pas basé là-dessus. Il y a bien sur des oss qu'on utilise gratuitement mais ceux qu'on utilise pour nos activités critiques, on paie pour bénéficier des services de support. Par exemple, nos serveurs centraux tournent sous RedHat Linux. On pourrait utiliser des serveurs complètement gratuitement, mais on préfère utiliser RedHat pour avoir certaines garanties niveau sécurité et présence et support en cas de problèmes. L'*open source* nous permet d'avoir accès au code, ce qui est très utile étant donné l'usage qu'on en fait. Il est toujours plus facile de comprendre un programme quand on peut avoir accès au code. Et en plus de ça, on paie pour avoir accès aux services.
Il n'y a pas vraiment de règle pour dire que de l'*open source* ou du closed source va coûter plus ou moins cher. Initialement, on ne pait rien en *open source* mais on paie des services ultérieurement. En closed source, on paie l'achat de licence, qui peut être plus ou moins important, et généralement on paie annuellement x % du prix initial pour la maintenance et le support. Du coup le modèle *open source* n'est pas fondamentalement différent au niveau des paiements. Par contre, je pense personnellement qu'on reçoit un service de meilleure qualité avec l'*open source* car si on n'est pas content, on peut facilement aller ailleurs. Par exemple si RedHat ne me donne pas satisfaction, je peux aller chez des concurrents ou tout simplement le faire tout seul. Du coup ça oblige à être efficace dans le service. Contrairement au closed source où il y a moins de concurrence. Par exemple si on n'est pas satisfait avec Oracle, il est plus difficile de passer vers une autre base de données.

Mais du coup, il est difficile de dire si l'*open source* nous revient moins cher ou non dans le long terme. Et honnêtement, ce n'est pas vraiment une question que nous nous posons. Nous sommes satisfaits des OSS que nous utilisons..

Cloud & closed source

Pour les activités critiques, on ne se pose pas la question. Actuellement, il n'y a qu'un seul fournisseur de services en traitement de vol et c'est nous même. Par contre, pour des activités mineures, on y réfléchit. Par exemple, on utilise un service cloud pour le suivit de certains incidents. On est passé d'un système interne à des SaaS. Donc le cloud et externalisation des activités via du service, bien sûr. Mais dans notre cas, je ne pense pas que cela arrivera pour notre activité première étant donné que c'est un domaine très spécifique et pointu. On utilise de l'*open source* sur lequel on écrit et on développe nos logiciels mais nous n'écrivons pas de l'*open source* pour d'autres personnes. Cela reste en interne et nos applications sont protégées. On peut dire que des lignes *open source* servent de fondations et de base à nos applications. On ne modifie pas ou extrêmement peu les OSS qu'on utilise. Après on développe nos produits nous mêmes. Si l'on modifie l'un ou l'autre logiciel, nous notifions bien entendu le fournisseur. Mais ce n'est pas notre boulot de modifier et corriger ces logiciels.
Vu notre secteur d'activité, nous ne pensons pas que quelqu'un va développer une application répondant à nos besoins. Après si quelqu'un se présente avec une solution closed source, nous ferons un appel d'offres et une étude pour opter pour la meilleure solution par rapport à nos besoins. C'est vraiment du cas par cas. Tout le reste étant égal par ailleurs, entre un oss et un logiciel closed source, nous prendrions très certainement l'oss car le fait d'être ouvert représente une qualité.

Avenir

On voit qu'il y a de plus en plus d'*open source*. Lorsqu'on compare à quelques années, il ne fait aucun doute qu'il est de plus en plus présent au sein des entreprises. Bien que nous nous dirigeons vraisemblablement vers le cloud. L'*open source* sera toujours présent pour bâtir ces solutions. Par contre, le cloud et les services à distance peuvent faire poser certaines questions : à qui appartient ces données ? Que se passe-t-il si le fournisseur de services arrête ses activités ? Ce n'est pas mon domaine mais il y a matière à se poser des questions. Par contre si c'est basé sur de l'*open source*, peut-être que cela peut permettre à quelqu'un de reprendre le service plus facilement. L'*open source* peut être une garantie à la continuité des services fournis dans le cloud.

Limites

Je ne vois pas vraiment où seraient les limites techniques de l'*open source*. Même d'un point de vue sécurité, *open source* ne veut pas dire accès à tout. J'ai par exemple assisté à une conférence où il était mentionné que l'*open source* était également utilisé dans le monde militaire. D'autant plus si l'utilisation reste en interne, même les licences ne sont pas un frein si les modifications apportées ne sont utilisées qu'en interne. Par exemple, nous décidons de ne pas publier les

modifications que nous apportons aux logiciels *open source* que nous utilisons car selon nous, notre fournisseur doit s'y retrouver aussi et on paie ses services en espérant qu'ils continuent à développer le logiciel et le maintenir. Si on se met à distribuer chaque nouvelle version, cela va à l'encontre de notre intérêt et du leur. Après, je pense que l'*open source* ne se prête pas forcément à tous les business modèles. Par exemple, l'univers des jeux vidéos. Si un jeu vidéo est développé via de l'*open source*, en fonction des licences, il ne serait peut-être difficile d'en tirer un revenu.