Forks impacts and motivations in free and open source projects

R. Viseur
Teaching Assistant, Department of Economics and Innovation Management,
Faculty of Engineering, University of Mons, 20, Place du Parc, 7000 Mons, Belgium.

Senior Research Engineer, Centre of Excellence in Information and Communication Technologies,
29/3, Rue des Frères Wright,
6041 Charleroi, Belgium.

***Abstract*— Forking is a mechanism of splitting in a community and is typically found in the free and open source software field. As a failure of cooperation in a context of open innovation, forking is a practical and informative subject of study. In-depth researches concerning the fork phenomenon are uncommon. We therefore conducted a detailed study of 26 forks from popular free and open source projects. We created fact sheets, highlighting the impact and motivations to fork. We particularly point to the fact that the desire for greater technical differentiation and problems of project governance are major sources of conflict.**

Keywords- open source; free software; community, co-creation, fork.

I. INTRODUCTION

Bar and Fogel define forks as situations occurring when developers "make a separate copy of the code and start distributing their own divergent version of the program" [2]. Free and open source software has four freedoms: the freedom to run, to study, to redistribute copies and modify the software (gnu.org). The free and open source software licenses guarantee the four freedoms, which involve the provision of source code [20]. Forks are usually observed in the field of free software. Forking is indeed a right that stems from the four freedoms associated with the software.

Mateos Garcias and Steinmueller distinguish mechanisms of forking and hijacking. The hijacking occurs when individuals "depose the project leader who has resisted the revision, leaving this original leader with no followers" [18]. In this paper, we will use "fork" for "forking" or "hijacking".

The title of Rick Moen's essay, "Fear of Forking", is characteristic of the fear of forks among entrepreneurs [19]. When he announced the LibreOffice fork (from OpenOffice.Org), Bruce Guptill, consultant for the analyst firm Saugatuck (www.saugatech.com), estimated for example that "the nature of open source leads to fragmentation, itself leads to uncertainty". As a failure of cooperation, forks are an interesting research topic.

The paper is organized as follows.

We will explore the concept of forks. We will then study a set of forks that occurred within popular free and open source software projects, and identify their motivations and impacts. Finally we will discuss the results, and propose ways to better prevent forks.

II. BACKGROUND

A. Perception of fork

If the fear of forks is visible with companies, Gosain also points to the sensitivity of the open source community beside the forks and the fragmentation of projects [10].

Bar and Fogel estimate that forks are often the result of a management mismatch [2]. They recommend forking only if necessary and if able to do better job. If the motivation for forking is the slowness of patches release, they recommend producing patches instead. Fogel notes, however, the scarcity of forks and a preference for trying to reach an agreement [8].

Eric Raymond estimates that forking "spawns competing projects that cannot later exchange code, splitting the potential developer community" [29]. He also distinguishes the case of "pseudo-forks", i.e. distinct projects that share a large common code base (this is for instance the case of GNU/Linux distributions). Weber considers that specialization may, in some cases, be managed through a system of patches, so as to avoid fragmentation of the project [39].

B. Forks and governance

For Hemetsberger and Reinhardt, management of online collaboration is less a question of coordinating tasks than overcoming conflicts arising from the contradictions between collective strategy and individual actions [13]. The voluntary nature of contributions often prevents the enforcement of duties or decisions (principle of consensus). Dahlander and Magnusson also consider that capture of network externalities requires specific skills (it has a cost) and that gains associated with the opening decrease when the number of players increases [5]. They highlight the difficulty in aligning business and community strategies. Bowles and Gintis distinguish the operating logic of a community, and the ones of companies and states [4]. The tensions that may result do not necessarily cause a fork. However the example of Netscape illustrates the difficulty of finding a tradeoff between a company and a community [36].

Implementation of common rules and effective governance structures should limit the tensions and especially their consequences. Eric Raymond distinguishes several structures

for the management of free and open source projects [28]. First, a single developer can work on the project and take all decisions alone. He is expected to pass the torch in case of failure to maintain the project.

Second, multiple developers can work under the direction of a "benevolent dictator". This structure is found in the Linux kernel (Linus Torvalds) or in Emacs (Richard Stallman). The potential for conflict is higher. Authority comes from responsibility and some developers become in practice responsible for one or more parts of the software. Another principle complements this rule: seniority prevails. The title of benevolent dictator may be passed on to another developer, as in the Perl project. Third, the decisions can be made by a panel of voters. This is for example the case for the Apache project.

The 2000s have seen the increasing involvement of businesses in the development of free and open source softwares, by initiating projects, freeing existing projects or collaborating with well-established communities [34, 38]. The increasing size of projects and cooperation between sometimes competing businesses (coopetition) also contributed to the creation of more complex and formal governance structures.

C. Forks and licenses

In practice, project license modulates the interest in whether to fork, even if no free and open source license cancels the risk [32]. Two major types of free licenses exist: permissive licenses (also named academic or unrestrictive licenses) and copyleft licenses (also names reciprocal or restrictive licenses) [1, 16, 20, 35]. A permissive license allows the user to apply a different license, possibly a proprietary license, to derivative works (thus also to forks).

A copyleft license "links the rights to the obligation to redistribute the software and its changes only under the same license as that by which the licensee has obtained those rights" [20]. In case of copyleft licensed software, exchanging source code is still possible between the original software and its forks. In case of a permissive free software license, the license can change and forbid the exchange of source code. In particular, the exchange will be impossible if the new software is published under a proprietary license, and one-way if it is published under a copyleft license (due to the fact that copyleft imposes conservation of the original license) [20]. St. Laurent considers other legal provisions limiting forkabily (or, if not, the consequences), such as brand protection in the Apache license [32]. Incompatibilities between licenses, sometimes due to apparently innocuous terms in legal texts, also reduce the opportunities for exchange and combination of source code [9, 32, 35]. Yamamoto, Matsushita, Kamiya and Inoue show, through a study of source code similarities applied to BSD (BSD-Lite, FreeBSD and NetBSD), a progressive divergence of the source code, despite the license compatibility and the similarity of features [41]. St. Laurent also considers this divergence as inevitable with time [32].

Finally, a copyleft license would also limit the financial incentives to fork as it is not possible to create a proprietary branch from the original development [40].

Elie considers unstable (and subject to a higher risk of fork) projects characterized by the coexistence of free release of the software and a second version published under a proprietary license (dual licensing, delayed publication,...) [7]. Elie names "hybrid model" this principle of "discrimination between users". Dahlander and Magnusson estimate on the contrary that the detention of copyright (and other controls) hampers forks initiatives (and allows the return to a proprietary development in case of insufficient network externalities) [5]. The technical complexity of the software would also reduce the risk of fork [33].

Note that the hybrid model suggested by Elie is distinct of the hybrid model described by Muselli [7, 22, 23, 35]. The later indicates a strategy of openness, promoting greater distribution while allowing to retain control over the project. This approach is supposed to facilitate the capture of value by the company and nullify the risk of fork. Muselli gives Sun Microsystems SCSL license as an example.

D. Forks impacts

Wheeler shades the presumed dangerousness of the fork and associates it with a system of healthy competition [40]. He compares it to the principle of a censure motion in parliament or to a strike. The fork would allow the developers community to attract the leaders' attention on the requests that are not taken into account. Some authors even see an "invisible hand" that guarantees the projects sustainability and continuity [26]. The ability to fork would also keep "the communities vibrant, and the companies honest" [21]. Elie sees the fork as "a fundamental right" but also insists on the risk of being cut from the wealth of the core [7]. He often sees in forks the consequence of "ill-defined control systems". Merit in free software communities would come from charisma and ability to live in the conflict rather than technical competences.

Wheeler recognizes that too many forks can cause a weakening of a projects family in the long term [40]. Spinellis and Szyperski see it as a waste of efforts and a source of confusion for the community [31]. Wheeler also distinguishes the forks as variants of software created with a goal of experimentation. A "winning mutation" can finally be accepted as constituting the best approach to a problem. Wheeler sees four possible outcomes to a fork:

- The fork does not convince and disappears.
- The original project and the fork evolve and gradually diverge.
- The original project and the fork merge after a period of cohabitation.
- The original project disappears.

III. Related works

Nyman and Mikkonen, in a study of 566 projects hosted on Sourceforge.net and presented by their maintainers as forks, identify motivations classifiable into four categories: technical motivations (adding features, specialization, porting, improving), license changes, local adaptations (language or regional differences) and revival of abandoned projects [25]. Open source company Smile also mentions disagreements about technology directions and licensing, but adds disagreement on trade policy as possible cause of fork [30].

Many forks benefit from more or less extensive studies (or are briefly discussed) in the literature. It includes the family of BSD operating systems [39, 41], KHTML [11], Roxen [5], GCC [8], CVS [2], NCSA HTTPd [34, 38] or SPIP [7]. These results will be used in this study.

IV. Methodology

We have studied 26 forks of popular free and open source projects. Popular projects have been found more likely to provide usable observations. We relied on existing documents: books, scientific articles, press releases, news on portals about open source and computer science, or projects pages. We have not considered forks leading to the creation of proprietary software, like Kerberos [32].

For each fork we gather relevant information in dedicated forms (fact sheets). They describe the chronology of each fork, its actors and their motivations. The results were summarized in a table, including the initial project name, fork name, fork motivation(s) and its impact on the original project. The impact was evaluated according to the possible outcomes identified by Wheeler [40].

The influence of the license type and the degree of openness of the project management structure were also observed. We assigned a score for openness on a scale from 1 to 4:

- the project is under a free and open source license but centrally managed,
- the project is managed by a team and the rules are informal,
- the decision-making procedures are planned, but favor core team,
- the procedures are documented, decisions and appointments are subject to the votes of active community members.

Note that the management structure may be difficult to precisely determine when the fork is old and/or a project has been completely abandoned.

V. Results

Six motivations to fork have been identified: death of the original project (19%), technical motivations –e.g. new specialization, divergent technical views, different technical objectives,...– (42%), license change (15%), conflict over brand ownership (12%), problems of project governance (38%), cultural differences (8%) and searches for new innovation directions (4%).

In practice, the case studies show that the successful forks (which are likely to be harmful to the original publisher, if there is one) usually start for an important reason.

Stopping the support of popular free and open source software often leads to a fork (see NCSA HTTPd, 386BSD, Red Hat Linux or Roxen). The open source fork succeeds but usually can coexist with a closed version of the product (see Red Hat Linux or Sourceforge).

A fork can occur after the emergence of technical differences. The BSD systems have thus often adopted different technical specializations such as portability or security [31]. This is the most common cause (42%).

Project governance is a source of conflicts for nearly half of the studied cases (38%). The problem is usually a lack of openness of development teams: slowness for taking external contributions into account (see OpenOffice.org), discussion of project objectives (see Sodipodi), maintainer's reluctance to switch to a community development process (see OpenOffice.org, Dokeos, PHP Nuke),... This is therefore the second most common cause of fork.

Brand ownership also appears as a source of conflict (see Claroline, Mambo and OpenOffice.org). It may be related to the issue of governance as the trademark allows the software editor to keep a check on the progress of the project. The brand then crystallizes the tensions between an editor and a community once their objectives diverge.

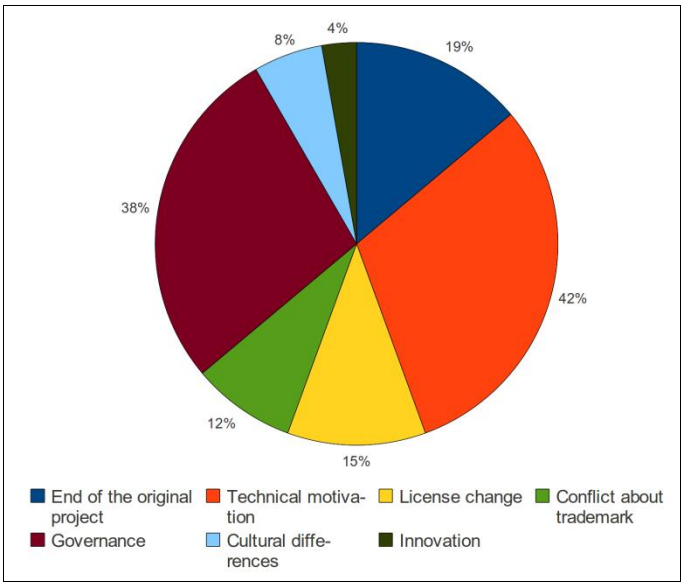

Figure 1. Motivations to fork.

Licensing problems sometimes cause a fork. It may not affect the type of license (see Xfree86) but rather increase (see Ext JS) or reduce (stop the free branch) the software freedom. Licensing software under the GPL or AGPL can facilitate exchanges between projects, since the original license can hardly be changed. The license change is not a dominant motivation to fork (15%).

Forks that have been raised by Theo de Raadt, leader of OpenBSD, can be justified, at least in part, by political or ideological positions. This configuration seems quite marginal in the free and open source landscape. Culture shocks between community and company (see KHTML) or community and administration (see Spip) appear as a possible cause (8%) and illustrate the difficulty in aligning business and community strategies.

The case studies show that the majority of forks do not cause the extinction of the original project (81%). Exception made of the Apache server, X.Org, Joomla or Inkscape,

cohabitation appears in more than half of the cases studied (54%). In some cases, the exchange of source codes exists (see FreeBSD, NetBSD, OpenBSD). Subsequent projects fusion (see GCC and EGCC) is possible. The progressive divergence may hamper the merger (see Webkit and KHTML). The complete failure of a fork occurs in less than one case out of five (19%).

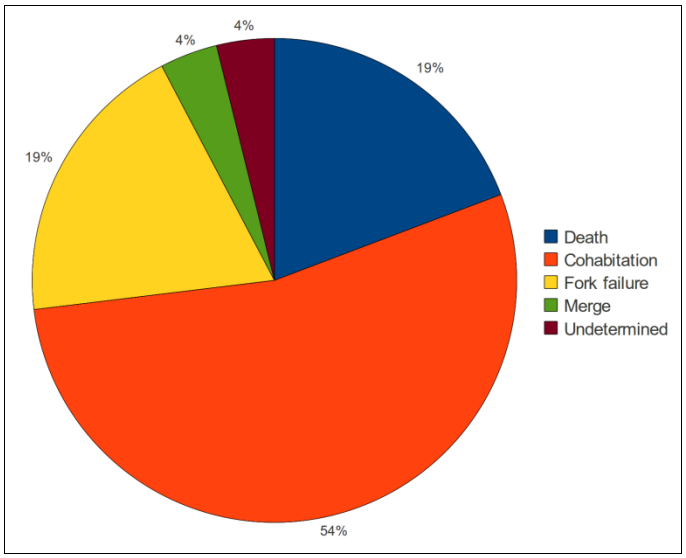

Figure 2. Forks impacts on original projects.

Finally, we find that nearly eight out of ten forks adopt a governance structure characterized by comparable or greater openness than in the original project. The formal rules of processes can give a biased impression of openness, that complaints made against the source code contribution mechanisms may moderate. The OpenOffice.org project (before entering the incubator of the Apache Foundation) is an example.

VI. Discussion

Compared to the study of Nyman and Mikkonen, our research groups several motivations under the label of "technical motivations" and highlights three additional causes: governance issues, difficulties associated to culture differences (already mentioned in state of the art) and conflicts over the ownership of a brand [25]. The changes in technical guidance also occupy a prominent place in our study, although proportionally less. The recovery of stopped projects is most frequent. These differences may be explained by the wider spectrum of motivations considered in our study but also by the different nature of considered projects. Nyman and Mikkonen are based on a set of projects taken on Sourceforge.net, which hosts many small projects, whereas our study was based on popular and mature projects. These have already an active community that plays a role in regulating and empowering the actors.

Many beliefs are refuted by our study. First, the use of copyleft licenses does not reduce the risk of forks. More than six out of ten studied forks were indeed published under a copyleft license (about 75% of free and open source projects are released under a copyleft license [16]). Second, hybrid models do not seem particularly subject to forks (except Chamilo). Third, the fear of a fork driven by competition (and perceived as an act of predation) seems exaggerated: only the case of OpenBravo could possibly be taken as such.

Privatization of popular free and open source software often results in a free software fork. However, the transition from a more permissive free license to a less permissive free license may also lead to a fork. The license change, regardless of its meaning, very often raised tensions in the community. The license choice must be well thought out from the beginning.

The risk of fork due to technical divergences is high. However, it may be limited by adopting a suitable architecture from the beginning. MacCormak, Rusnak and Baldwin recommend a modular architecture [17]. They point to the need for an "architecture for participation" to ease the comprehensibility of the code and the contribution. Mozilla project is an good example. The code left by Netscape was made more modular, and that contributed to attract patches from community [6].

The "kernel-extension model" is an example of modular architecture. It allows the improvement of the software without impacting its core. The editor then guarantees the performance of a core incorporating common features. Integrators and advanced users improve the functionality by developing extensions [3]. This approach can also reduce conflicts with the development team because the integrators need only understand the software interfaces for extensions development. Understanding the specifics of the kernel is not needed. Conflicts may occur on the other hand between community extensions and proprietary extensions sold by the editor.

Promotion of such an architecture underpins the creation of application programming interfaces (APIs), and reminds of the "user toolkits for innovation" described by Von Hippel [37]. These toolkits permit a form of outsourcing to users for innovation tasks requiring deep understanding of customers' needs. The expected benefit is a better satisfaction of customers and, in a free software project, a lower risk of tensions around the project orientations.

Samba illustrates the "killer of innovation" side due to the quality requirement when a large user base exists. This example highlights the value of incubators, such as Apache incubator, allowing experimentation next to the main project. In a way Samba TNG plays an incubator role. A similar effect can be achieved by creating experimental branches in the repository (see Linux).

VII. Conclusion

The goal of this proposal is to shed some light on the motivations and impact of the fork mechanism in free and open source software projects. This paper identified the main motivations to fork, that are technical divergences and governance mismatches. Other causes were highlighted: end of the original project, license change, conflict about trademark and strong cultural differences.

We discussed some ways to manage tensions and prevent project splitting, for example by improving software modularity.

VIII. FUTURE WORKS

The governance issues generally relate to a lack of communication with the community.

However, it seems difficult to conclude definitely on the choice of a specific governance model. Indeed, some projects governance structures appear to be open (cf. FreeBSD, KHTML, OpenOffice.org,...) but are also subject to forks. Moreover some successful projects are build on main developers' strong authority. Thus Mozilla community enforces code ownership (e.g.: module owner) despite the risk of disputes in the community [15, 24].

A more detailed study of these structures, and in particular their interactions with developers, should therefore be considered. The analyze of messages exchanged between developers before, during and after forks would maybe allow to identify specific reasons for the schisms. Data could be extracted (for qualitative or quantitative researches) from public collaborative tools such as mailing lists and bugtrackers (e.g.: [6, 14, 15, 27]).

REFERENCES

[1] T.A. Alspaugh, H.U. Asuncion, and W. Scacchi, "Intellectual Property Rights Requirements for Heterogeneously-Licensed Systems", 17th IEEE International Requirements Engineering Conference (RE '09), September 2009.

[2] M. Bar and K. Fogel, Open Source Development with CVS, Paraglyph Press, 2003.

[3] P. Bertrand, "Les extensions, arme fatale des solutions Open Source", Journal du Net, 28 juin 2010.

[4] S. Bowles and H. Gintis, "Social Capital and Community Governance", Economic Journal, Royal Economic Society, vol. 112 (483), November 2002, pp. 419-436.

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

[6] J.-M. Dalle and M.L. den Besten, "Voting for Bugs in Firefox", FLOSS Workshop 2010, July 1, 2010.

[7] F. Elie, Économie du logiciel libre, Eyrolles, 2006.

[8] K. Fogel, How To Run A Successful Free Software Project - Producing Open Source Software, CreateSpace, 2004.

[9] D.M. German and A.E. Hassan, "License integration patterns: Addressing license mismatches in component-based development", IEEE 31st International Conference on Software Engineering, ICSE 2009, May 2009, p. 188-198.

[10] S. Gosain, "Looking through a window on Open Source culture : lessons for community infrastructure design", Systèmes d'Information et Management, 2003, 8:1.

[11] A. Grosskurth and M.W. Godfrey, "Architecture and evolution of the modern Web browser", Preprint submitted to Elsevier Science, June 20, 2006.

[12] B. Guptill, "OpenOffice Changes Highlight Fragmentary Nature and Future of Open Source", Saugatuck Technology, 2010.

[13] A. Hemetsberger and C. Reinhardt, "Collective Development in Open-Source Communities: An Activity Theoretical Perspective on Successful Online Collaboration", Organization studies, vol. 30 n°9, September 2009, pp. 987-1008.

[14] J. Howison, K. Inoue, and K. Crowston, "Social dynamics of free and open source team communications", IFIP International Federation for Information Processing, vol. 203, 2006, pp. 319-330.

[15] S. Krishnamurthy, "About Closed-door Free/Libre/Open Source (FLOSS) Projects: Lessons from the Mozilla Firefox Developer Recruitment Approach", Libre software as a field of study, Upgrade, vol. VI, issue n°3, June 2005.

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

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

[18] J. Mateos Garcia and W.E. Steinmueller, "Applying the Open Source Development Model to Knowledge Work", INK Open Source reasearch working paper n°2, January 2003.

[19] R. Moen, "Fear of Forking - How the GPL Keeps Linux Unified and Strong", Linuxcare, November 17, 1999.

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

[21] G. Moody, "Who Owns Commercial Open Source – and Can Forks Work?", Linux Journal, April 2, 2009.

[22] L. Muselli, "Les licences informatiques : un outil de modulation du régime d'appropriabilité dans les stratégies d'ouverture. Une interprétation de la licence SCSL de Sun Microsystems"., 12ème Conférence de l'Association Information et Management, Lausanne, juin 2007.

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

[24] M. Nurolahzade, S.M. Nasehi, S.H. Khandkar, and S. Rawal, "The role of patch review in software evolution: an analysis of the Mozilla Firefox", Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, 2009.

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

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

[27] L. Prechelt and C. Oezbek, "The search for a research method for studying OSS process innovation", Empirical Software Engineering, vol. 16 (4), 2011, pp. 514-537.

[28] E.S. Raymond, "Homesteading the Noosphere", First Monday 3 (10), October 1998, pp. 90-91.

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

[30] Smile, Comprendre l'open source et les logiciels libres, Livre blanc, www.smile.fr.

[31] D. Spinellis and C. Szyperski, "How is Open Source affecting software development?", IEEE Software, January-February 2004, pp. 28-33.

[32] A.M. St.Laurent, Understanding Open Source and Free Software Licensing, O'Reilly Media, 2004.

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

[34] R. Viseur, "Gestion de communautés Open Source", 12ème Conférence de l'Association Information et Management, Lausanne, juin 2007.

[35] R. Viseur, "La valorisation des logiciels libres en entreprise", Jeudis du Libre, Université de Mons, 15 septembre 2011.

[36] R. Viseur, "Associer commerce et logiciel libre : étude du couple Netscape / Mozilla", 16ème Conférence de l'Association Information et Management, Saint-Denis (France), 25-27 mai 2011.

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

[38] E. von Hippel and G. von Krogh, "Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science", Organization Science, vol. 14 no. 2, March / April 2003, pp. 209-223.

[39] S. Weber, The success of open source, Harvard University Press, April 30, 2004.

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

[41] T. Yamamoto, M. Matsushita, T. Kamiya, and K. Inoue, "Measuring similarity of large software systems based on source code correspondence", Product Focused Software Process Improvement, vol. 3547, Springer Berlin / Heidelberg, 2005, pp. 530–544.

AUTHORS PROFILE

Robert Viseur was born in Mons, Belgium, in 1977. He graduated from the Faculty of Engineering of Mons. He earned a Ph.D. in Applied Sciences in 2011.

He is Teaching Assistant at the Department of Economics and Innovation Management in the Faculty of Engineering (University of Mons, Belgium) and Senior Research Engineer at the Centre of Excellence in Information and Communication Technologies (Charleroi, Belgium).