# Challenges in Using Open Source Software in Product Development: A Review of the Literature

Klaas-Jan Stol
Lero, the Irish Software Engineering Research Centre
University of Limerick
Limerick, Ireland
+353 61 23 3737
klaas-jan.stol@lero.ie

Muhammad Ali Babar
IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S, Denmark
+ 45 7218 5107
malibaba@itu.dk

## ABSTRACT
Component-Based Software Development has become a popular approach to building software intensive systems. Besides using Commercial Off-The-Shelf components, an organization may choose to use Open Source Software components. Using OSS has been reported to have many benefits, but there are also challenges involved. Understanding the potential challenges of using OSS in developing products is important for practitioners, so they become aware of them and can anticipate them and take appropriate measures to address these challenges. We have performed a thorough review of the literature to identify challenges that may arise, as reported in the literature. This paper presents and discusses these findings. Researchers can discuss potential causes and solutions of our synthesized findings as well as benefit from provided references to literature on OSS challenges as input for future research.

## Categories and Subject Descriptors
D.2.13 [**Software Engineering**]: Reusable Software – *Reuse model*

**General Terms:** Management, Theory

**Keywords:** Open Source Software, Challenges, Component-based Development, Literature Review

## 1. INTRODUCTION
In the last decade or so, it has become quite common to build software intensive systems following a Component-Based Software Development (CBSD) approach using Off-The-Shelf (OTS) components [38, 44]. Components may be built in-house, for instance as part of product line development or through an "inner source" approach [45], or acquired from third parties. When acquiring third-party components, an organization may purchase Commercial OTS (COTS) components, or decide to use Open Source Software (OSS) products. OSS products are becoming more commercially viable [16]. An increasing number of available OSS products make the use of OSS an attractive alternative to COTS components. Using OSS is often reported to have many benefits, such as significantly lower (purchasing) costs, availability of high quality products, adherence to open standards and no vendor dependency. Software development industry has taken note of these benefits, and has been increasingly using OSS components (OSCs) in combination with, or as an alternative to COTS components [27]. For small independent software vendors (ISVs) that operate in a niche market, using OSCs instead of proprietary (i.e. not free of charge) software allows them to use advanced technologies without the need to purchase expensive licenses that may minimize any margins on the final product [36, 42].

However, besides these benefits, several studies have also reported different challenges involved in using OSCs in software development. However, the studies reporting such challenges tend to focus on specific aspects. Merilinna and Matinlassi reviewed the literature on OSC integration and compare this with real-world practices [31]. Ven and Verelst studied challenges and strategies for ISVs when dealing with modifications and contributions to OSS [42]. Morgan and Finnegan [32] provide an overview of benefits and drawbacks of adopting OSS. However, their review is mostly based on online articles, such as TechSoup (www.techsoup.org), reports by commercial research institutions (e.g. Forrester) and other types of reports, rather than findings from scientific research literature. Goode reported a literature review and industrial survey on management barriers to OSS adoption [21]. Our study focuses on challenges involved in using OSS for product development (hereafter referred to as OSS challenges).

There has been no systematic synthesis of the OSS challenges reported in the literature. We assert that a synthesis of the reported OSS challenges can help practitioners to fully understand the potential OSS challenges and enable them to take appropriate measures to deal with them. Researchers can use the findings for deliberating and debating the possible causes and appropriate strategies for the identified challenges. This assertion motivated us to undertake a thorough review of the literature to systematically identify and synthesize the reported OSS challenges. This paper reports the research method used and findings from our study. The remainder of this paper proceeds as follows. Section 2 outlines our research method. Section 3 presents and discusses the results of our review. Section 4 concludes and presents an outlook to future work.

## 2. RESEARCH METHOD
In order to identify all the relevant research papers, we conducted a thorough search of the literature based on the guidelines for conducting a Systematic Literature Review

(SLR), as presented by Kitchenham in [25]. We do not claim that our study is an SLR. Our intention is to identify and synthesize all challenges of using OSS in product development, rather than providing a comparative overview or synthesis of any empirical evidence for these challenges.

To identify all relevant papers, we relied on two different sources. Firstly, we selected a number of papers from our search results from our ongoing extension of the SLR reported in [37]. For this extension, we searched a number of digital libraries for all studies that were related to open source software research, disregarding the type of the papers. The resulting repository contains approximately 550 papers. Secondly, we manually selected papers reported in the proceedings of the five editions of the International Conference on Open Source Systems (2005 to 2009).

We included papers that report on product development with OSS, including CBSD with OSS. Papers that report on adoption of, or migration to OSS were excluded, since they do not address issues in product development but rather focus on usage (by end-users) of the software. We acknowledge that the difference between product development and adoption can be a very thin line; we required that a paper had to report on some sort of development activity, rather than just replacing an existing proprietary or commercially purchased system with an OSS alternative.

Based on these criteria, we identified 44 studies. We inspected each of these papers to identify issues, concerns and challenges related to product development with OSS. We did not require that challenges in the studies were empirically grounded. In other words, a paper could be an experience report, workshop report, or otherwise. A number of papers that we initially selected did not mention any challenges. Others discussed the use of OTS in general, not differentiating between OSC and COTS components. After our selection procedure, we were left with 17 papers.

We thoroughly read all identified papers to identify any reported challenges related to using OSS in product development. Some papers explicitly listed challenges, whereas others implicitly reported the issues. We extracted both challenges that were reported as authors' experiences or as cited from other literature. While reading the papers, we recorded all challenges in a spreadsheet. After extracting the challenges, we annotated each one with one or more keywords. In particular, we found a number of challenges that could be classified as 'community', 'support' and 'maintenance', since maintenance (e.g. bug fixes) is a type of support, which is provided by the community. Based on these keywords, a number of categories emerged, which we used to cluster related challenges. During analysis of the challenges, we merged similar challenges that essentially stated the same issue.

# 3. RESULTS AND DISCUSSION

The results of our review are presented in Table 1. We identified 21 challenges that have been reported in literature. The table indicates the number of times each challenge has been reported (column "#") as well as references to the reporting studies. The bracketed number after the category names indicates the total number of reports of challenges in that category.

## 3.1 Product selection

### *3.1.1 Too much choice*

Various studies reported that identifying quality products among the many available OSS products is difficult due to an uncertainty about the quality (C1). The quality is typically referred to in terms of quality attributes such as usability, reliability and performance. Sourceforge, the largest repository for OSS projects, alone hosts more than 230,000 projects. This challenge has long been recognized, and has resulted in a variety of OSS evaluation methods and frameworks, such as Capgemini's Open Source Maturity Model (OSMM) [14], Navica's OSMM [20], OpenBQR [39] and QSOS [4]. However, despite these efforts, research has shown that practitioners typically do not use these evaluation methods and frameworks [28, 31]. Rather, they use ad-hoc approaches and information sources to select components, such as experiences of colleagues. Hauge et al. found that a "first fit" rather than a "best fit" principle is applied [22].

### *3.1.2 Lack of time to evaluate*

A related challenge to C1 is a lack of time to evaluate components (C2). Though this is of course a direct consequence of having a large number of OSS products to evaluate, we decided it should be listed separately, since it was separately identified by a study ([5]) that also reported challenge C1. Furthermore, possible measures that practitioners could take to address C2 are different from measures for C1 and distinguish C1 more clearly from C2. Such measures include: 1) limiting the number of components to evaluate to a small number, and 2) a decision on management level to allocate more resources (time, manpower) for evaluating OSCs.

### *3.1.3 Choosing a fork*

Another related challenge to C1 is deciding what "fork" of the OSS project should be used (C3). If a fork occurs, a new project is spun off from the original project, and can occur if a project's core developers have fundamental disagreements about the future of the project [36]. This challenge is different from C1, since C1 refers to making the decision on what product to select, whereas C3 refers to what fork of that product should be selected. Forking projects rarely happens, and there is a strong social pressure against forking [34]. Nevertheless, if it happens, developers need to decide which fork of the project to select. One experience paper reported that this decision caused a temporary delay in development [7].

## 3.2 Documentation

A lack of good quality documentation remains to be a challenge that is difficult to overcome (C4). Well-documented software is easier to understand by others, which makes it easier to modify the software. However, OSS contributors are typically more interested in coding, and some consider adding comments in the source code is sufficient [17]. Researchers have proposed various architecture recovery methods to overcome the lack of design and architecture documentation [13].

The availability of different descriptions of the same components is problematic as well (C5). OSS products may have documentation, but due to the active evolution of many OSS products, this documentation may quickly go out of date.

## 3.3 Community, support and maintenance

A wide variety of challenges have been reported with respect to the interaction with the community. This interaction can be related to (future) support for the product as well as contributions to the project. Maintenance can be done by both the OSS product's community as well as through contributions from the product's users. In fact, the boundary between "community members" and "users" may not be that clear according to the onion model [12], which states that the social structure of an OSS community is layered, and users are just another layer. As Gacek and Arief state: 'all OSS developers are users, but not all users are developers' [17].

**Table 1. Challenges in integrating OSS in product development**

| Category | ID | Challenge | # | Reported in |
|---|---|---|---|---|
| Product Selection (9) | C1 | Identifying quality products among the large supply is difficult due to uncertainty about quality (e.g. usability, stability, reliability) | 7 | [5, 10, 11, 23, 26, 31, 41] |
| | C2 | Lack of time to evaluate components | 1 | [5] |
| | C3 | Decide what "fork" of the project should be chosen | 1 | [7] |
| Documentation (5) | C4 | Lack of, or low quality documentation | 4 | [2, 5, 29, 31] |
| | C5 | Several descriptions of the same component | 1 | [6] |
| Community, support and maintenance (19) | C6 | Dependency on the community for further support and upgrades; possible need to hire additional talent for maintenance; difficult to control the quality of the support; lack of helpdesk and technical support. | 5 | [10, 11, 26, 31, 41] |
| | C7 | Custom changes need to be maintained, which is time-consuming and may cause problems with future versions/community may take a different, incompatible approach. | 6 | [7, 23, 24, 30, 42, 43] |
| | C8 | Convincing OSS community to accept changes (modifications may be too specific); contributions can be difficult or costly. Difficult to control the architecture if not a core member. | 5 | [7, 23, 30, 31, 42] |
| | C9 | Uncertainty about product future and consequences for company product | 1 | [7] |
| | C10 | Community members would like to have a bigger say in features and integrating final product with company | 1 | [24] |
| | C11 | Contributing and investing in OSS project costs resources | 1 | [24] |
| Integration and Architecture (8) | C12 | Backward compatibility concerns | 2 | [24, 41] |
| | C13 | Modifications needed to implement missing functionality or fit into architecture | 2 | [41, 42] |
| | C14 | Incompatibility between components or existing systems | 2 | [11, 41] |
| | C15 | Horizontal integration | 1 | [31] |
| | C16 | Vertical integration / Mismatch of platform/programming language | 1 | [31] |
| Migration and usage (3) | C17 | Complexity of configuration | 1 | [41] |
| | C18 | User training/learning costs | 2 | [11, 41] |
| Legal and Business (5) | C19 | Complex licensing situation | 5 | [1, 24, 29, 35, 41] |
| | C20 | Concerns about, or no clear strategy on Intellectual Property and Rights issues | 3 | [1, 35, 41] |
| | C21 | Lack of clear business models that are appealing to industry | 2 | [1, 11] |

### *3.3.1 Dependency for future support*

If an organization decides to use an OSS product, it is dependent on the community for future support and upgrades. A challenge is to acquire support for the OSS product that is of sufficient quality (C6). Support for an OSS product is provided on a voluntary basis by the community, which makes it difficult to control the level of quality of support that is needed. However, for some products, there is an option to acquire support and training from companies such as Red Hat and IBM [15]. Various studies report the support to be a challenge (see Table 1). Ven and Verelst reported a study that investigated the reliance of organizations on commercial support [43]. They found that the absence of available commercial support is not an insurmountable obstacle for adopting OSS. However, they also found that the OSS community is primarily used by organizations with a strong technical background.

### *3.3.2 Need to maintain custom changes*

If changes are made to an OSS product, and these modifications are not given back to the product's community, then the software developers that made such changes need to maintain these custom changes themselves (C7). This means that additional resources must be allocated for the maintenance

efforts. Furthermore, if an organization does not give back the modifications, it effectively 'forks' the project as the customized modifications define a new version of that product. This may have serious consequences for future compatibility. Modifications may have to be re-applied whenever new versions of the OSS are used. As an OSS product evolves, patches that implement modifications may no longer be easily applied. Alternatively, an OSS product's community may tackle a certain feature or issue themselves by taking an approach that may be incompatible with the customized modifications [42].

### *3.3.3 Difficult to get changes accepted*

An organization may decide to contribute the changes made back to an OSS component (that is integrated into a product). However, sending patches to an OSS's community does not automatically imply that these patches get accepted (C8). A key characteristic of an OSS development process is that contributions are thoroughly scrutinized by community members that have commit access [15]. In general, any changes or proposals for change will be subject to a review process [34]. Furthermore, an OSS project may have specific practices that developers will have to get familiar with [7]. Specific extensions may be rejected to prevent that too many new features (code bloat) are introduced by one-time contributors [42].

### *3.3.4 Uncertainty about product future*

A realistic concern that organizations may have is the future, or longevity of the OSS product (C9). Obviously, if a certain OSS product is adopted, an adopter does not like to be in a situation where the community supporting that product disappears. If that happens, it means no support or updates for that particular product. In such a case, an organization may choose to take over the maintenance of the project. However, this would result in additional maintenance efforts, and may distract the organization from its core business.

### *3.3.5 Community wants more influence*

One study reporting experiences of product development with OSS at Nokia reported a challenge similar to C8, but in the opposite direction. OSS developers expressed their wish to be more closely involved in features of the final product. The closed way of integrating these OSS products can cause frustration among OSS developers [24] (C10). To partially address this issue, Nokia started a special distribution, to allow anybody participate more closely in the development. However, as stated in [24], product companies must have the final control over their products.

### *3.3.6 Contributing costs resources*

An organization may choose to use OSS as-is without further development. Alternatively, using OSS becomes more effective if the organization actively participates in a community's development process. This, however, requires additional resources (C11). The amount of resources required depends on the level of involvement. Bonaccorsi et al. [9] list three kinds of involvement: 1) project coordination; 2) code development collaboration, and 3) provision of code.

## 3.4 Integration and architecture

### *3.4.1 Backward compatibility issues*

An OSS product is continuously evolving, depending on the liveliness of a community. Changes to products include new features, bug fixes and architectural changes. After an organization starts using an actively evolving product, new versions are released. As a product's development continues, at some point newer versions are no longer backward compatible, which can become a problem if the product in which the OSS is integrated depends on certain features or APIs (C12).

An organization will have to adopt a strategy for updating any used OSCs. On the one hand an organization may choose to stay close to the latest version of the OSC. However, this has consequences for backward compatibility, as features may be deprecated and architectural changes may occur [24]. Ven and Mannaert describe four possible strategies for contributing to OSS projects [42]: 1) contributing any modifications, 2) taking regular snapshots, 3) forking and 4) initiating an OSS project as a set of patches to an existing OSS project. One solution to this problem is to use only those distributions that are provided by packaging companies [35].

### *3.4.2 Need for modifications*

A consequence of a CBSD approach is that components must be fitted into a system. That means the components may have to be modified. Furthermore, OSCs may have to be modified if they do not have all required functionality. Such modifications require additional resources (C13). Obviously, OSCs are more flexible than COTS because the source code of OSCs is available for modifications. However, many organizations do not usually make any changes to the source code before using OSCs [28].

### *3.4.3 Component and architecture incompatibilities*

OSCs may not be compatible with each other, or with existing architectures (C14). This phenomenon is called architectural mismatch, which may have serious consequences for the development schedule and costs [18]. Another compatibility issue that may arise is that components may have dependencies on conflicting libraries [42].

### *3.4.4 Horizontal integration issues*

Merilinna and Matinlassi [31] distinguish horizontal integration issues at the architectural level and the component level (C15). No specific architectural level issues have been identified. For the component level, four design level contracts have been discussed in [8]. These levels are: 1) syntactic interface, 2) (pre- and post-conditional) constraints, 3) synchronization and timing and 4) quality-of-service. Each of these levels can have associated challenges.

### *3.4.5 Vertical integration issues*

A mismatch of platform and programming language is an example of vertical integration issues (C16). Platforms may be hardware (i.e., processor types) or software (operating systems and virtual machines (VM) such as the Java VM, .NET and Parrot VM). Some of the techniques to overcome the vertical integration problems are use of middleware (e.g. CORBA), virtual machines (e.g. JVM, a hardware platform-independent virtual machine) and Model-Driven Architecture (MDA) [31].

## 3.5 Migration and usage

Complexity of configuring or setting up a user-environment can be an issue (C17). One study reported that significant effort was required to set up an installation (approx. three weeks) [41].

Two studies reported additional cost involved in migration to an OSS alternative and staff training to be a challenge (C18). In our study, we focus on product development with OSS rather than adopting OSS in favor of proprietary solutions. Migration cost seems to imply the migration from a proprietary solution to an OSS solution, such as the migration from Microsoft Office to OpenOffice.org. However, such end-user applications may be integrated as a sub-system of a larger solution.

## 3.6 Legal and business

### *3.6.1 Complex licensing situation*

Not surprisingly, several studies reported the complex OSS licensing situation to be an issue (C19). One study reports a lack of consistency between licensing agreements and little guidance on interpreting the open source licenses [41]. At the time of writing, the Open Source Initiative lists 65 licenses that comply with the "Open Source Definition" [33]. It is therefore not surprising that OSS licensing is perceived to be a complex issue. Some research efforts have been made to address this issue. Alspaugh et al. [3] present a license analysis scheme, and an approach to automatically analyze license interactions. German and González-Barahona have documented a number of strategies that developers have used to legally circumvent some restrictions of the GPL [19].

### *3.6.2 Concerns and issues regarding IPR*

Organizations that use OSS products may have concerns about intellectual property and rights (IPR) (C20). Code may have been illegally used and propagated. For instance, in recent years there have been some claims from Microsoft saying that Linux uses their intellectual property [40]. In that particular case, a deal was made with Novell, so that customers of Novell's SUSE Linux distribution are protected from any claims.

### *3.6.3 Lack of clear business models*

Two studies reported a lack of clear business models for using OSS (C21). We note that both studies were published in 2005; however, this issue has not been mentioned in more recent literature. In [17], three business models have been identified that motivate organizations to get involved in OSS: 1) software for own use, 2) packaging and selling of the software and 3) a platform for commercial or research software development.

## 3.7 Limitations of this study

Our study has some limitations, which we discuss here. Though we performed a rigorous literature search, we may have unintentionally excluded studies due to the subjectivity of our inclusion and exclusion criteria. Our classification of challenges is necessarily subjective. However, it is not our intention to present a definitive classification; rather, we intend to present our findings in a structured way that can help practitioners to inform them of challenges in using OSS in product development that have been reported so far, and may therefore arise in their situation.

# 4. CONCLUSION AND FUTURE WORK

Open Source Software products are being increasingly used as an alternative to Commercial Off-The-Shelf components. Using OSS has been reported to have many benefits, but also has various challenges. Therefore, practitioners may be reluctant to integrate OSS products as part of their final product. We assert that a good understanding of OSS challenges can help practitioners to be well prepared for such challenges so that they can take appropriate measures to address those challenges. This paper presents the results from our study aimed at systematically identifying and synthesizing the challenges reported to be involved in using OSS in product development. We believe these findings can be equally useful for practitioners and researchers. Practitioners will become more aware of the potential challenges of using OSS in product development. The research community can deliberate and discuss the potential causes and solutions of the synthesized challenges. Furthermore, the references to the literature on OSS challenges can benefit researchers interested in doing future research in this area. We intend to continue this research, which will focus on the challenges regarding the software architecture and the effects on a system's quality as a result of integrating OSS components.

# 5. ACKNOWLEDGMENTS

This work is partially funded by IRCSET under grant no. RS/2008/134 and by Science Foundation Ireland grant 03/CE2/I303_1 to Lero—The Irish Software Engineering Research Centre (www.lero.ie).

# 6. REFERENCES

[1] Ågerfalk, P.J., Deverell, A., Fitzgerald, B., and Morgan, L.: Assessing the Role of Open Source Software in the European Secondary Software Sector: A Voice from Industry, Proceedings of the First International Conference on Open Source Systems, 2006.

[2] Akkanen, J., Demeter, H., Eppel, T., Iványfi, Z., Nurminen, J., and Stenman, P.: Reusing an open source application — practical experiences with a mobile CRM pilot: Open Source Development, Adoption and Innovation (2007).

[3] Alspaugh, T., Asuncion, H., and Scacchi, W.: Analyzing software licenses in open architecture software systems, Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS '09. ICSE Workshop on, 2009.

[4] Atos Origin: Method for Qualification and Selection of Open Source software (QSOS) version 1.6, 2006.

[5] Ayala, C., Hauge, O., Conradi, R., Franch, X., Li, J., and Velle, K.S.: Challenges of the Open Source Component Marketplace in the Industry. Proc. Fifth IFIP WG 2.13 International Conference on Open Source Systems, 2009.

[6] Ayala, C., Sorensen, C., Conradi, R., Franch, X., and Li, J.: Open Source Collaboration for Fostering Off-The-Shelf Components Selection. Proc. Third IFIP WG 2.13 International Conference on Open Source Systems, 2007.

[7] Bac, C., Berger, O., Deborde, V., and Hamet, B.: Why and how to contribute to libre software when you integrate them into an in-house application?, Proceedings of the First International Conference on Open Source Systems, 2005.

[8] Beugnard, A., Jezequel, J., Plouzeau, N., and Watkins, D.: Making components contract aware, Computer, 1999, 32(7).

[9] Bonaccorsi, A., Lorenzi, D., Merito, M., and Rossi, C.: Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research. Proc. First International Workshop on Emerging Trends in FLOSS Research and Development, 2007.

[10] Chen, W., Li, J., Ma, J., Conradi, R., Ji, J., and Liu, C.: An empirical study on software development with open source components in the chinese software industry, Software Process: Improvement and Practice, 2008, 13(1).

[11] Conlon, P., and Carew, P.: A Risk Driven Framework for Open Source Information Systems Development, First International Conference on Open Source Systems, 2005.

[12] Crowston, K., and Howison, J.: The social structure of free and open source software development, First Monday, 2005, 10(2).

[13] Ducasse, S., and Pollet, D.: Software Architecture Reconstruction: A Process-Oriented Taxonomy, IEEE Transactions on Software Engineering, 2009, 35(4).

[14] Duijnhouwer, F., and Widdows, C.: Open Source Maturity Model, Capgemini Expert Letter, 2003.

[15] Feller, J., and Fitzgerald, B.: Understanding Open Source Software Development (Pearson Education Ltd., 2002).

[16] Fitzgerald, B.: The transformation of open source software, Management Information Systems Quarterly, 2006, 30(3).

[17] Gacek, C., and Arief, B.: The many meanings of open source, IEEE Software, 2004, 21(1).

[18] Garlan, D., Allen, R., and Ockerbloom, J.: Architectural mismatch: why reuse is so hard, Software, IEEE, 1995, 12(6).

[19] German, D.M., and Gonzalez-Barahona, J.M.: An Empirical Study of the Reuse of Software Licensed under the GNU General Public License. Proc. Fifth IFIP WG 2.13 International Conference on Open Source Systems, Skövde, Sweden, 2009.

[20] Golden, B.: Succeeding with Open Source (Addison-Wesley, 2004).

[21] Goode, S.: Something for nothing: management rejection of open source software in Australia's top firms, Information & Management, 2005, 42(5).

[22] Hauge, Ø., Osterlie, T., Sorensen, C.-F., and Gerea, M.: An Empirical Study on Selection of Open Source Software - Preliminary Results. Proc. ICSE Workshop on Emerging Trends in FLOSS Research (FLOSS '09), Vancouver, Canada, 2009.

[23] Hauge, Ø., Sørensen, C.-F., and Røsdal, A.: Surveying Industrial Roles in Open Source Software Development: Open Source Development, Adoption and Innovation (2007).

[24] Jaaksi, A.: Experiences on Product Development with Open Source Software. Proc. Third IFIP WG 2.13 International Conference on Open Source Systems, 2007.

[25] Kitchenham, B.: Guidelines for performing Systematic Literature Reviews in Software Engineering, 2007, Technical Report no. EBSE-2007-1.

[26] Krivoruchko, J.: The Use of Open Source Software in Enterprise Distributed Computing Environments, Open Source Development, Adoption and Innovation, 2007.

[27] Li, J., Conradi, R., Slyngstad, O.P.N., Bunse, C., Khan, U., Torchiano, M., and Morisio, M.: An Empirical Study on Off-the-Shelf Component Usage in Industrial Projects: Product Focused Software Process Improvement (2005).

[28] Li, J., Conradi, R., Slyngstad, O.P.N., Bunse, C., Torchiano, M., and Morisio, M.: Development with Off-the-Shelf Components: 10 Facts, IEEE Software, 2009, 26(2).

[29] Madanmohan, T.R., and De', R.: Open source reuse in commercial firms, Software, IEEE, 2004, 21(6).

[30] Mannaert, H., and Ven, K.: The use of open source software platforms by Independent Software Vendors: issues and opportunities. Proc. Fifth Workshop on Open Source Software Engineering, 2005.

[31] Merilinna, J., and Matinlassi, M.: State of the Art and Practice of Open Source Component Integration. Proc. 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, Cavtat, Dubrovnik, Croatia, August 29-September 1, 2006.

[32] Morgan, L., and Finnegan, P.: Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms: Open Source Development, Adoption and Innovation (2007).

[33] Open Source Initiative, http://www.opensource.org/licenses, accessed January 21, 2010.

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

[35] Ruffin, C., and Ebert, C.: Using open source software in product development: A primer, IEEE software, 2004, 21(1).

[36] Spinellis, D., and Szyperski, C.: How is open source affecting software development?, IEEE Software, 2004, 21(1).

[37] Stol, K., and Ali Babar, M.: Reporting Empirical Research in Open Source Software: The State of Practice. Proc. 5th IFIP WG 2.13 International Conference on Open Source Systems, Skövde, Sweden, June 3-6, 2009.

[38] Szyperski, C.: Component software: beyond object-oriented programming (Addison-Wesley, 1998).

[39] Taibi, D., Lavazza, L., and Morasca, S.: OpenBQR: a framework for the assessment of OSS. Proc. Third IFIP WG 2.13 International Conference on Open Source Systems (OSS 2007), Limerick, Ireland, 2007.

[40] TheRegister, http://www.theregister.co.uk/2006/11/20/microsoft_claims_linux_code/, accessed January 21, 2010.

[41] Tiangco, F., Stockwell, A., Sapsford, J., Rainer, A., and Swanton, E.: Open-source software in an occupational health application: the case of Heales Medical Ltd. Proc. The First International Conference on Open Source Systems, 2005.

[42] Ven, K., and Mannaert, H.: Challenges and strategies in the use of Open Source Software by Independent Software Vendors, Information and Software Technology, 2008, 50(9-10).

[43] Ven, K., and Verelst, J.: The Importance of External Support in the Adoption of Open Source Server Software, Open Source Ecosystems: Diverse Communities Interacting, 2009.

[44] Wallnau, K.C., Hissam, S.A., and Seacord, R.C.: Building Systems from Commercial Components (Addison-Wesley, 2002).

[45] Wesselius, J.: The Bazaar inside the Cathedral: Business Models for Internal Markets, IEEE Software, 2008, 25(3).