Open Source Software Usage Implications in the Context of Software Development

Gregor Polančič, Marjan Heričko and Romana Vajde Horvat
Institute of Informatics,
University of Maribor,
FERI, Smetanova 17, SI-2000 Maribor,
Slovenia
E-mail: gregor.polancic@uni-mb.si, marjan.hericko@uni-mb.si, romana.vajde@uni-mb.si

Keywords: open source software, open source projects, business process diagrams, risk benefit analysis.

Received: May 6, 2005

Open source software (OSS) is becoming increasingly popular in several aspects of software engineering activities, ranging from using OSS for development or execution environments to incorporating OSS directly into developed products. OSS and its development projects differ from proprietary software and closed source projects in several aspects. Therefore, these aspects should be known and analyzed, before making a decision for using OSS in a software development project. This paper analyses various OSS usage strategies in the context of software development projects. Dependent on cases of usage, different open source project collaboration models, based on business process models, are analyzed from several relevant aspects.

Povzetek: Na osnovi procesov sodelovanja in definiranih atributov so analizirane prednosti in tveganja različnih modelov uporabe odprtega programja v kontekstu projektov razvoja programske opreme.

1 Introduction

Software development projects are often timely and financial ineffective, while on the other hand producing low qualitative and vulnerably artefacts (software). Lack of quality and productivity in software development projects has raised several strategies capable of confronting with this problem.

According to Boehm (Boehm 1999), there are three major strategies for improving software development productivity and software quality:

- working faster (usually with better tools),
- working smarter (usually with more optimized processes) and
- work avoidance (usually with software reuse).

Two strategies presented above (working faster and work avoidance) are realized with software. Such software can be developed "in-house", obtained from another company (for free or purchased) or open source based.

In this article we analyse implications of incorporating open source software into software development strategy. Open source software (OSS), which is becoming increasingly popular and important (Brown & Booch 2002; Ruffin & Ebert 2004), is computer software that has its source code made available under an open source definition (OSD) based license (Open Source Initiative 2005). OSD based license implicates that the source code of software is released with binary, allowing users and developers to use and to modify the software and to distribute any improvements they make. Consequently, most of OSS is being developed in public accessible projects where everyone capable of contributing knowledge, ideas or code is welcome to join in. Such projects are called open source projects – OSP (see also Figure 1).

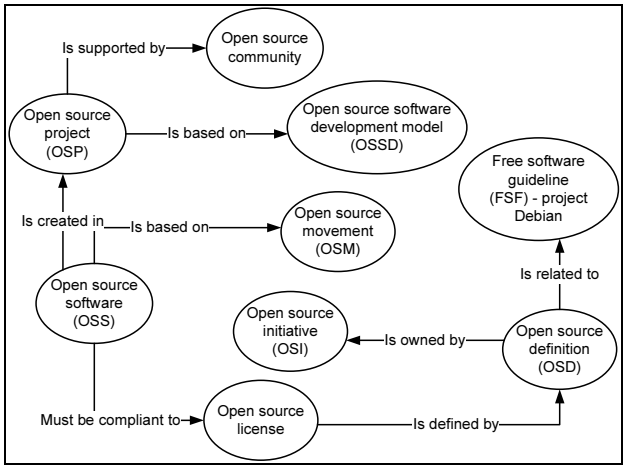

Figure 1: Relations between common open source terms

According to open source advocates, such development model leads directly to more robust software and more diverse business models (Wu & Lin 2001).

Software development companies are looking toward OSS as a way to provide greater flexibility on their development practises, jump-start their development efforts by reusing existing code and provide access to a much broader market of users (Brown & Booch 2002; Kasper Edwards 2004).

On the other hand, there are several risks and limitations concerned with using open source software, which should be properly addressed. Low code quality, non-

existing project plan and non-deterministic stability of the project are some of them (Fitzgerald 2004).
Related to open source software (potential) benefits and risks, which were mentioned above, the research question can be stated as "What are the implications of a specific open source software usage strategy in a software development project?"
Based on the research question, we identify and analyse different open source software usage strategies, for the purpose of determine benefits and risks of each strategy, with respect to software license, development processes and software, from the point of view of closed source software developer and in the context of business process models.

1.1 Scope of the Paper

Section two of the paper connects this research to the existing body of knowledge. In the section three, open source projects, their development model, its common design and characteristics are introduced. Additionally, a comparative study is performed, comparing open source and closed source (proprietary) projects.
Based on open source development model, its unique characteristics and related work (concerned with open source software usage in commercial environment), different usage strategies are presented and evaluated accordingly to predefined attributes.
The research has the following limitations. Closed source projects are defined as projects which are based on a well established development model. In the context of software collaboration processes between open source and closed source projects, only technical activities are analysed. Additionally, because of parsimony, the open source and closed source software development processes are presented on a high level view.

2. Related Work

Several descriptive studies exist in the field of using open source software (OSS) in commercial context.
In the article "Using open source software in product development: A primer", Ruffin and Ebert (Ruffin & Ebert 2004) state, that the use of OSS in industrial products is growing. They discuss major legal aspects and risks in using OSS and how to mitigate them in product development. Additionally, OSS must meet several criteria, required to reduce risks of technical and legal exposure during deployment.
Madanmohan and De in the article, titled "Open source reuse in commercial firms" (Madanmoban & De 2004) state, that using OSS components raises many issues, from requirements negotiation to product selection and integration. They define a model of the stages involved in locating and using an OSS component. Five critical issues for reusable OSS components are identified: cost, customization requirements, component characteristics, licensing, maintenance and support. They state that if the OSS component offers the best solution and reliability for the price, then it is the most appropriate.
In the article titled "Reusing open-source software and practices: The impact of open-source on commercial vendors", authors Brown and Booch (Brown & Booch 2002) find out that as a result of the open-source movement there is a great deal of reusable software available in the public domain, which can be used in commercial projects. Open source movement is described as a diverse collection of ideas, knowledge, techniques and solutions. Additionally, the authors state, that there are several questions concerned with applying OSS ideas into commercial environment.
The paper, titled: "Towards a Product Model of Open Source Software in a Commercial Environment", from Deng, Seifert and Vogel (Jianjun Deng, Tilman Seifert, & Sascha Vogel 2003) state that there are many reasons for commercial organisations to be interested in using OSS. Aspects of OSS development for commercial use are analysed in the paper. Second, different categories of OSP are identified together with typical requirements, which have to be realized by instances of OSS. Third, an open source process model, based on the concept of work products and product networks is defined.
Another type of research has published Edwards in the article titled "An economic perspective on software licenses—open source, maintainers and user-developers" (Kasper Edwards 2004). Based on economic theory, he defined several models, which illustrate the possible choices available to users and developers once a program has been distributed under a specific type of software license. The basics premise of the research is that users are prepared to contribute to projects if there is a net benefit. Based on two different open source (GPL and BSD) and a proprietary (Microsoft EULA) software license, three different models are developed by deducting the behaviour (activities) possible for software developers and users. Based on developed models, the incentives for developers and users together with their relationships are analysed. Individuals and organisations related to open source software are treated differently, because of different incentives for contributing to open source projects.

3. Open Source Projects

Open source projects (OSP) are software projects, which are based on open source software development model (OSSD), a recent phenomenon, which became available with the existence of the global communication infrastructure – internet. Because of open source license, OSP have different project structure, compared to "traditional" software projects.

3.1 Open Source Software Development Model

Most of commercial or proprietary software projects are based on closed source software development model (CSSD) (Vidyasagar Potdar & Elizabeth Chang 2004). Such development model follows strictly defined activities and their relationships. Several CSSD models exist, for example: cascade, spiral, iterative-incremental

(Figure 2), V-model and RUP (Rational Unified Process).

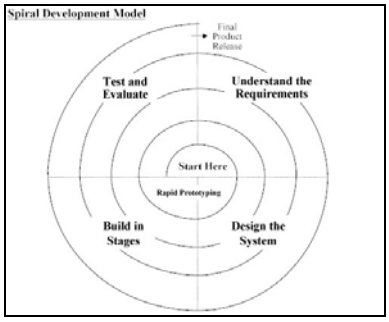

Figure 2: Spiral software development model

On the other hand, open source project are based on open source software development model (OSSD) (Vidyasagar Potdar & Elizabeth Chang 2004). OSSD is an evolutionary development model (Figure 3), where software is permanently evolving according to user needs (Vidyasagar Potdar & Elizabeth Chang 2004). OSS never reaches its final state, because it keeps evolving as long as there is an active user community available. Consequently, such development model emphasizes frequent minor point releases and as much feedback on these releases as possible.

Because no strict sequence of phases is defined in OSSD (Figure 3), OSP cannot be tracked according to phases. Instead, the progress is usually tracked with file versioning system, for example CVS (Concurrent Versioning System).

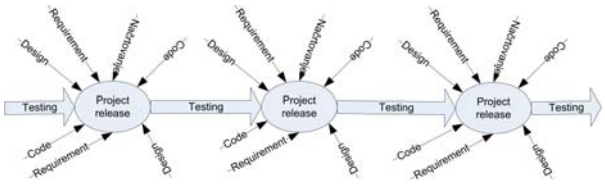

Figure 3: Evolutionary software development model (Michael Nash 2003)

3.2 Open Source Community Structure

Open source projects (OSP) are based on virtual community concepts. Because the available project resources are proportional to the user community size, they support open standards and standard development and collaboration tools. Because OSP usually lack of finances, they are trying to minimise project costs with using public available information infrastructure (for example "Sourceforge.net" repository).

OSS software communities are virtual work groups consisting of members with skills in software development. They work in temporary, cultural diverse, geographically dispersed, electronically communicating work groups (Wolfgang Maass 2004). Based on user roles, open source communities, are generally organized as presented below (Jen-Fang Lee & Tzu-Ying Chan 2004; Richard P.Gabriel & Ron Goldman 2002).

In the centre of the community is a small group of core developers (see also Figure 4). Core developers have most rights and also responsibilities in OSP. They have write access to source code's baseline. They make decisions concerned with code merging, quality assurance and releases.

Beside code developers, there is usually a larger group of code developers, which are developing new functions and performing other, less responsible tasks, for example: improving user interface, fixing bugs and writing documentation.

The largest group is represented by active and passive users. Active users participate in OSP in form of identifying bugs, proposing new features, creating documentation and offering user support. Passive users only use OSS and other project artefacts.

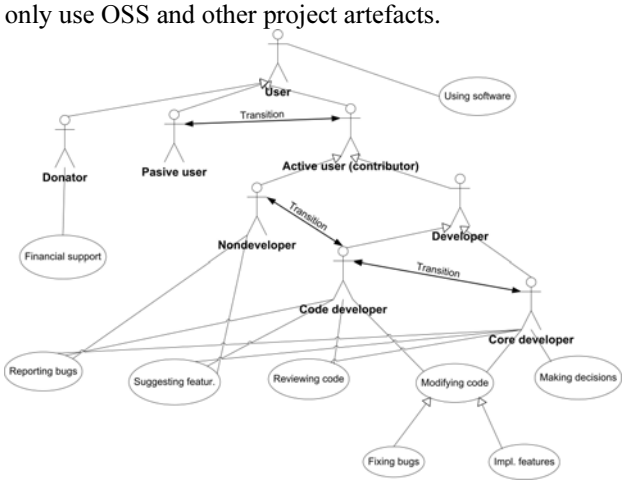

Figure 4: High level use case diagram of open source community

3.3 Open Source Project Characteristics

Open source projects have in common following characteristics (Gacek & Arief 2004):

a. Adherence to OSD (Open Source Definition), which acts as an open source accordance guideline.
b. Open source software developers represent a subset of open source user community (see also Figure 4). Consequently OSS developers are also OSS users.

Despite of commonalities presented above, OSP differ in several aspects (Gacek & Arief 2004):

a. Project starting point. OSP can start from scratch or from existing proprietary or research (closed source) project.
b. Motivation. A lot of open source research is related to motivational aspect of willing to freely participate in OSP (Andrea Bonaccorsi & Cristina Rossi 2005; Wolfgang Maass 2004). Individuals usually participate from personal believes or because they require functions which might be provided by OSS. Corporations usually get involved to gain market share, to lower their software infrastructure costs or to be less dependent from commercial software vendors.
c. Community. Two basic types of open source communities exist: centralized and decentralized. Central organized communities have a strict hierarchy of active users, which allows a more centralized power structure. Their opposites are

decentralized communities, which have looser organisational structures with most of developers on the same level. One-level organisational structure requires more sophisticated decision making processes.
The basic idea, underlying open source projects, is that knowledge, shown through contributions, increases the contributor's perceived merit, which in turn leads to power (this is called meritocratic culture).

d. Software development support. OSP differ in their modularity (high modularity is prerequisite for effective remote collaboration), visibility of software architecture (system architecture might be available or not), documentation, testing, submission acceptance (involves choosing the work area, decision making and disseminating the submission information), tools and collaboration support.
e. Licensing. Several types of licenses conform to OSD. From the user point of view the most important license characteristics are its impact on derived works and possibility to "close" the licensed software (Table 1).

Table 1: Implications of main OSD licenses (Gacek & Arief 2004)

OSD based license	Impact on derived works?	Can be closed?
GPL (GNU General Public License)	Yes	No
LGPL (GNU Lesser GPL)	No	No
BSD (Berkley Software Distribution)	No	Yes
IBM Public License	No	Yes
MPL (Mozilla Public License)	No	Yes

3.4 Open Source Project Compared to Closed Source Projects

Beside different development models, open source projects differ from closed source (proprietary) projects in several other aspects. Some of them are briefly presented below (Vidyasagar Potdar & Elizabeth Chang 2004):

a. Documentation. Within CSP, the process of writing documentation is defined in project plan or requirements. On the other side, OSP participants usually prefer writing code. Consequently, there is usually lack of qualitative and updated documentation.
b. Testing. In OSP software users act as software testers. This is called "many eyeballs" principle (Eric S.Raymond 2000). They either try to solve problems or to notice the community. CSP are tested by specified number of software testers.
c. Security. In CSP the security of software is achieved through obscurity, while in the OSP the security is achieved through openness of the code. Both strategies have their strengths and risks. However in highly secure systems, openness is preferred.
d. Release and delivery. In CSP, software might be released because of market pressures or defined project milestones. OSS is released when it meets release criteria. OSP releases are usually frequent but not scheduled.
e. Development environment. CSP are usually centralized on a single physical location. OSP development occurs in virtual communities which offer decentralized and distributed development.

4. Modelling Open Source Software Usage Strategies

Despite of differences between open source and closed source projects a lot of different collaboration opportunities exist between them (Brown & Booch 2002; Kasper Edwards 2004). Such OSS usage models depend on several factors, for example: business strategy, software license and software type.

4.1 Identification of Usage Strategies

Several OSS usage classifications exist. According to Gacek and Arief (Gacek & Arief 2004) following OSS business models are viable:

- using OSS for personal use,
- packaging and selling OSS,
- using OSS as a platform or foundation for commercial or research software development.

On the other hand, Edwards classifies software use, according to software licenses (Kasper Edwards 2004) into:

- commercial or proprietary license,
- BSD based open source license and
- GPL based open source license.

Ruffin and Ebert (Ruffin & Ebert 2004) classify OSS usage, dependent on the licensee role, into:

- end user OSS and
- OSS that is embedded into in a product that is further distributed. This is called software reuse.

Based on classifications presented above, their differences and commonalities, a use case model of common open source software usage strategies in closed source projects can be defined (Figure 5):

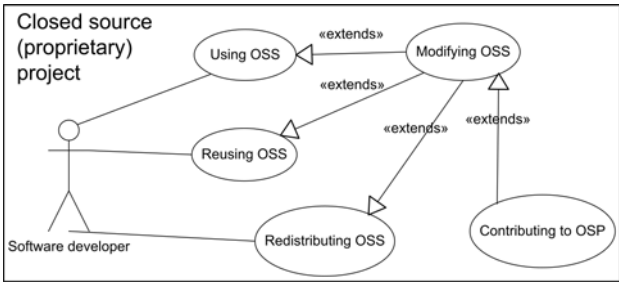

Figure 5: Use case model of common open source software usage strategies in proprietary projects

The identified strategies of using OSS in proprietary software projects are following (Figure 5):

a. Using OSS. OSS is used for project or product infrastructure, which includes: development tools, collaboration tools, software testing environment and software execution environment.
b. Reusing OSS. Reused OSS (for example: software snippet, software component or software framework) is embedded into developed product.
c. Redistributing OSS. Added value, based on additional artefacts (commercial software, documentation, plug-ins, etc.) and services is included and distributed with OSS. Distribution can be proprietary or open source based.

Each of the main strategies presented above, can be additionally divided accordingly to (Figure 5):
- using OSS as-is or suiting it to specific needs;
- treating modifications as intellectual property or committing them to the open source community.

Based on use case model presented on Figure 5, twelve (3x2x2) different OSS usage scenarios might occur, each with its strengths and risks.

4.2 Notation Used for Modelling Usage Strategies

Models of OSS usage strategies, resulting from use case model presented in section 4.1 (Figure 5), are based on business process modelling notation – BPMN (BPMI 2004). BPMN is developed by business process management initiative (BPMI). The current specification of BPMN, which is 1.0, was released to the public in May, 2004. BPMN defines a business process diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations. A business process model is a network of graphical objects, which are activities and the flow controls that define their order of performance. Four basic categories of elements in BPMN are (Stephen A.White 2004):
- flow objects (events, activities, gateways),
- connecting objects (sequence flows, message flows, associations),
- swimlanes (pools and lanes) and
- artefacts (data objects, groups and annotations).

We decided to use BPMN because it is easily understandable, supported by OMG (Object Management Group) and highly expressive.
We used Microsoft Visio as a software modelling tool. Additional, an open source based BPMN stencil was used. The stencil is available on Sourceforge.net repository (https://sourceforge.net/projects/bpmnpop).

4.3 Analysis of Usage Strategies

Based on resulting business process models, we performed two types of analyses.
First, we performed a high level risk-benefit analysis for each resulting model. Risk is the potential harm that may arise from some present process or from some future event. Risk-benefit analysis is the comparison of the risk of a situation to its related benefits. Risk-benefit analysis was performed on activities and relevant events that occur in resulting business process models.
Second, we performed a comparative study of all three usage strategies. Several attributes were defined for comparative study, ranging from user types, major benefits and desirable OSS characteristics. These attributes are presented in section 5.5.

5. Resulting Models

Based on OSS usage strategies, defined in section 4.1 we modelled and descriptively presented one generic and three special business models. They are presented and analysed in following subsections.

5.1 Generic Model

All special OSS usage models are derived from the top level usage model which is presented on Figure 6. Therefore the special models include same BPMN constructs (pools, events, messages, processes) as presented on generic model.
The generic model consists of two pools (rectangles), representing independent processes of OSP (Open Source Project) and CSP (Close Source Project), which differ in the underlying software development model. CSP development and OSP development are modelled with repeatable sub-processes (rounded rectangles with curved arrow and "+" sign).
The collaboration between projects is modelled with bi-directional data exchange using BPMN messages mechanism (dotted arrows) exchanging data objects (documents).

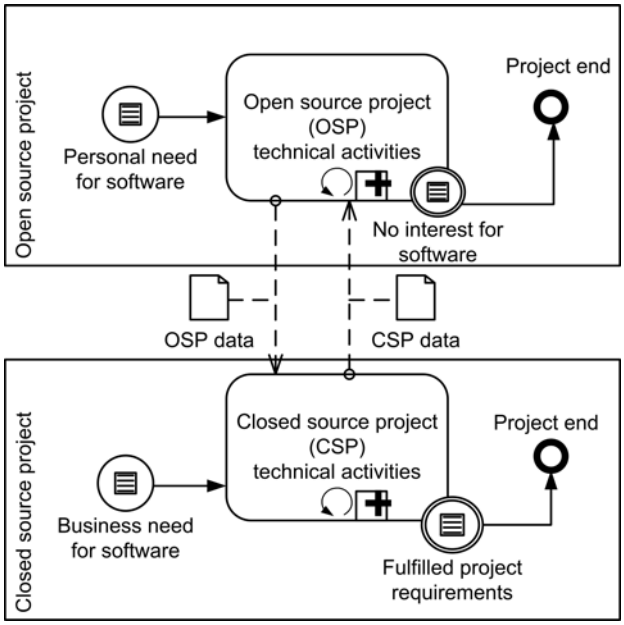

Figure 6: Generic OSS usage model

Additionally, different events (presented as rules in circles) initiate, direct flow and finish OSP and CSP.
There is usually a business need for starting a CSP, requiring sufficient human and financial resources. On

the other hand, OSP start, because there is a personal need for some functionality (software).
CSP usually have a predictive end, consisting of documented list of functional and non-functional requirements, which have to be fulfilled. OSP usually do not have a predictive end. Non predictive end might present a risk to CSP. Because OSP are "organic projects" they are finished if there is no interest for software being developed.

5.2 Using of OSS

Based on the business model on Figure 7, using OSS is comparable to using proprietary software. Because OSS is (in most cases) used as provided by OSP, modification activities are not modelled. However, OSS can be suited to specific needs, if necessary. As a new version of OSS is released, software developer (if necessary) installs new release and uses it as infrastructure software (development, maintenance, execution or collaboration software). When OSS is used, feedback information can be sent to OSP, for example: modification proposals, new feature requests and identified bugs. Using OSS might end with fulfilled CSP project requirements.

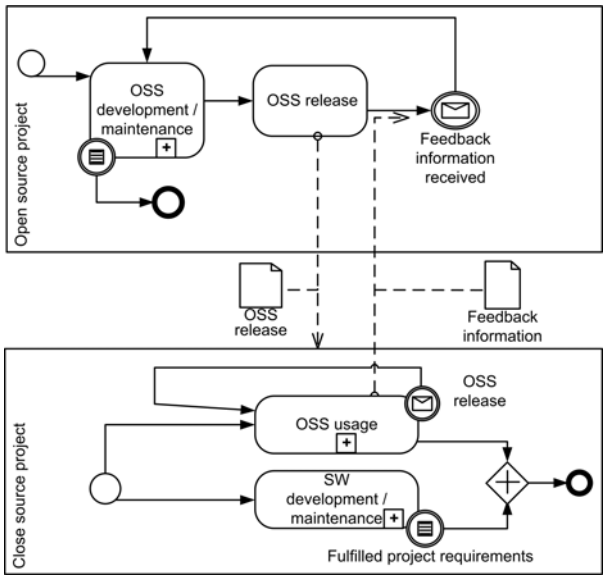

Figure 7: Model of using OSS

a. Benefits. The main objective of using OSS in CSP is to lower the cost of project infrastructure or decrease dependency from specific commercial software vendors. Additional, a benefit of using OSS can be free support and add-ons which are available from open source community. Beside, OSS can be influenced with sending feedback to CSP. In such way, OSS can be better suited to CSP needs.
b. Risks. There are several risks concerned with using OSS. First, releases are usually not determined. Therefore, planning the OSP on some future OSS releases is risky. Second, there are no legal guaranties for using OSS. For example, if there is a bug in OSS or a defined release date was postponed, nobody is responsible for potential damage. Third, there is no guarantee that feedback information will be considered by OSP. Feedback is usually considered if there is a community size interest for them.

5.3 Redistributing OSS

Commercial vendor can decide to redistribute OSS. Based on the model on Figure 8, an OSS redistribution project is restarted each time new stable version of OSS is released. Additional, CSP can make some modifications or additions to OSS, which can be sent back to community (for example: identified bugs or functions which can be further developed by user community) or (if the OSS license allows), treated as intellectual property of commercial vendor. Finally commercial vendor releases software (SW) package. Final users might send feedback information to CSP, which can further be mediated to OSP.

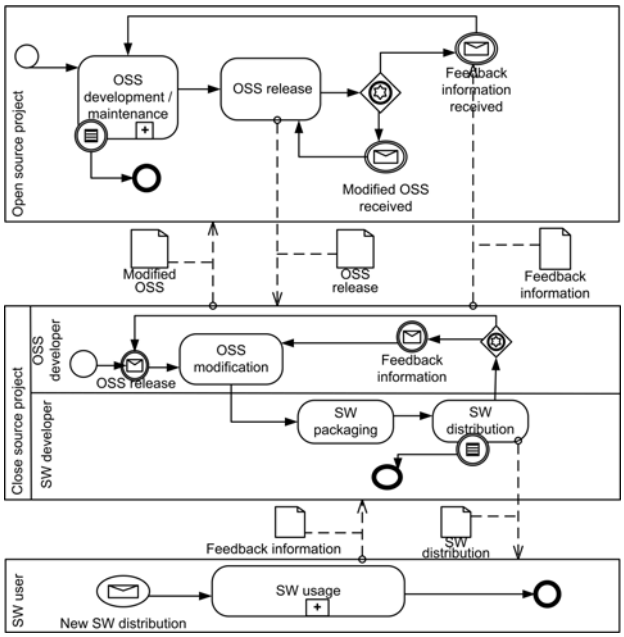

Figure 8: Model of redistributing OSS

a. Benefits. The main objective of redistributing OSS is to gain market share or to make profit from selling software, supporting services or distributions. Second OSP can be directly influenced by CSP with sending modified OSS code back to the open source community. In such way open source community can further develop or maintain code, which was primary developed by CSP. Consequently CSP costs are lowered.
b. Risks. Most of the risks, concerned with redistributing OSS, are related to non determined OSS releases and potentially unstable open source community. Therefore, planning release dated might be risky. Second there might be legal problems concerned with viral OSS licenses which prohibit that OSS changes licensing model. For example we cannot make binary distributions of GPL based software. Third, future directions of OSS might change unpredictably. For example, if CSP is distributing OSS with a proprietary plug-in,

problems could be caused with changed plug-in interface.

5.4 Reusing OSS

When reusing OSS in CSP, following activities occur (Figure 9). First, if a specific OSS component is suitable for software development, it can be adapted (if necessary) and afterwards included into developing software. Modified OSS can be sent back to OSP or it can be treated as intellectual property of CSP. Finally software is released together with reused OSS. End users use released software (SW) and if necessary, send feedback information to CSP. CSP can react to feedbacks with direct software changes or mediate feedbacks to open source community. OSS modification and integration activities are usually performed, when there is a new version of OSS available.

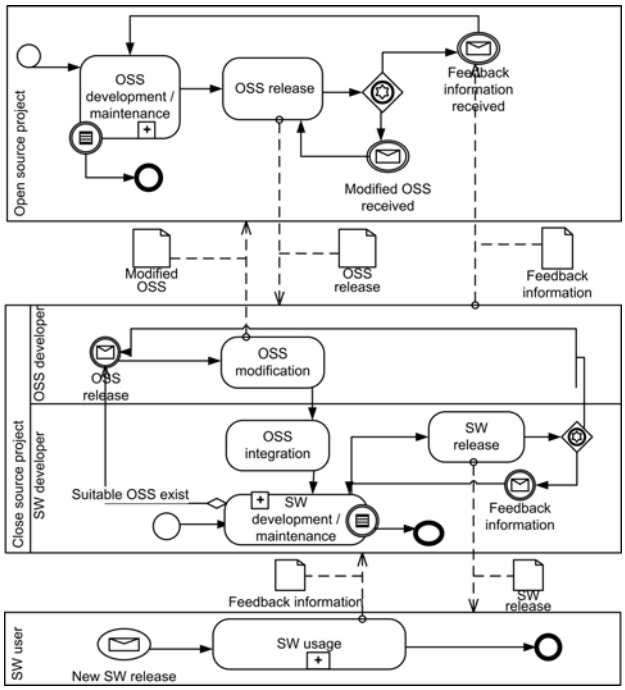

Figure 9: Model of reusing OSS

a. Benefits. The main objectives of reusing OSS in proprietary projects are simultaneously increasing productivity and software quality through OSP developed and maintained reusable software artefacts. Productivity is increased, because parts of software (reused OSS) are developed and maintained by OSP. Second software quality is increased because reused OSS is tested and improved by open source community.
b. Risks. Several risks are present in such reuse strategy. First, rarely or delayed OSS releases might influence (expand) CSP project plan. Second if, there are to frequent releases and unstable OSS architecture, a lot of effort is spent for OSS integration. Third, OSS license might prohibit reusing OSS in proprietary software (for example GPL or LGPL license).

5.5 Comparing Three Usage Models

Models, defined in previous section differ in complexity, benefits and risks. Additional, there are several other factors that should be considered before making a decision for a specific usage strategy. Following factors and sub-factors were considered in the comparative study:

a. Open source software (suitable software licenses according to Table 1, desirable software characteristics and most suitable software types).
b. Open source software user (OSS user roles, closed source developer activities when using OSS, most frequent collaboration artefacts between OSP and CSP).
c. Open source project major desirable characteristics.
d. Closed source project (major benefits, major investments and major risks).

Results of the comparative study are summarized below in Table 2.

Table 2: Results of comparing different OSS usage models

OSS usage strategy	Using OSS	Redistribute OSS	Reusing OSS
Suitable software license	All	Non viral licenses	Non viral licenses (BSD, IBM, MPL)
Desirable OSS characteristics	Quality in-use	Quality in-use, software quality, process quality	Software quality, process quality, reusability
Suitable OSS types	Infrastructure software	Infrastructure software and office tools	Reusable components and frameworks
OSS user roles in CSP	Active user	Developer	
Closed source developer activities related to OSS	Usage	Usage, modifications, packaging	Reuse, modifications, integration
Collaboration artefacts between CSP and OSP	Identified bugs, feature requests	Identified bugs, feature requests, code	
OSP major desirable characteristics	Good support	Stable releases	Stable architecture
Major benefits	Lower direct and indirect cost	Commercial distributions, market penetration	Increased productivity and software quality
Major OSS cost factors	Learning OSS	Learning OSS modifying OSS, collaborating with OSP	Learning OSS modifying OSS, integrating OSS, collaborating with OSP
Major risk	Low OSS quality, lack of support	Unsuitable OSS license, undetermined OSP stability	Unsuitable OSS license, unstable OSS architecture, week OSS reusability.

6. Conclusion

In this study we analysed open source projects from the closed source software development point of view. We presented open source project structure its characteristics, and specialities compared to traditional software projects. Because of increasing interest in using open source software in commercial projects, following basic open source software usage strategies were identified: using, redistributing and reusing open source software. All strategies were presented in business process models, based on business process modelling notation - BPMN. Additionally risk-benefit analysis was performed on activities and events of each business model. Finally a comparative study, comparing all three models was performed, based on predefined attributes.

Future research might be directed into specifying cost models of specific usage strategies. Additional, empirically testable success factors should be defined for OSS that is commonly used in a specific usage strategy.

To summarize, open source software has a huge usage potential in commercial software development environment, where open source community acts as a resource of software developers and testers. Open source can supply commercial projects with software infrastructure, reusable components or products, which can be further commercially redistributed. However technical, managerial and legal aspects should be properly studied before deciding for a specific usage strategy.

References

[1] Andrea Bonaccorsi & Cristina Rossi "Contributing to OS Projects. A Comparison between Individual and Firms", in *Collaboration, Conflict and Control*, pp. 18-22.

[2] Boehm, B. 1999, "Managing software productivity and reuse", *Computer*, vol. 32, no. 9, pp. 111-113.

[3] BPMI. Busines Proces Modelling Notation ver 1.0. 2004. Busines Process Management Initiative (BPMI).
Ref Type: Generic

[4] Brown, A. W. & Booch, G. 2002, "Reusing open-source software and practices: The impact of open-source on commercial vendors", *Software Reuse: Methods, Techniques, and Tools, Proceedings*, vol. 2319, pp. 123-136.

[5] Eric S.Raymond 2000, "The cathedral and the bazaar", *Computers & Mathematics with Applications*, vol. 39, no. 3-4, p. 263.

[6] Fitzgerald, B. 2004, "A critical look at open source", *Computer*, vol. 37, no. 7, pp. 92-94.

[7] Gacek, C. & Arief, B. 2004, "The many meanings of open source", *IEEE Software*, vol. 21, no. 1, p. 34-+.

[8] Jen-Fang Lee & Tzu-Ying Chan 2004, "Organisational Structure of "User Collaboration Community": Insights from the Case of an Open Source Software Project", in *4th Workshop on Open Source Software Engineering*, pp. 105-109.

[9] Jianjun Deng, Tilman Seifert, & Sascha Vogel "Towards a Product Model of Open Source Software in a Commercial Environment", in *3rd Workshop on Open Source Software Engineering*, pp. 31-38.

[10] Kasper Edwards 2004, "An economic perspective on software licenses—open source, maintainers and user-developers", *Telematics and Informatics*.

[11] Madanmoban, T. R. & De, R. 2004, "Open source reuse in commercial firms", *Ieee Software*, vol. 21, no. 6, p. 62-+.

[12] Michael Nash 2003, *Java Frameworks and Components: Accelerate Your Web Application Development* Cambridge University Press.

[13] Open Source Initiative. Open Source Initiative - OSI - The Open Source Definition. 2005. 20-8-2005.
Ref Type: Generic

[14] Richard P.Gabriel & Ron Goldman 2002, "Open Source: beyond the Fairytales", *Perspectives on Business Innovation* no. 8.

[15] Ruffin, M. & Ebert, C. 2004, "Using open source software in product development: A primer", *Ieee Software*, vol. 21, no. 1, p. 82-+.

[16] Stephen A.White. Introduction to BPMN. July 2004. 2004. BPTrends.
Ref Type: Unpublished Work

[17] Vidyasagar Potdar & Elizabeth Chang 2004, "Open Source and Closed Source Development Methodologies", in *4th Workshop on Open Source Software Engineering*, pp. 105-109.

[18] Wolfgang Maass 2004, "Inside an Open Source Software Community: Empirical Analysis on Individual and Group Level", in *4th Workshop on Open Source Software Engineering*, pp. 105-109.

[19] Wu, M. W. & Lin, Y. D. 2001, "Open source software development: An overview", *Computer*, vol. 34, no. 6, p. 33-+.