Control Objectives in Open Source Projects

Kris Ven and Jan Verelst
University of Antwerp
Faculty of Applied Economics – Department of Management Information Systems
Prinsstraat 13, B-2000 Antwerp, Belgium
{kris.ven,jan.verelst}@ua.ac.be

Abstract

Many studies on Open Source Software Development (OSSD) have been published in the past years. In these studies, OSSD has received positive comments and is proposed as a new way of developing software. We investigate how project management is used in Open Source Software (OSS) projects to exercise control of development activities. Project management in OSSD is worth studying, because research has shown that project management is a critical success factor in traditional software development. We use the COBIT framework to audit project management practices commonly found in OSS projects with the aim of determining which practices are currently missing or can be further improved upon. The framework identifies several potential threats to the long term continuity of OSS projects.

1. Introduction

In most literature concerning Open Source Software Development (OSSD), the OSSD approach is presented as a new method of developing software, with the distinct characteristic that the source code of the project is freely available to everyone. Open Source Software (OSS) is frequently reported to be of equal or higher quality, compared to non-OSS [6,17,10]. Moreover, some studies have shown that the methodologies used in OSSD are based on best practices in software engineering [21], and that certain elements of the OSSD approach could be beneficial to the development of commercial software [1,10].

We certainly acknowledge these advantages of OSSD. We are however interested in how we can further increase the success rate of OSS projects. Research has shown that control exercised through project management is a critical success factor in traditional software development [2,5,15]. We think that these critical success factors can also be relevant for OSS projects. Therefore, we study project management in OSS projects. The aim of this paper is to show which concepts of control are currently present and/or missing in OSS projects. We will hereby make use of the COBIT framework, since this framework lists several control objectives for IT management. We will first briefly present the COBIT framework. Next, we will apply the COBIT framework to OSSD practices, and determine which processes are currently missing in OSS projects. Finally, we will discuss the implications of our findings for OSSD.

2. COBIT

COBIT stands for *Control Objectives for Information and related Technology* and is developed by the *IT Governance Institute* [14]. The aim of COBIT is to provide a framework that assesses the alignment between IT infrastructure and business processes and is an important tool in IT audit. Additionally, it provides guidelines for good management practices concerning IT.

The COBIT framework considers four different domains: *Planning & Organisation*, *Acquisition & Implementation*, *Delivery & Support* and *Monitoring*. For each domain, high-level control objectives are specified. There are a total of 34 high-level objectives, which can be further specified in 318 detailed control objectives. These control objectives have an impact on different *Information criteria* (effectiveness, efficiency, confidentiality, integrity, availability, compliance and reliability) and *IT resources* (people, applications, technology, facilities and data). Each objective can be relevant for one or more of these information criteria and IT resources. A schematic representation of the COBIT framework can be found in figure 1. The meaning of the different objectives (PO1 till PO11) can be found in table 1 and in section 3.

objectives	Information Criteria							IT Resources				
	effectiveness	efficiency	confidentiality	integrity	availability	compliance	reliability	people	applications	technology	facilities	data
PO1	P	S						Y	Y	Y	Y	Y
PO2	P	S	S	S				Y				Y
PO3	P	S								Y	Y	
PO4	P	S						Y				
PO5	P	P					S	Y	Y	Y	Y	
PO6	P				S			Y				
PO7	P	P						Y				
PO8	P					P	S	Y	Y			Y
PO9	P	S	P	P	P	S	S	Y	Y	Y	Y	Y
PO10	P	P						Y	Y	Y	Y	
PO11	P	P			P		S	Y	Y	Y	Y	

P = Primary, S = Secondary, Y = Applicable

Figure 1: The COBIT framework – Planning and Organisation

3. Application

In this section, we will apply the COBIT framework to OSSD. For our purposes, the *Planning & Organisation* (PO) domain is the most relevant. Hence, we will limit ourselves to the study of the 11 high-level control objectives that are defined for this domain. The conclusions we will present here are based on an exploratory study of OSS projects investigating which management practices are present or missing in OSS projects.

In some cases, we had to give the COBIT control objectives a more meaningful interpretation within OSSD context. This is due to the specific nature of OSSD. Nevertheless, our experiences show that even in its current form the COBIT framework is already capable of highlighting some fundamental issues in OSSD.

PO1 : Define a strategic IT plan This objective states the need to develop a strategic IT plan to maximise the fulfilment of the requirements and to ensure the further accomplishment of the project. Strategic planning should result in long-term plans which should be translated into operational plans and concrete short-term goals.

The importance of planning in ICT projects has also been illustrated in studies by Curtis et al. and Guinan et al. [2,5]. This means OSS projects as well should have a long term vision. More concrete, this could be in the form of a development *roadmap*. A short term planning is also required, for example, a concrete list of improvements and must-fix bugs for each upcoming release should be made so that each developer can concentrate his or her efforts on these issues.

The Mozilla project follows these guidelines, whereby the roadmap can be found on the project homepage, while some bugs can be marked as "blocking" a certain release. MySQL also has a roadmap in their documentation, illustrating which features should be implemented in which release.

PO2 : Define the information architecture This objective requires that all data available within a company is inventoried so that information use can be maximised, or in more simpler terms: ensuring that everyone in the organisation can access the data he or she needs.

Although primarily intended for management information systems, this guideline requires OSS projects to identify relevant information for developers and users. It will be important for example that everyone can easily access the latest source code (this is mostly done by CVS) and there is sufficient documentation for developers (on the software design) as well as for end users. Unfortunately, OSS projects generally provide little documentation, since it is commonly known that developers don't like to write documentation [16]. An example of this is ArgoUML of which several sections of the help file still have to be written. There are exceptions, such as OpenOffice.org which ships with an extended help program.

PO3 : Determine technological direction According to this objective, one has to fully maximise the potential of emerging technologies. New technologies can help to achieve the business goals or better support the current processes.

As OSS projects mostly involve IT enthusiasts, it seems reasonable to assume that they will be aware of new technological advantages and how they can be used in the development process. It is however doubtful that there are formalised processes to see how new technologies can help the development of the project. For changes in the technological infrastructure, money can be a limiting factor for OSS projects.

PO4 : Define the IT organisation and relationships This objective requires structuring the technical organisation of a project or company – where roles and responsibilities are clearly set for each position – and ensuring that these positions align with the business goals. This includes creating an organisational chart, setting job descriptions and staffing the positions.

This means that OSS projects should have at least a clearly defined structure of responsibilities, maybe in the form of an organisational chart. In certain projects such delegation of responsibilities is present. In the Mozilla project for example, there are module owners who are responsible for a specific part of the project. Although little in-depth research is performed on this topic, we assume that the forming of the organisational structure is an *ad hoc* process. In commercial enterprises, these structures usually form taking into account several contingencies in the environment [19].

PO5 : Manage the IT investment Managing investment costs include defining budgets, control of the actual spending and analysing and making investment decisions.

Little information is available on how OSS projects manage their finances. Since they are mostly dependent on gifts, the budgeting will probably consist of checking which infrastructure is needed and can be acquired (f.e. web servers).

PO6 : Communicate management aims and direction The goals and direction the organisation is heading (coming from PO1) need to be communicated throughout the entire organisation. Accompanying these communication channels are policies and practices employers should adhere to, in order to ensure the realisation of the goals.

This objective refers to how top management in enterprises communicate their goals to employees, to make sure everyone understands where the organisation is heading. In OSS projects we find a similar mechanism: the core team or project leader must be able to communicate the project priorities so that developers can focus their efforts accordingly.

Mailing lists, notices on the project website and IRC are possible channels through which the goals of OSS projects can be communicated to developers and users. Guidelines for developers can be explicit (f.e. coding standards published in the project home page) or implicit (rules that are learned by being involved in the project).

PO7 : Manage human resources Human resource management has to attract and motivate people in order to maximise their contribution to the project. This includes amongst others recruitment, compensation, training and promotion.

Looking at OSS projects, we see that in general, there is no formal HR management. Some informal means of promotion (such as taking on someone as core developer in a project) exist, but these promotions are solely based on efforts made in the past (the so-called *meritocracy* model of OSS projects [18]).

PO8 : Ensure compliance with external requirements This refers to which degree the project meets external obligations, such as legal and contractual restrictions.

An important goal for OSS projects is their compliance to (semi-)official standards set by organisations such as the World Wide Web Consortium (W3C). In general, OSS projects will try to use open standards where possible. On the other hand, OSS projects seem to frequently suffer from legal actions. Examples are the recent name change of Lindows to Lindash [9] and the name changing of Mozilla's Phoenix to Firebird and eventually to Firefox [4].

PO9 : Assess risks Risk management mainly concerns looking for ways to reduce threats – and therefore the risks – in the project. This consists of making an inventory of possible project risks and finding solutions.

A well-known source of risk is uncertainty about end-user requirements. This risk seems more limited than in traditional software development, since OSS is usually written by user/developers. A second source of risk are technical uncertainties. Unfortunately, little information exists on how risk assessments are made for OSS projects.

PO10 : Manage projects The management of projects is concerned with the timely and within-budget delivery of projects. This is basically project management in the strict sense of the word.

Several best practices of project management can be found in several OSS projects. The setting of milestones, setting quality targets and finding developers responsible for functional parts of the project are examples of this. Time and budget on the other hand, play a much smaller role in OSS projects. In general, there are no strict delivery dates that have to be respected. Therefore, the releases can be feature based. How this can lead to problems for commercial vendors was illustrated by the release of Netscape 6, which was based on Mozilla 0.6 [12]. Money is also a lesser concern, since most development in OSS projects is done by unpaid volunteers. Increasing manpower will therefore not lead to a high increase in cost.

PO11 : Manage quality Managing quality means ensuring that the software produced meets the expectations and requirements of the client or user. Quality control is an important part of any project.

Several studies have argued that OSS projects produce software of higher, or at least equal, quality compared to non-OSS projects [6,17,10]. Some of the tools and practices used in OSS projects, such as code review, seem to support this. In the Wine project this is taken to a higher level, where the priority of each core developer is code review. Each patch made by contributors must be sent by a mailing list where anyone can give comments on the patch. Only if it is certain that the patch will not introduce regression bugs, the patch is committed in the CVS repository [10]. Other quality assurance techniques are daily builds. Mozilla Tinderbox is an automated build system that compiles the program each day, and runs a series of tests to check whether regression bugs have entered in the last day.

Objective	Present
PO1 : Define a strategic IT plan	Y
PO2 : Define the information architecture	P
PO3 : Determine technological direction	P
PO4 : Define the IT organisation and relationships	P
PO5 : Manage the IT investment	N/A
PO6 : Communicate management aims and direction	P
PO7 : Manage human resources	N
PO8 : Ensure compliance with external requirements	P
PO9 : Assess risks	N/A
PO10 : Manage projects	P
PO11 : Manage quality	Y

Y = Yes, N = No, P = Partly, N/A = No data available

Table 1: Overview

4. Discussion

Table 1 shows an overview of our findings. In general, we find that most objectives are more or less covered in OSSD, although this is very different from project to project. Due to space limitations, we want to focus on three important observations.

First, we find that OSSD does not pay attention to Human Resource Management. This is at least a worrying fact, given the dependence of OSS projects on the *motivation* of their contributors. Several studies suggest that intrinsic rewards, such as peer recognition and personal satisfaction are the primary motivations of developers [7,3,11]. However, we feel there are several threats to this motivation. Some projects for example are heavily dependent on a project guru, such as Linus Thorvalds or Richard S. Stallman. The question is what will happen if these gurus leave the project. This will certainly be a problem in projects with a small number of core developers. Another example concerns the nature of the development task. While taking a look at OSS project archives such as Freshmeat or SourceForge, it shows that most of the active projects are still in development phase, where new features are added to the program. However, in traditional software engineering 50–70% of development efforts goes to maintenance [8,13], which is considered to be a less attractive task. It is therefore not impossible that when software becomes more mature, it will become more difficult to motivate people to work on maintenance. We hope that the factors that are motivating people at this moment to work on OSS projects, will continue to do so in the future. However, we feel that this motivation can be seriously threatened in the future by scenarios such as the two described above. Without capable and enthusiastic contributors, there is no future for OSS projects.

A second observation is that no matter how well control objectives, processes and guidelines are defined, the *effective control* of development activities still remains a problem. Most developers have no formal ties with OSS projects, and there is no way to force people to do a certain task [20]. This means that there is no way to force people to comply to certain guidelines that are set by the project team. It appears that in OSSD, control is largely based on *trust*. A project administrator for example must trust his co-developers and contributors that their contributions to the project will be worthwhile. A contributor on the other hand will need to know that the effort he will make in writing software for no fee, will be compensated by the benefits of using the resulting software. The role of trust in OSS communities has been downplayed by Gallivan [3], however we feel that some degree of trust will be present in OSS communities, albeit implicitly. An article by Markus et al. [11] further suggests that social control and self control are sufficient control mechanisms, and that company employees should even be managed as unpaid volunteers. In our opinion, Markus et al. are overly optimistic: currently, there is little evidence that these management practices will work in practice.

A third observation concerns the use of the COBIT framework. The need for an *audit framework* for OSS projects is large. Potential users could base their decision on it whether to adopt the software or not. Companies considering to fund OSS projects might use it to assess the stability or potential of the project. Our study shows that COBIT in its current form is capable of highlighting a number of fundamental issues in OSSD. Further research is required to determine whether audit frameworks tailored to OSSD are required.

5. Conclusions

In this paper, we applied the COBIT framework to OSSD in order to study how control is exercised by using project management. This leads us to three important conclusions.

First, a basic level of support for the high-level control objectives in the Planning & Organisation domain seems to be present in OSSD.

Second, COBIT identifies a number of threats to the long term future of OSS. We consider it possible that the initial motivation of developers will disappear as project gurus find new challenges and the programming work changes from developing exciting new systems to cumbersome maintenance. Furthermore, in such difficult times, effective control would enable the project team to lead the project in a new and better direction. However, it is this control that is missing. Practising good project management may result in certain improvements, but project management alone won't be sufficient if motivation decreases. More research on how motivation can be stimulated in OSS projects seems appropriate.

Research in the field of non-profit organisations relying on volunteer work might prove useful in this context.

Third, more research is required on audit frameworks for OSSD. They would be useful to users deciding to adopt a certain OSS project, as well as to companies assessing the project potential. Such frameworks should try to incorporate as much traditional management practices as possible, but take into account the specific nature of OSSD.

References

[1] A. W. Brown and G. Booch. Reusing open-source software and practices: The impact of open-source on commercial vendors. In C. Gacek, editor, *Proceedings: Seventh International Conference on Software Reuse*, volume 2319 of *Lecture Notes in Computer Science*, pages 123–136. SpringerVerlag, 2002.

[2] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31(11): 1268–1287, Nov. 1988.

[3] M. J. Gallivan. Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies. *Information Systems Journal*, 11(4): 277–304, 2001.

[4] S. Garrity, G. Markham, B. Goodger, B. Decrem, et al. Mozilla firefox – brand name FAQ. http://www.mozilla.org/projects/firefox/firefox-name-faq.html.

[5] P. Guinan, J. G. Cooprider, and S. Faraj. Enabling software development team performance during requirements definition: A behavioural versus technical approach. *Information Systems Research*, 9(2): 101–125, June 1998.

[6] T. Halloran and W. L. Scherlis. High quality and open source software practices. In *Meeting Challenges and Surviving Success: Second ICSE Workshop on Open Source Software Engineering*, Orlando, Florida, USA, May 25, 2002.

[7] J. Lerner and J. Tirole. Some simple economics of open source. *Journal of Industrial Economics*, 50(2): 194–234, June 2002.

[8] B. P. Lientz and E. B. Swanson. *Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations*. Addison-Wesley, Reading, MA, 1980.

[9] Lindash. http://www.lindash.com/.

[10] S. Lussier. New tricks: How open source changed the way my team works. *IEEE Software*, 21(1): 68–72, 2004.

[11] M. L. Markus, B. Manville, and C. E. Agres. What makes a virtual organization work? *Sloan Management Review*, 42(1): 13–26, 2000.

[12] Mozilla Project. Old mozilla milestone plan. http://www.mozilla.org/releases/milestones.html.

[13] J. T. Nosek and P. Palvia. Software maintenance management: changes in the last decade. *Journal of Software Maintenance*, 2(3): 157–174, 1990.

[14] IT Governance Institute. *COBIT framework*. Information Systems Audit and Control Foundation, 3 edition, July 2000.

[15] The Standish group. The CHAOS report, 1995.

[16] J. Sametinger and G. Pomberger. A hypertext system for literate c++ programming. *Journal of Object Oriented Programming*, 4(8): 24–29, 1992.

[17] W. Scacchi. Is open source software development faster, better and cheaper than software engineering? In J. Feller, B. Fitzgerald, F. Hecker, S. A. Hissam, K. R. Lakhani, and A. van der Hoek, editors, *Meeting Challenges and Surviving Success: Second ICSE Workshop on Open Source Software Engineering*, Orlando, Florida, USA, May 25, 2002.

[18] W. Scacchi. Free and open source development practices in the game community. *IEEE Software*, 21(1): 59–66, 2004.

[19] J. D. Thompson. *Organizations in action: social science bases of administrativetheory*. McGraw-Hill, New York, 1967.

[20] J. Warsta and P. Abrahamsson. Is open source software development essentially an agile method? In J. Feller, B. Fitzgerald, S. A. Hissam, and K. Lakhani, editors, *Proceedings of Taking Stock of the Bazaar: The 3rd Workshop on Open Source Software Engineering, the 25th International Conference on Software Engineering (ICSE)*, pages 143–147, Portland, Oregon, USA, May 3, 2003.

[21] L. Zhao and S. Elbaum. Quality assurance under the open source development model. *Journal of Systems and Software*, 66(1): 65–75, Apr. 15, 2003.