2009

Do best practice frameworks fit open source software customization?

Steffen Kebler
Philipps-Universität Marburg, steffen.kessler@wiwi.uni-marburg.de

Paul Alpar
Philipps-Universität Marburg, alpar@wiwi.uni-marburg.de

DO BEST PRACTICE FRAMEWORKS FIT OPEN SOURCE SOFTWARE CUSTOMIZATION?

Keßler, Steffen, Institute of Information Systems, Philipps-Universität Marburg, Universitätsstraße 24, 35032 Marburg, Germany, steffen.kessler@wiwi.uni-marburg.de

Alpar, Paul, Institute of Information Systems, Philipps-Universität Marburg, Universitätsstraße 24, 35032 Marburg, Germany, alpar@wiwi.uni-marburg.de

Abstract

Most management models that support software development, information technology (IT) service management, or other IT related tasks have been developed based on extensive experience with these tasks. Their recommendations are, therefore, often named base, good, or best practice. We examine in this paper whether these models are also suitable to support use and adaptation of open source software (OSS) within adopting organizations. OSS is widely used as it exhibits some very appealing features. However, if companies take full advantage of the flexibility OSS affords, especially the possibility to customize it, relating processes get quite complex. Careful control of these processes becomes crucial and use of proper management processes almost mandatory.

In the paper, we first determine the special needs of OSS customization and then examine three popular models as to how well they can support this activity. The examined models are ITIL Version 3, SPICE (ISO/IEC 15504), and V-Modell XT.

Our research shows that none of the models sufficiently covers the needs of OSS customization. While some aspects of OSS customization can be dealt with in the models or require only minimal modifications of the models, none of the models includes best practices for the management of intertwined and concurrent internal and external development.

Keywords: Open Source Software, best practice, customization, adaptation, software development.

1 INTRODUCTION

OSS is developed by many individuals and many organizations. An organization that uses and customizes OSS must carefully manage the development process in order to achieve high quality software (SW) and avoid errors. Frameworks or models for management of SW development were mainly designed for development under single control. Therefore, our research question is: Can models for SW development and use directly be employed by an organization that customizes OSS for its own purposes? In the first subsection, we briefly describe some aspects of OSS that are important in our case. Then, we briefly recall the rationale for software development models and we name three models that we want to examine in detail.

1.1 Open Source Software

The term OSS denotes SW which source code is available under a copyright license in compliance with the Open Source Definition (Open Source Initiative 2008). It has to be distinguished from the term Free Software which is defined by the Free Software Foundation (Free Software Foundation 2008) founded by Richard Stallman. It originated the popular OSS Licence GNU GPL. The Free Software Foundation places an emphasis on the freedom of the SW; for this paper we chose the term Open Source Software (OSS) to place an emphasis on the availability of the source code.

OSS has gained popularity mostly through Linux, an operating system project started by Linus Torvalds, but joined by many developers distributed all over the world (Raymond 2001). Linux was first popular with technically versatile persons but is now available in distribution packages that do not require extensive computer skills. Its use increased with the diffusion of Internet. One example of an OSS commonly used with Linux is the Apache web server [http://httpd.apache.org]. While both Linux and Apache offer general core functions, increasingly OSS for specific applications is gaining popularity, e.g., in the area of business intelligence The Eclipse BIRT (Business Intelligence and Reporting Tool) Project, [http://www.eclipse.org/birt/phoenix/]). Even Enterprise Resource Planning Systems are meanwhile available as OSS (e.g., SQL Ledger [http://www.sql-ledger.org/].

While OSS needs not necessarily to be available at no cost, it often is. Therefore, it is highly relevant for companies trying to lower licence fees. In fact, costs seem to be the main reason for OSS adoption (e.g., Dedrick and West 2004).

The development of OSS by many distributed developers is referred to as an OSS project. The core developers provide an “official” release. Adaptations between releases may be published by the core developers or anyone else. However, there may be also adaptations executed within an organization that are not made available to the public. Adaptations by firms outside of the SW industry seem to be rare. There are different reasons for the reluctance of organizations to modify OSS. First, they may not have a need to do so. Second, they may not possess the skills or capacity to do so (Madanmohan and De’ 2004). There is also a difference in attitude towards OSS adaptation depending on the type of OSS. Organizations are more willing to adapt application-oriented non-infrastructure SW than infrastructural SW (Wichmann 2002, Dedrick and West 2004). Both, adoption and adaptation of OSS may be hindered by licensing intricacies. Some OSS licenses require that all modifications are provided to the public. This is just the opposite of why companies customize standard SW: because they do things differently than others. They usually do not want to reveal their processes to others.

According to a recent citation of a study by Gartner (Morgan 2008), OSS is used in 85 percent of all companies, the other 15 percent planned to do so within the next 12 months. Given this state, whether organizations customize OSS or “just” use it without own adaptations, the question arises whether the models considered below support OSS use. In fact, a possible lack of adequate support may lead to problems or create an obstacle to a wider adoption and use of OSS.

Even if companies do not adapt OSS after adoption, the quick release of new versions of OSS, modifications that are not part of an official release, and possible forking in OSS projects require an active management of OSS use. Therefore, it is worthwhile examining whether the specific needs of OSS are covered by the models.

The deployment of OSS is similar to the deployment of commercial of-the-shelf SW (COTS). Both need to be customized to fit the requirements of the adopting organization. Configuration and parameterization are the approaches of choice in this case. If they are not sufficient, customization of the SW code is necessary. With COTS, customizations are limited to customizations utilizing Application Programming Interfaces (APIs) provided by the vendor. With OSS, anyone can modify the source code. This flexibility offers great advantages but also leads to more managerial complexity. Since research on OSS customization within organizations is rare, we first need to identify the different events in the lifetime of an OSS that can occur before we can analyze how well the models for SW development and use support OSS use.

Research on maintenance of OSS does exist. But it covers maintenance processes within the OSS projects (e.g., Koponen and Hotti 2005). To our knowledge, no research on the case of internal OSS maintenance in companies does exist. As a result, the interactions between the process of adaptations in companies and the ongoing development process within the OSS project have not received much attention yet. This intertwined process may already start with the deployment of OSS within the organization. In this context, it is especially of interest whether the below mentioned models can also support such important tasks as OSS configuration and change management within organizations.

1.2 Frameworks for Software Development and Use

SW development, maintenance, and its continuous use are intellectual tasks that are difficult to manage. Given the economic and legal importance of these tasks, they must be controlled and managed. A number of models or frameworks have been developed towards these goals. They should help to produce high quality SW efficiently (Sommerville 2007, pp. 692). Both of these characteristics have often been badly missed in SW development and SW products (Boehm 2006). Most of the management models have evolved based on the experiences gained in many SW projects in many organizations. Practices that led to desired outcomes have been gathered and further improved so that the resulting frameworks are considered to contain best practice. A best practice process denotes a process, which is the most effective and efficient process for a certain task. A bundle of such processes may become the standard for this specific task and can be applied in diverse organizations.

We chose for our research three frequently used frameworks to represent models for different purposes. They contain guidelines which partly relate to the same tasks (e.g., change management).

Any organization that develops SW systems should follow a system development model. Such models support project management from systems analysis through decomposition and integration including quality assurance. An example of such a model is the V-Model where activities are arranged according to the letter V (Sommerville 2007, pp. 110). The use of its German version, currently called V-Modell XT (V-Modell n.d.), is mandated for systems developed by or for the German federal administration (Lange 2009). Many private organizations make use of it as well.

The use of a SW system in an organization can be considered a service to its internal and external users. The framework called ITIL (Information Technology Infrastructure Library) has been developed with the goals of efficient delivery of such services and high service quality. ITIL is a collection of concepts and guidelines for IT planning and operations that are supposed to represent best practices [http://www.itil-officialsite.com].

A third type of models supports the assessment of the capability of an organization to develop high quality SW. The international standard ISO/IEC 15504 (ISO/IEC 15504-5 2006) constitutes an example of such a framework. This standard, often referred to as SPICE (Software Process Improvement and Capability Determination), defines processes grouped in process categories, capability levels which the processes can achieve, and an assessment model which guides the measurement of processes with respect to their capability. SPICE is especially relevant to SW vendors but can also be valuable to large organizations developing large programs for their own use.

It is not easy for an organization which wishes to achieve the right level of managerial control of IT use to choose the appropriate combination of frameworks given their number, their overlap, differences in terminology and structure (Paulk 2004). We address yet another question in this context as mentioned above: How well do these models and frameworks support the full use of OSS?

2 RESEARCH APPROACH

Given the scarcity of relevant research and the difficulty to collect detailed data through survey research, we decided to observe an OSS adoption and customization process long enough to be able to induct the managerial needs of these processes. Therefore, we chose to follow the paradigm of action research (Avison et al. 1999). Of course, this approach does not guarantee that we will discover all the events that take place in the life of customized OSS but if we observe enough release changes, we should be able to give an initial answer to our research question.

2.1 Action research

Action research is a qualitative participatory method. The researcher becomes part of the project team in order to study the object of research in all necessary detail. It is widely used in the discipline of information systems (IS) as an interpretive research method (Baskerville 1999, de Villiers 2005). Qualitative research in IS is often applied in new problem domains where empirical data is unavailable or available only in small sample sizes. The process of action research consists of four consecutive sub-processes: plan, act, observe, and reflect (Zuber-Skerritt 1993). Our plan consisted of decisions which information on OSS deployment and adaptations to record and what tools to use. Since proper documentation of code versions and changes is a necessary action in OSS use, this was in the same time our "action" part and observation activity. We archived source code as well as change and feature requests. During our research, seven iterations triggered by official releases of the OSS were examined. One external release was not incorporated into the internal code base by the development team. The collected documents were examined in the reflection phase.

Action research, with its root in the behavioral sciences and founded by Kurt Lewin and other researchers in the 1940s (Peters and Robinson 1984) uses a practitioner-researcher relationship and is participative (Avison et al. 1999, de Villiers 2005). The study of an existing product development over several iterations distinguishes this research method from other interpretative research methods, e.g., research during the creation of a new product (de Villiers 2005). Using action research, researchers participate as well as criticize in the process and, therefore, influence it. As a result, action research enhances the practice and is able to reveal new knowledge.

In addition to the study of the internal code development, we also studied code changes in the external OSS community that did not enter the official releases yet. All internal and all external releases were archived for ex-post examination. Minor internal releases were triggered by internal adaptations or non-official external code used in the internal code base; major internal releases were triggered by the aforementioned official external releases.

Close cooperation with the project team allowed us to gain insights regarding several code changes within the official code that were introduced into the internal code base and led to incompatibilities between the official and internal release. These were examined by the project team manually. Since new releases were never adopted completely, files with changes in the official release as well as in the internal code base needed to be updated manually.

2.2 OSS case

Most OSS projects are in constant flux. They follow the "release early, release often" approach (Golden 2005, pp.19-20) because they are not restricted by business practices and strategies of COTS.

New releases may include bug fixes and new features. The constant changes require formalized support processes (Hoyer et al. 2007). The OSS characteristic of continuous change certainly applies to the OSS system we studied. It is a special-purpose web-based OSS Content Management System (CMS) that includes some typical Web 2.0 features. It was adopted and adapted by a small entrepreneurial company. The OSS system was developed within the Pligg CMS project [www.pligg.com] starting in 2005. It was chosen for use in our case in 2006, when our observation immediately started. The CMS instance we studied is online at http://www.colivia.de.

Users of the system can submit news items that are placed in a waiting queue. The news item is usually a short reference to another website. If other users find the news interesting, they can comment on it and vote for it. News that receives votes above a certain limit is placed on the first page which gives it more exposure. Each vote is not necessarily equal. Its weight may depend on the previous activity of the user who is giving the vote. The exact vote calculation is usually not disclosed. The same is true for the calculation of a reputation score for each user. News items are tagged and assigned to groups. Most common tags within a certain time period are displayed in a tag cloud. Such sites are referred to as social news sites. Pligg has also been used for other applications. Some of the sites built with Pligg are http://www.dealigg.com, http://www.ecofuse.com and http://software.intel.com/sites/coolsw/. Pligg is used all over the world, 40,000 users are registered in its support forums.

3 EVENTS IN OSS CUSTOMIZATION

3.1 Software evolution

As a basis for an OSS evolution process in companies, we use the system evolution process (Fig. 1) proposed by Arthur (1988) and amended by Sommerville (2007, p. 540).

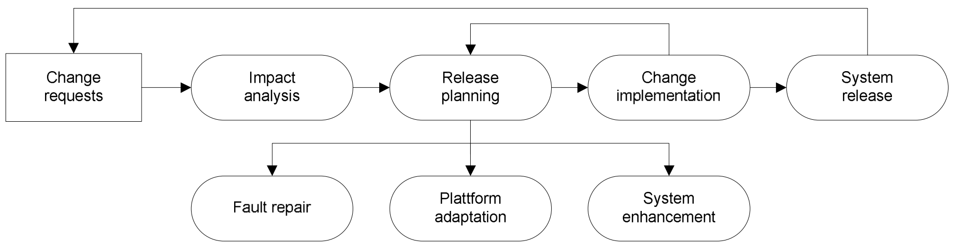

Figure 1. The system evolution process (Sommerville 2007, p. 540)

The SW evolution process is the result of a sequence of several maintenance iterations. Maintenance can be divided into adaptive, perfective and corrective maintenance (Swanson 1976). While corrective maintenance addresses errors found within the SW product, perfective maintenance is used to increase performance and maintainability as well as minimize inefficiencies. Adaptive maintenance is used to adapt SW to a changing environment. Maintenance often starts with the deployment of a SW product in an organization and stops with its retirement (Bennett and Rajlich 2000).

While maintenance has to be seen as a continuing process throughout the life of a SW product, this is contrary to the usual organization of SW tasks in projects. As a result, SW maintenance is usually organized as a sequence of maintenance projects (Kneuper 2003, p. 84f.).

Studies of SW that evolves over time (E-Type products) led to several research papers often referred to as "Lehman's Laws" (Lehman and Ramil 2001). They include the proposal of continuing change and increasing complexity during the process of use of a SW product as an E-Type product. Here, we focus on the interdependencies between the concurrent external and internal development of an OSS system as well as the influence of the external releases on the internal code base.

3.2 Events during OSS customization

Even before the deployment of a SW system within the company, required changes are identified, their impact is analyzed, the change is planned and carried out resulting in a new system; change requests after deployment initialize the same steps (Sommerville 2007, p. 540).

Based on our research of the OSS CMS customization, a distinction of two process chains in the evolution of an OSS within an organization can be made. Both process chains use the generic system evolution process from Fig. 1 as the basis but include different process steps to establish a new system release. This distinction is due to the differentiation of internal change requests, triggered from inside the company, and external change requests, triggered by external events, e.g., new official releases or bug fixes available in the OSS project's support forums. A firm may opt to ignore the second kind of change requests, but it then looses over time the free resources of the OSS developer community. Based on this process model, we identified several tasks in the areas of configuration and change management which are required in the case of entangled internal and external customizations of OSS.

Special requirements within the configuration management

Internal source code changes of OSS in a company require the establishment of an internal code base different from the code within the official OSS release. For OSS projects, a situation with two competing code bases would be called a fork; such a situation should be avoided to minimize inefficiencies (Raymond 2001, pp. 72-73). In OSS customization, the configuration management is required to manage the diverse code branches and minimize inefficiencies by supporting a merging of the internal code and new official releases.

For the internal code branch, the configuration management needs to keep track of both internally programmed code as well as imported external code (e.g., a bug fix available within the OSS project's support forums). Official code versions of Pligg, like in many other OSS projects, are stored in the version control system Subversion (SVN). External code which becomes available between two official releases in SVN, support forums or in another place, and is used for modifications of the internal code tree, has to be treated as internal code until it is integrated into the official release and the official release is integrated into the internal code base. As a result, a distinction between internal and external code is not sufficient in the internal documentation. The origin of external code that is used in the internal version but has not become part of the official release (yet) has to be recorded as well. Any modification to the internal code base may be the cause for incompatibilities between the internal and official release during the next update.

During the observation period, conflicts caused by incompatibilities of internal changes with changes in a new official release occurred in two iterations, while internal adaptations got obsolete in five iterations. As a result, incompatibilities had to be resolved by modifications to the internal code base if the official release were to be adopted.

Special requirements within the change management

In the life-cycle of the OSS CMS we studied, in case of an internal request for change external sources were always checked first to save efforts. The change management was responsible for the search, assessment and possible adaptation of appropriate code. The search included resources provided by the OSS project, in our case the OSS support forums and the SVN repository. Only if no suitable code from the OSS community was found, internal code was developed. In both cases, this new code remained in the internal release at least until a new official release became available. With a new official release available, internal modifications were checked against the new external code changes.

In general, OSS projects use diverse places to document changes made to the SW. OSS users will, therefore, usually need to monitor several places for relevant changes. In addition to communication tools, e.g., discussion forums and mailing lists, code changes are also entered into bug reporting tools (e.g., Bugzilla). In some OSS projects, users and developers may submit all kinds of source code changes, including updates, fixes, and sometimes even new features, rather than only bugs in such tools (Michlmayr, Hunt and Probert 2005, Koru and Tian 2004). As mentioned above, OSS projects often utilize revision control systems for configuration management purposes. Usually, all source changes will be tracked with these systems. Read access to revision control systems is usually unrestricted so they are an additional resource to search for relevant code.

The retrieval of information regarding new official releases was also a task assigned to the change management as well as the merging of the code bases in case of a new official release. The merging was supported by the configuration management with information about prior releases; also, the configuration management was responsible for the mapping of the branch merging.

We summarize the observations described above in Table 1.

Configuration management	**Change management**
Manage different versions of the same configuration element (separate branches)	Retrieve information regarding new releases from the OSS project (web site, news group, mailing list, etc.)
Build and archive baselines of relevant configuration items in case of the adaptation of external components	Assess external components; adopt if useful and adapt, if necessary
Support mapping of branch mergers	Merge internal and external branches in case of new official releases

Table 1. Additional tasks for maintenance in case of OSS customization

Code feedback

A task which may be assigned to the change management as well as to the configuration management is code feedback to the OSS project. It was not performed during our action research study, but code feedback can be beneficial to both the company that gives feedback as well as to the OSS project. If internal changes get integrated into the official release, official code can replace the corresponding internal code making later configuration management easier.

4 ANALYSIS OF FRAMEWORKS

All three models were examined based on original documents. The basic structure of each model is given first. Then, we report the results of our analyses referring to the specific chapters.

4.1 V-Modell XT

The V-Modell XT (V-Modell n. d.) is a SW development model widely used in the German government sector. It is divided into nine parts:

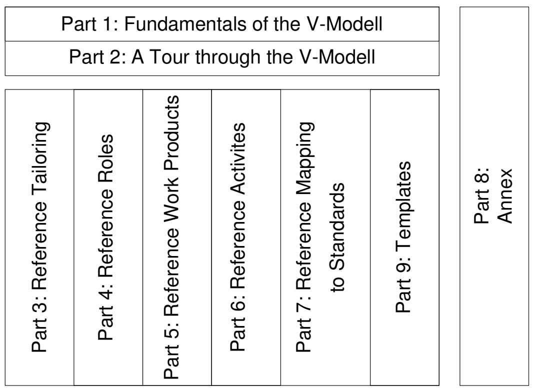
Part 1: Fundamentals of the V-Modell
Part 2: A Tour through the V-Modell
Part 3: Reference Tailoring
Part 4: Reference Roles
Part 5: Reference Work Products
Part 6: Reference Activites
Part 7: Reference Mapping to Standards
Part 9: Templates
Part 8: Annex

Figure 2. Structure of the V-Modell XT based on (V-Modell n.d., p. 1)

Within Part 4, relevant roles for configuration management and change management are defined, e.g., Configuration Management Administrator and Change Request Manager. The defined roles are sufficient for configuration and change management in the OSS context.

The view of the V-Modell XT is not limited to internal developments; it does incorporate the assessment and use of external products, even including market surveys of off-the-shelf products (Chapter 4.3 of Part 5). External SW modules are mentioned in Chapter 4.15 of Part 5. Links to chapters with further information within the standard are provided, including a link to make-or-buy decisions. Another relevant part for OSS customization is product evaluation (Part 6, Chapter 3.5); accordingly, external components can be evaluated before they are integrated into the internal code branch as non-official external code.

The possibility of code feedback is contained in the model, but it is based on the assumption that both the supplier and the buyer of SW products are using the V-Modell XT; feedback is provided in form of a Statement of Acceptance (Part 6, Chapter 5.1), which does not really fit in the case of OSS.

The V-Modell XT does support dividing of projects into partial projects (Part 3, Chapter 5.21), but there is no explicit support of ongoing concurrent development using separated code branches. For external SW units, an adoption procedure is defined in Chapter 3.8.12 of Part 6, which also does not reflect the concurrent development during the OSS life cycle. As a result, the handling of the code branches has to be represented via separate baselines or with the official code base carried within a baseline as a special configuration item (CI) within the configuration management. A baseline is a set of all relevant work products for one specific configuration.

In summary, V-Modell XT does include adequate practices for assessment and adoption of external components. It also offers some ideas which could be used for concurrent development in OSS projects in companies. However, it does not offer a consistent treatment of concurrent development.

4.2 IT Infrastructure Library (ITIL)

The IT Infrastructure Library (ITIL) is an example of best practices within IT service management.

They are grouped in functions or processes which are themselves assigned to five core volumes in ITIL Version 3:

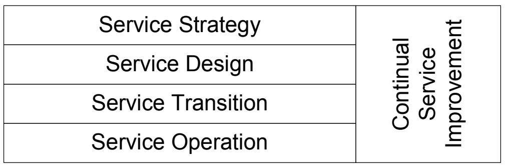

Figure 3. Core volumes of ITIL Version 3

Here, the volume Service Transition is especially relevant, as it "covers the broader, long-term change management role, release and deployment practices, so that risks, benefits, delivery mechanisms and the support of ongoing operational services are considered" (OGC 2008). The subsequent statements are based on the official German translation of the OGC Service Transition volume (OGC 2007).

Chapter 3 presents the Service Transition (ST) Principles; best practices and principles are contained here in several policies. Policy 3.2.2 confirms that all changes of service should be done by the service transition; the second best practice requires that internal and external changes have to be distinguished. While this is essential in case of OSS, according to our research it is not sufficient, as in the case of OSS customization with external code which is not part of the official release, three code types have to be distinguished. Policy 3.2.6 concerns the initiation and maintenance of relations to stakeholders. A best practice suggests building relationships with stakeholders (including suppliers). Since the OSS project can be considered a kind of supplier, this relationship can be the basis for feedback to the OSS project. According to a best practice in Policy 3.2.9, any update of a release has to be recorded within the configuration management system. This is in line with the required archival of all changes to CIs by the configuration management determined in our research of the OSS project. The inclusion of information about the origin of code is also essential.

Chapter 4 presents ST processes; several processes can be used in case of OSS use and customization in companies. In Chapter 4.1 (Transition Planning and Support) a release policy is detailed (4.1.4.2), recommending specific IDs for any release. It uses Major Release, Minor Release and Emergency Release as typical examples for release types. This concept needs to be extended for OSS use and customization to include a differentiation between internal and external release.

Several relevant topics are contained within Chapter 4.2 (Change Management). Considerations regarding the design and planning of the change management in 4.2.4.2 include the proposal of both a change advisory and emergency change advisory board as well as competences and responsibilities of stakeholders (e.g., suppliers). With suppliers as stakeholders within the change management, the OSS project may be defined as a stakeholder; as a result, code or other feedback to the OSS project can be covered via this stakeholder relationship. In addition, 4.2.6.7 (Review and Closure of Change Records) explicitly mentions feedback to the stakeholders and 4.2.7.3 includes Sourcing and Partnerships within the change management interfaces. Therefore, ST implicitly includes the option to give feedback to an OSS project. The kind of feedback is not specified, this may include revealing experience (e.g., in form of participation in OSS support forums or submission of bug reports) or actual code submissions to the OSS project. In Chapter 5 (ST Common Operation Activities), the introduction of a Stakeholder Management Strategy with respect to suppliers of services and products is mentioned in 5.3.

Chapter 4 also includes definitions of changes, with standard changes defined in 4.2.4.5. A new official OSS release shares several criteria with a standard change, especially a defined trigger. Without a standard process for merging of the new official version within the external code branch with the internal code branch, this particular change will not share the criterion *low risk* with the standard change and, therefore, remain a special kind of change every time. ST does not provide actual implementation practices for changes; therefore, a detailed change process for OSS needs to be specified within the company.

ST defines typical roles in Chapter 6. Service Asset and Configuration and Change Management are combined as one responsibility; six roles are defined as typical: Service Asset Manager, Configuration Manager, Configuration Analyst, Configuration-Administrator/Officer, Change Management System/Tool-Administrator and Change Manager. Additional tasks and responsibilities can be assigned to these roles for internal use and customization of OSS without need for additional roles.

The configuration management system is specified in great detail in chapter 4, where examples for configuration splitting are given. ST does not, however, specify best practices for the handling of separate code branches in which the same OSS version is concurrently modified by different development teams. Like in the V-Modell XT, the different code branches can be handled as different baselines, or the official code could be defined as a specific CI. The required support for code merging in OSS customization projects is not included within the ST.

In summary, the adaptation process can in part be structured on recommendations of the ITIL framework, but ITIL does not provide advice for the case of concurrent development of the same base version of SW in different locations (i.e., within the OSS project and the company). Copies of the same CI are only mentioned in Chapter 7.3 (technological considerations of the configuration management system), but it is unclear whether this includes separate copies of the same CI.

4.3 ISO/IEC 15504 (SPICE)

The standard ISO/IEC 15504 (SPICE) is an example of a quality management and assessment standard. It incorporates ISO 12207 AMD1 and AMD2, a process reference model for SW life cycle processes. The assessment model consists of 48 processes distributed over three process categories with one or several process groups within these categories (ISO/IEC 15504-5 2006, pp. 3).

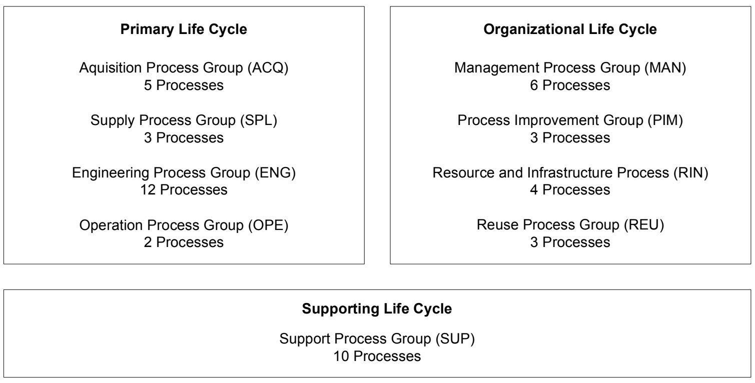
Primary Life Cycle
Aquisition Process Group (ACQ)
5 Processes
Supply Process Group (SPL)
3 Processes
Engineering Process Group (ENG)
12 Processes
Operation Process Group (OPE)
2 Processes
Organizational Life Cycle
Management Process Group (MAN)
6 Processes
Process Improvement Group (PIM)
3 Processes
Resource and Infrastructure Process (RIN)
4 Processes
Reuse Process Group (REU)
3 Processes
Supporting Life Cycle
Support Process Group (SUP)
10 Processes

Figure 4. SPICE Process Categories and Groups based on (ISO/IEC 15504-5 2006, p. 4)

SPICE includes three process categories Primary Life Cycle, Organizational Life Cycle, and Supporting Life Cycle. The last process category covers configuration management and change management which are of special interest here. Individual processes within Supporting Life Cycle Processes are grouped in Support Process Groups (SUP). SUPs can be utilized by processes within the Primary Life Cycle and Organizational Life Cycle. For maintenance, SUP.8 (configuration management) and SUP.10 (change request management) are utilized by ENG.12 (software and system maintenance) within the Engineering Process Group.

Ten base practices (BPs) for configuration management are defined for SUP.8. Several BPs cover topics also found within ITIL ST, e.g., configuration management elements (BP 2), baselines (BP 4) and the archival of changes of every configuration management element (BP 7, BP 10). However, BP 3 and BP 8 cover areas relevant to OSS use and integration not found within ITIL ST. BP 3 relates to the development of a branch management strategy. As part of the development of a configuration management, it covers the handling of more than one code branch resulting in developers working on separate copies of the same configuration element. For this case, special quality requirements are in place. Reviews are mentioned as an example of a possible method; the branch management can be supported by configuration management tools. BP 8 covers reports on the configuration status while BP 9 covers the verification of configuration elements. These BPs are especially relevant in case of integration activities and if parallel development or variants are used (Hörmann et al. 2006, p. 198-199). The demand for a verification of the correctness of a baseline emphasizes the particular relevancy of verification in this case.

The actual changes are reflected in SUP 10 (change management). Change management as a support process includes here BPs that encompass the change management process from the development of a change strategy until the review of the actual change. Several BPs focus on the handling of change requests, which are also defined as one of the work products (ISO/IEC 15504-5, p. 55). Within the BPs, there is no explicit reference to external change triggers. Also, none of the BPs includes the explicit possibility of assessment and inclusion of external code.

4.4 Comparison of frameworks

V-Modell XT provides adequate practices for the assessment and adoption of external components but it is unable to represent the intertwined development processes that occur in case of customization of OSS in companies as a whole. Feedback processes are focussed on the traditional understanding of a buyer-supplier relationship.

ITIL covers the areas of OSS customization that are consistent with the customization of closed-source SW, but it does not provide appropriate best practices for the management and merging of code branches resulting from concurrent development in case of OSS customization. Some practices exist as basic approaches but they need to be modified or expanded to match the requirements of OSS customization, e.g., with the introduction of the OSS project as a new stakeholder.

SPICE is less comprehensive in the areas of change management and configuration management; nonetheless, BP 3 within the configuration management support process recommends developing a strategy for branch management, which is essential for OSS customization. SPICE proposes BPs for the relevant process areas but it does not include examples of typical cases as ITIL does.

5 SUMMARY

The possibility to modify the source code in order to custom tailor it to the needs of the organisation is one of the most unique features of OSS. Through action research of the use and customization of an application OSS over several official releases and internal modifications, we identified differences between the customization process of closed-source SW and OSS. These differences were mainly caused by the concurrent development in the OSS project and within the company and by the possibility of use of externally available non-official code.

Our literature review of selected frameworks on good practices in SW development and use revealed that the process of customization of OSS is not sufficiently covered by these frameworks.

Some tasks that were performed within the OSS CMS customization we studied could not be found in any of the frameworks considered, e.g., the differentiation of code according to its origin (internal code, external non-official code, or official release code).

The overall economic benefit of OSS cannot be disputed (e.g., MERIT 2006). If more companies outside of the SW industry would take advantage of its feature of open source, not just of low costs, the benefits for the whole economy would further rise. Companies may be more willing to choose this option if they get adequate support in managing the process of OSS customization. Such support can come from best practice frameworks that are increasingly being used if the frameworks get extended by practices needed in OSS customization.

While our research of OSS customization is limited by the analysis of a single project, it is sufficient to reveal difficulties if ITIL, SPICE, or V-Modell XT are used as they are to manage OSS customization in companies. According to our results, users who want to customize OSS inside a company cannot rely only on best practices included in the observed frameworks. These frameworks particularly do not cover the concurrent and intertwined process of customization which is characteristic for internal customizations of OSS. Extensions of the frameworks have to be made to cover the complete process. We have indicated in the previous section some directions for extension.

Like in SW testing where a test can only detect errors but not prove the correctness of a program, the experience with one OSS customization can just indicate some problems of the frameworks but not necessarily all of them. Therefore, more research on OSS customization in companies is needed to establish best practices in this area.

References

Arthur, L.J. (1988). Software Evolution. John Wiley & Sons, New York, USA.

Avison, D.; Lau, F.; Myers, M.; Nielsen, P. A. (1999). Action Research. Communications of the ACM, 42 (1), 94-97.

Baskerville, R. L. (1999). Investigating Information Systems with Action Research. Comm. of the AIS, Volume 2, Article 19, October 1999.

Bennett, K. and Rajlich, V. (2000). Software Maintenance and Evolution: a Roadmap. Proceedings of the Conf. on The Future of Software Engineering, Limerick, Ireland, 73-87.

Boehm, B. (2006). A view of 20th and 21st century software engineering. Proceedings of the 28th Int. Conf. on Software Engineering, Shanghai, China, 12-29.

de Villiers, M.R. (2005). Three Approaches as Pillars for Interpretive Information Systems Research: Development Research, Action Research and Grounded Theory. Proceeding of SAICSIT, 141-151.

Dedrick, J. and West, J. (2004). An Exploratory Study into Open Source Platform Adoption. Proceedings of the 37th Annual Hawaii Int. Conf. on System Sciences (HICSS'04), Hawaii, USA, Track 8.

Free Software Foundation (2008). The Free Software Definition. URL: http://www.gnu.org/philosophy/free-sw.html, Download 2008-11-28.

Golden, B. (2005). Succeeding with Open Source. Addison-Wesley, Boston, USA.

Hörmann, K.; Dittmann, L.; Hindel, B.; Müller, M. (2006). SPICE in der Praxis: Interpretationshilfe für Anwender und Assessoren. d.punkt Verlag, Heidelberg, Germany.

Hoyer, V.; Schroth, C.; Stanoevska-Slabeva, K.; Janner, T. (2007). Web 2.0-Entwicklung – ewige Beta-Version. HMD – Praxis der Wirtschaftsinformatik, 255, 78-87.

ISO/IEC 15504-5 (2006). ISO/IEC 15504-6:2006(E): Information technology – Process Assessment – Part 5: An Exemplar Process Assessment Model. First Ed. 2006-03-01. ISO, Switzerland.

Kneuper, R. (2003). CMMI. 1st Ed., d.punkt Verlag, Heidelberg, Germany.

Koponen, T. and Hotti, V. (2005). Open Source Maintenance Process Framework. Proceedings of the Fifth Workshop on Open Source Software Engineering, St. Louis, MO, USA, 30-34.

Koru, A.G. and Tian, J. (2004). Defect Handling in Medium and Large Open Source Projects. IEEE Software, 21 (4), 54-61.

Lange (2009). Erfahrungen zum V-Modell XT, BIT 7, Köln, Germany. URL: http://www.hisolutions.com/33665/level2/hiscout_daten/VMXT-Erfahrungen-IT-Standards-DrChristianLange_33703.pdf, Download 2009-04-08.

Lehman, M.M. and Ramil, J.F. (2001). Rules and Tools for Software Evolution Planning and Management. Annals of Software Engineering 11, 15–44.

Madanmohan, T.R. and De', R. (2004). Open Source Reuse in Commercial Firms. IEEE Software, 21 (6), 62-69.

MERIT (2006). Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU. Final report, 2006-11-20, UNU-MERIT, the Netherlands. URL: http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf, Download 2008-11-29.

Michlmayr, M., Hunt, F. and Probert, D. (2005). Quality Practices and Problems in Free Software Projects. Proceedings of the First Int. Conf. on Open Source Systems, Genova, Italy, 24-28.

Morgan, T.P. (2008). Gartner: open source software 'pervasive'. In: The Register (Online), URL: http://www.theregister.co.uk/2008/11/18/gartner_open_source/, Download 2008-11-27.

OGC - Office of Government Commerce (2007). Service Transition. German Translation. The Stationary Office, Norwich, UK.

OGC - Office of Government Commerce (2008). Service Management - ITIL® Version 3 Publications. URL: http://www.best-management-practice.com/Publications-Library/IT-Service-Management-ITIL. Download 2008-11-29.

Open Source Initiative (2008). The Open Source Definition. URL: http://opensource.org/docs/osd, Download 2008-11-28.

Paulk, M. C. (2004). Surviving the Quagmire of Process Models, Integrated Models, and Standards. Proceedings of the Annual Quality Congress, Toronto, Ontario, Canada, 429-438.

Peters, M. and Robinson, V. (1984). The Origins and Status of Action Research. The Journal of Applied Behavioral Science, 20 (2), 113-124.

Raymond, E.S. (2001). The cathedral & the bazaar. Revised Edition. O'Reilly, CA, USA.

Sommerville, I. (2007). Software Engineering. 8th Edition, Pearson Studium, München, Germany.

Swanson, E.B. (1976). The dimensions of maintenance. Proceedings of the 2nd Int. Conf. on Software Engineering, San Francisco, CA, USA, 492-497.

V-Modell - Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (n.d.). V-Modell-XT. URL: http://v-modell.iabg.de/dmdocuments/V-Modell-XT-Complete-1.2.1.1-english.pdf, Berlin, Germany, Download 2008-10-25.

Wichmann, T. (2002). Free/Libre Open Source Software. Berlecon Research GmbH, Berlin, Germany. URL: http://www.berlecon.de/studien/downloads/200207FLOSS_Use.pdf, Download 2007-10-15.

Zuber-Skerritt, O. (1993). Improving Learning and Teaching Through Action Learning and Action Research. Higher Education Research & Development, 12 (1), 45-58.