# Usability Engineering Processes and Open Source Software Development

Hartti Suomela

***Abstract*—Open source development model is very different from traditional process model of software engineering and therefore the traditional processes and engineering activities are seldom used in open source projects. Likewise usability engineering in traditional software development projects has evolved to a fairly structured – albeit iterative – set of activities. To find out if the usability engineering practices used in traditional software engineering could be used to overcome the usability hurdles experienced in many open source products, we provide an overview of each of these models and study the applicability of well-structured usability engineering in open source projects using Usability Maturity Model (ISO/IEC 18529) process definition as a framework. We also briefly introduce the various efforts to make open source products more usable.**

***Index Terms*—open source software, software engineering, usability, usability engineering, usability maturity model, UMM**

## 1. Introduction

Open Source (OS) software communities have developed a number of hugely popular applications in the past ten years. In some areas the OS applications even dominate the market. Take for example Apache Web server. In October 2004 this OS server had close to 68 percent market share across all domains, whereas the closest competitor Microsoft IIS (Internet Information Server), a proprietary solution, had a little over 21 percent of the market.[1] Other good examples of successful OS software products are the Linux operating system – with its many flavors[2] – and Mozilla Web and e-mail applications suite.

In general, the OS applications fare well against proprietary solutions when functionality, reliability, or efficiency is used as measurement. However the usability of OS applications has traditionally perceived to lower than the usability of proprietary solutions (see for example Nichols and Twidale, 2002, Benson et al., 2004b, and Thomas, 2002).

The less than optimal usability of the OS products was not an issue in the early days of OS movement, as long as the programmers were developing the products for themselves or for people like them (technically-oriented programmers and administrators). But when developing applications targeted for broader use – so called end-user applications – the usability plays much bigger role.

In this paper we study how well defined and formal usability engineering practices and processes could be applied to OS projects. We start with briefly defining usability and usability engineering, and continue then to provide an overview of the usability in OS projects. Next we describe the usability process model using Usability Maturity Model (UMM) as a framework. To counterbalance we also provide an overview of the structure and practices in OS development. We conclude with elaborating if the process model of UMM and its base practices could be applied in OS development projects.

## 2. Usability Defined

In software engineering community the term usability is very often associated with user interface design only. This kind of definition is a limited view on usability, which in general means making products and systems easier to use, and aligning them more closely to the needs and requirements of the user.

To avoid ambiguities in this paper we spend some effort in defining the terminology around usability. We use the definitions in ISO/IEC standards (as summarized by Bevan, 2001) and a definition by Nielsen (1993). Our definition for **usability** consists of the attributes of the product or system as seen by the user.

The ISO/IEC standard 9241-11 (1998), which provides guidance on usability, defines usability to be *the extent, to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*. This definition is further used in related ergonomic standards, although some software process related standards define usability in somewhat different way.

Another often used and simple definition for usability is from Nielsen, who describes usability to have the following five attributes (Nielsen, 1993):

- Ease of learning
- Efficiency of use
- Memorability
- Low error rate with no catastrophic errors
- Subjective satisfaction.

To achieve good usability, different tools, methods and practices are used. **Usability engineering** and **human-centered design**[3] define an approach to product development

[1] Netcraft Web Server Survey for October 2004, http://news.netcraft.com/archives/web_server_survey.html, accessed October 13th, 2004.

[2] Examples of well-known Linux distributions are Debian GNU, Mandrake, MontaVista, RedHat, and SuSE. For more complete list, please see http://www.linux.org/dist/index.html

[3] In some literature the term "human-centered design" is replaced with a similar term "user-centered design". The terms have, however, slightly

that incorporates direct user feedback throughout the development cycle in order to reduce costs and create products and tools that meet user needs. These terms are used interchangeably in this paper and – as defined in ISO/IEC 13407 – they are characterized by:

- The active involvement of users and a clear understanding of user and task requirements
- An appropriate allocation of function between users and technology
- The iteration of design solutions
- Multi-disciplinary design.

It is important to note that besides usability attributes, the products and systems have other attributes, which together make the capability of the system. These attributes include for example functionality, reliability, maintainability, and portability. In ISO/IEC 9126-1 (2000) all these attributes are combined together in term **quality in use**, which is defined to be *the capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use*.

## 3. Open Source Usability

A general note is that traditional usability engineering requires more structured approach than what is typically possible among OS communities, and therefore required usability practices have not been usually carried out in OS projects.

On the contrary, OS development tends to be very informal. Requirements are not typically gathered systematically, but the projects use "informalisms" as a lightweight method for directing the development activities (Scacchi, 2002). The projects are self-guiding, meaning that the OS developers choose by themselves which parts of the program they want to tackle without nobody really coordinating the development process or selecting the precedence for activities. Prototyping – as understood in traditional software and usability engineering – happens seldom, as the projects often start with a developer starting to code the real application in response to his needs of to the needs of his peers.

Nichols and Twidale (2002) have presented a comprehensive list of reasons for the usability problems in OS applications:

- In general, usability work should be started before any coding; but most of the OS projects start with coding.
- Usability experts do not get involved in OS software projects.
- OS projects lack resources to undertake high quality usability work.
- Developers are not typical end-users, but they develop for themselves (because they know what they want).
- OS development communities give better incentives for creating more functionality than better usability.
- Usability problems are hard to specify and they involve many parts of the products, whereas adding more functionality is easy, thanks to usually modular architecture of the OS applications.
- Developers add too much functionality, and do not remove anything (thanks to peer pressure), which bloats the software.
- Developers choose often power over simplicity.
- OS software user interfaces play catch-up to commercial software.

However there are useful practices in OS development, which could benefit the traditional software and usability engineering. An example is how bugs – including usability bugs – are handled and corrected in OS projects. Another good quality of the OS projects is the openness of the development and all the discussions enabled through public mailing lists and newsgroups, which provide a wealth of research material for software and usability engineering researchers.

It is notable that although OS philosophy is to release early and release often (Raymond, 1999), which should fit well in usability engineering practices benefiting from rapid iterative prototyping, usability assistance is usually sought only after interfaces have been "designed" (Benson et al., 2004b). Benson et al. also discuss the potential for large corporations to contribute to OS projects, especially providing usability know-how through hired usability professionals.

During the past few years the OS community has made progress in creating more usable products. Many OS projects have tackled the usability challenges by doing usability evaluations (for example GNOME as described by Smith et al., 2001), writing usability and user interface guidelines (for example Human Interface Guidelines for GNOME by Benson et al., 2004a), user interface specifications and specification proposals (see for example OpenOffice.org, 2003, and Mozilla.org, 2003), recruiting usability experts, and also activating the user base to give feedback on usability issues. In addition, at least one dedicated usability community for OS projects has been established (see below, section 3.1.1).

### *3.1 Resources for Improving Usability in OS Projects*

Many OS projects have recently added effort on the usability engineering side. For example on SourceForge.net[4] Web site OS projects can search for help – including usability – to volunteer time in their projects. However looking for help on usability does not seem to be very common through this medium, as only 3 out of the 186 volunteer openings advertised on this marketplace (SourceForge.net, 2004) were for usability/UI experts.

In addition to this kind of services available on general OS

different meanings, with HCD including also other stakeholders than just end-users into consideration.

[4] SourceForge.net is a repository of OS code and applications (current number of OS projects at SourceForge.net is over 91,000), and the site also provides free services to OS developers.

Web sites, there are a number of resources, which could help in making OS projects more usable. These sites are briefly described below.

*3.1.1 OpenUsability.org*

The mission statement of OpenUsability.org is *to bring open source developers and usability experts together*.

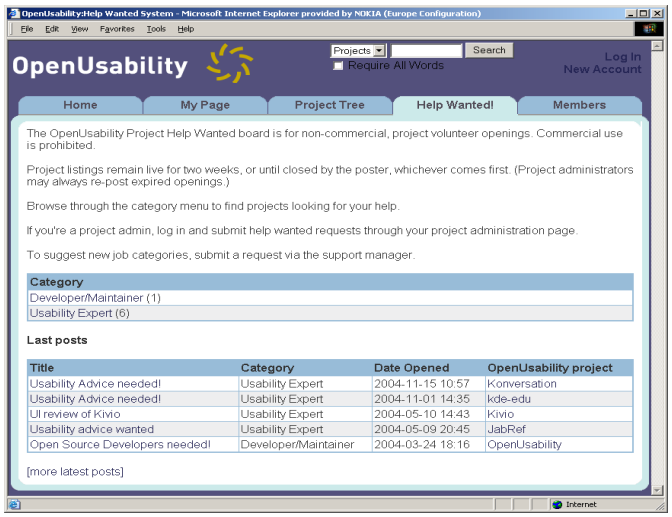


**Figure 1: OpenUsability.org Web site is a marketplace for usability experts and OS projects needing help on usability.**

Although not completely supported by research (Nichols and Twidale, 2002) the site claims that there are many usability experts who want to contribute to OS software projects, and that there are many developers who want to make their software more usable.

The site provides a place for open source projects to look for help to tackle the usability issues. On the other hand the site also provides a place for the OS inclined usability experts to specify their skills and interests. The system allows projects to rate the usability experts for their teamwork, coding, designing, reliability, and leadership abilities.

Besides being a marketplace between usability experts and OS projects, the site has started to build a virtual library of usability and OS related literature, articles, and research. The community has also generated a general XML specification for XML-based usability report language.

OpenUsability.org was launched in early 2004 and as of November 17th, 2004 there were 13 projects[5] and 105 profiles of usability experts listed on the site. Only 19 of those experts advertised participating in any project and on average those experts participated in approximately 1.7 projects. None of the experts were rated by the projects for their abilities.

Because of the lack of project feedback, low activity, and low participation numbers it is hard to estimate if the site has substantially helped open source projects to recruit usability experts. Also this kind of model is somewhat contradictory to the natural open source model, where developers gravitate towards tasks they are interested in and they can contribute to.

*3.1.2 FOSSUL*

FOSSUL (Free and Open Source Software Usability Laboratory, http://fossul.org) appears to be world's only research laboratory dedicated to usability of free and open source software. The laboratory is located in Indiana University School of Informatics.

Judging by the outdated site and the lack of published material by the laboratory personnel in ACM, IEEE, and Elsevier ScienceDirect, this research facility does not seem to be very active. Thus it is not clear if this facility has had any impact on advancing the usability of OS development projects.

*3.1.3 UsabilityNet*

One can argue that the tools and material provided on the UsabilityNet Web site are not tools for OS projects per se, but instead they are tools for the general audience interested in usability and usability engineering. However the easy access to the tools as well as the high quality of the available information makes the site a very valuable toolset also for OS development projects.

Among other things the site provides an interactive tutorial on different usability engineering methods (see Figure 2) needed during usability engineering lifecycle. Using this Methods table, the volunteers involved in usability aspects in OS projects can easily select the most suitable tools for the projects based on the resources, skills, and the user access available in their projects.

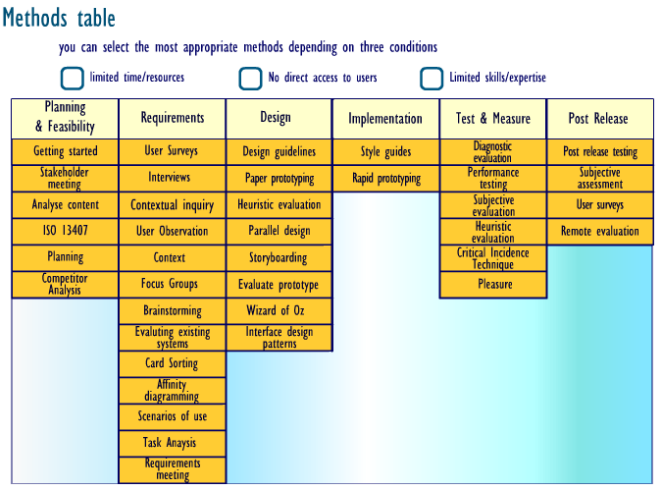


**Figure 2: The Methods table on UsabilityNet site provides easy access to usability engineering methods and their descriptions.**

[5] 10 separate OS projects, one project for the OpenUsability.org site itself, one project to compile the usability/OS bibliography/virtual library, and one project to compile a general usability resource library.

## 4. Overview of Process Models and Capability Models in Traditional Software Development

### *4.1 Modeling the Software Development Process*

Since the inception of large software systems in the 50s and 60s process models for software engineering have been developed. The familiar waterfall model of software life cycle (Royce, 1970) describes an initial set of activities needed to complete a software development project, and it also depicts the fact that the development process requires iterations and rework before completion.

The set of activities included in the software development process model has been later refined and they usually include a subset of following activities (Scacchi, 2001):

- System initiation/planning
- Requirement analysis and specification
- Functional specification or prototyping
- Partition and selection
- Architectural design and configuration specification
- Detailed component design specification
- Component implementation and debugging
- Software integration and testing
- Document revision and system delivery
- Deployment and installation
- Training and use
- Maintenance.

In general the software engineering has evolved and standardized to be quite well defined and measurable. The maturity of the software development processes of an organization can be measured using maturity models like CMM (Capability Maturity Model). The later models such as CMMI (Capability Maturity Model Integration) and SPICE (Software Process Improvement and Capability dEtermination, also ISO/IEC TR-2 15504:1998) allow organizations to measure all processes separately and also to measure the improvements in its processes. In general the models define a set of processes for software development. Each of the processes has a set of required base practices, which in return produce a set of required deliverables.

In this kind of software engineering model the activities for usability engineering have been distributed throughout the model and following these guidelines does not guarantee good usability. For example CMMI does not impose any requirements for usability. Even a development organization at the highest level of maturity may produce products with usability problems, although CMMI does include “hooks” where usability activities fit (Jokela and Lalli, 2003).

Similar analysis of usability against the SPICE framework has not been made, although a brief study by the author did not reveal any major differences between SPICE and CMMI in regards to usability (ISO/IEC 15504, 1998).[6]

[6] Interestingly, Part 5 of SPICE specification elaborates the requirements for constructing an assessment tool, including also discussion of the usability aspects and requirements of the assessment tools.

### *4.2 Modeling the Usability Engineering Process*

As the software engineering and maturity models do not provide adequate support for usability engineering, separate process models have been created.

To ensure the usability and quality in use of a product, the usability engineering activities ideally need to be planned and to take place throughout the lifecycle of the product, with significant activities happening at the earliest possible stages before the user interface design has even been started.

One overview of the human-centered design (HCD) process as described in ISO 13407 is provided in Figure 3. In is notable that there are other, more detailed descriptions available for the usability engineering process and in some models the usability tasks are grouped in somewhat different way. As the grouping and categorization of the usability engineering tasks is not in the scope of this research, the framework from ISO 13407 is used in the paper. For more detailed, and slightly different view of the usability process reader can refer to for example Jokela (2004b).

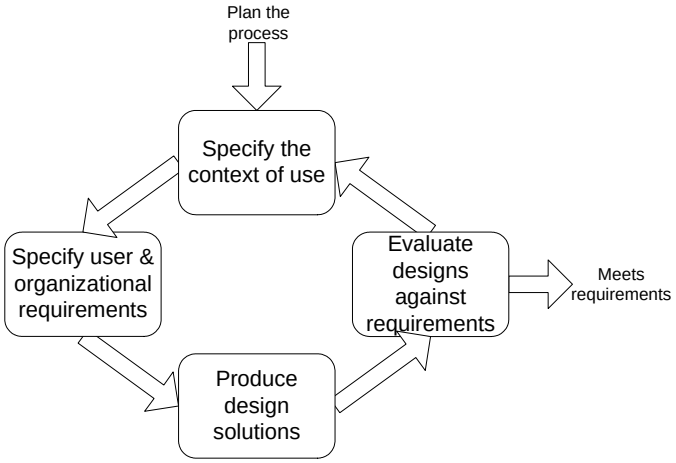


**Figure 3: The iterative human-centered design process as described in ISO13407.**

The HCD activities start with a planning phase, during which the role and extent of the usability engineering in the development project are defined.

After careful planning the next step is to define the context of use for the system, including defining the intended users, collecting information on their tasks, also identifying the technical and environmental constraints. Scenarios can be used to document descriptive examples of how an individual user using the system achieves a specific outcome while in certain context.

Following the context analysis it is necessary to gather and specify the requirements of the users and organizations involved in the lifecycle of the application. In some cases this includes evaluating an earlier version of the system or a system developed by a competitor to identify the usability requirements.

**Table 1: Human-centered design processes and their base practices as defined in ISO 18529 (ISO 13407).**

| Processes | Base practices |
|---|---|
| 1 Ensure HCD content in system strategy | 1.1 Represent stakeholders<br>1.2 Collect market intelligence<br>1.3 Define and plan system strategy<br>1.4 Collect market feedback<br>1.5 Analyze trends in users |
| 2 Plan and manage the HCD process | 2.1 Consult stakeholders<br>2.2 Identify and plan user involvement<br>2.3 Select human-centered methods and techniques<br>2.4 Ensure a human-centered approach within the team<br>2.5 Plan human-centered design activities<br>2.6 Manage human-centered activities<br>2.7 Champion human centered approach<br>2.8 Provide support for human-centered design |
| 3 Specify the stakeholder and organizational requirements | 3.1 Clarify and document system goals<br>3.2 Analyze stakeholders<br>3.3 Asses risk to stakeholders<br>3.4 Define the use of the system<br>3.5 Generate the stakeholder and organizational requirements<br>3.6 Set quality in use objectives |
| 4 Understand and specify the context of use | 4.1 Identify and document user's tasks<br>4.2 Identify and document significant user attributes<br>4.3 Identify and document organizational environment<br>4.4 Identify and document technical environment<br>4.5 Identify and document physical environment |
| 5 Produce design solutions | 5.1 Allocate functions<br>5.2 Produce composite task model<br>5.3 Explore system design<br>5.4 Use existing knowledge to develop design solutions<br>5.5 Specify system and use<br>5.6 Develop prototypes<br>5.7 Develop user training<br>5.8 Develop user support |
| 6 Evaluate designs against requirements | 6.1 Specify and validate context of evaluation<br>6.2 Evaluate early prototypes in order to define the requirements for the system<br>6.3 Evaluate prototypes in order to improve design<br>6.4 Evaluate the system to check that the stakeholder and organizational requirements have been met<br>6.5 Evaluate the system in order to check that the required practice has been followed<br>6.6 Evaluate the system in order to ensure that it continues to meet organizational and user needs |
| 7 Introduce and operate the system | 7.1 Management of change<br>7.2 Determine impact on organization and stakeholders<br>7.3 Customization and local design<br>7.4 Deliver user training Support users in planned activities<br>7.6 Ensure conformance to workplace ergonomic legislation |

Based on the requirements and contextual information gathered during previous steps, the project can then create prototypes and other design solutions taking into account the possible style guides and guidelines of the system. In the beginning of the project the prototypes can be really lightweight (like paper prototypes). As the project matures, the design solutions get more and more advanced.

**Table 2: Work products from UMM process number 3 specifying the stakeholder and organizational requirements.**

| Input | Output |
|---|---|
| • Project scope<br>• User representatives<br>• Work instructions<br>• Legislation<br>• Industry, national and international standards<br>• System strategy<br>• Context of use<br>• Competitor systems | • The range and relevance of users and other personnel in the design<br>• Risk assessment<br>• A statement of the human-centered design goals<br>• Stakeholder/User requirements specification<br>• Organizational requirements specification<br>• Priorities for different requirements<br>• Specific, measurable usability goals<br>• Benchmarks against which the design can be tested<br>• List of statutory or legislative requirements<br>• The sources from which the user and organizational requirements were derived |

These implementations (prototypes and other design solutions) are then evaluated against the criteria defined earlier to find out if they meet the requirements and to get feedback for the next round of design solutions. The usability issues found during the evaluation are added to the requirements and other criteria gathered before, which in turns leads to another iterative rounds of specification, implementation, and evaluation until the product meets the requirements.

Every step of this process requires a lot of reporting and documentation to distribute the results and reasoning to all stakeholders in the project.

A HCD process framework is provided by ISO/IEC 13407 standard and it is further refined in the Usability Maturity Model (UMM) as described in ISO TR 18529. The same way as in software engineering maturity models (like CMMI and SPICE), these standards describe seven processes. Each of those processes contains a set of base practices. The HCD processes and base practices are listed in Table 1.

The base practices in turn produce a set work products (deliverables). As an example Table 2 provides the work product list for process number 3 (Specify the stakeholder and organizational requirements). The complete list of work products is provided for example by Earthy (1999).

There are a multitude of practices and methods, which can be used to carry out the base practices defined in UMM. However UMM does not define these methods. Instead the method descriptions can be found in usability engineering literature, and although in some cases somewhat different set of methods are presented, there is in general a fairly good consensus of the most useful methods which is summarized at UsabilityNet.org Web site (Bevan, 2003).

The Usability Maturity Model defined in ISO 18529 is used as a general framework for this study. One has to remember that it is only one example of capability models developed for usability engineering (see for example Jokela, 2004a).

## 5. Open Source Development and Processes

The research on OS development processes has not traditionally presented any framework or guideline for how to design, manage, or evaluate OS software systems and processes (Gasser et al., 2003).

Although the amount of popular literature proves that the OS movement is growing, there are only a small number of empirical and systematic studies explaining how these communities produce software, how the participants coordinate efforts, and what software processes and work practices they need to succeed.

Mockus et al. (2002) examined the source code change history and the archive of problem reports from Apache and Mozilla development projects and created hypotheses on OS project organization. One of their hypotheses is directly related to processes in OS projects:

Hypothesis 2a: *If a project is so large that more than 10-15 people are required to complete 80% of the code in the desired time frame, then other mechanisms, rather than just informal, ad hoc arrangements, will be required in order to coordinate the work. These mechanisms may include one or more of the following: explicit development processes, individual or group code ownership, and required inspections.*

In short when the OS projects get bigger, multiple separate "project teams" are formed and there is no core development group which is allowed to commit code all over the project. Also policies start to emerge, for example regarding required inspections.

Scacchi (2004) notes that the OS projects do not adhere to traditional software engineering lice-cycle principles, but they rely on lean electronic communication media, virtual project management, and version management mechanisms to coordinate globally dispersed development projects. Also the project and processes co-evolve with their development communities. Scacchi states that OS projects employ at least five types of development processes:

- Requirements analysis and specification
- Coordinated version control, system build, and staged incremental release-review
- Maintenance of evolutionary redevelopment, reinvention, and revitalization
- Project management and career development
- Software technology transfer and licensing.

These processes do have counterparts in the traditional software engineering processes, although they are not readily adopted from traditional software engineering.

The OS projects do not use formal requirement elicitation, capture, analysis, validation. Instead the requirements take the for of threaded messages in discussions that are open for review, elaboration, refutation, or refinement. The developers do not explicitly generate requirements deliverables based on user representatives or focus groups, but the requirements are a side product of discussion of the implementation.

CVS (Concurrent Versions System) tools are integral to OS projects for coordinating the development and controlling the extensions and upgrades. At the same time online discussion are used to go through issues regarding various updates.

The maintenance of an OS project is not exactly what the term is understood to mean in the traditional software engineering. In OS projects the maintenance includes evolutionary redesign, reinvention and redevelopment, which leads to minor improvements or mutations with short life cycles. The reinventions are based on the learnings of the developers.

The project management in OS projects happens through Virtual Project Management utilizing the underlying meritocracy hierarchy of the developer pool. The Virtual Project Management exists to enable effective control via community decision-making.

The OS technology transfer is not an engineering process – at least not yet. Instead it depends more on the development of constructive social relationships, and willingness to search, browse, download and try out OS products. OS systems themselves, development assets, tools and project Web sites serve as a social venue and a place to build relationships and trust, as well as a place to share and learn with others.

In conclusion, informality is a main driver in OS development. Compared to quite strict and well-defined traditional software engineering processes or to the process maturity models presented briefly in section 4.1, work practices in OS are too laissez-faire that the maturity and process models would be useful or used in OS development communities. [7]

## 6. Applicability of UMM Processes in Open Source

In this section we go through the HCD processes described in UMM and contemplate the applicability of contained base practices (see Table 1) in OS projects. Substitute work practices used in OS projects are mentioned where available.

We conclude by providing a selection of UMM work products (deliverables) and comparable, less formal deliverables suitable for OS projects.

### *6.1 Ensure HCD Content in OS Projects*

Applying this part of the UMM to OS development world would be easier, if a corporation could participate in the OS development project, like Sun has done for example with OpenOffice.org.

Usually very little advance planning is done in the OS projects; the projects start often as the founder writes first lines of code of the system. Additionally, planning is not approached systematically, until the product gets too big to handle by the core developers.

Although most of the OS developers also use the product they develop, that does not mean that the stakeholders are

[7] The Open Source Maturity Model (OSMM) allows organizations successfully deploy and implement open source software (Golden, 2004). The model presents a three-phase process for selecting, assessing, and implementing open source products. It does not contain methodology to directly measure the maturity of the development organization the same way as for example CMMI and SPICE.

automatically adequately represented in the development process.

OS projects do not systematically collect market intelligence either. Online discussions might include occasional pointers to relevant competing solutions or some future development in the market, but as a systematic activity this is too resource-intensive for OS projects.

In OS projects defining and planning the system strategy is usually replaced with evolutional approach, as the formal planning does not fit well in to the OS philosophy. Features of the releases are not often planned ahead of time. Instead the feature set is more often frozen only as the time of some major release is near.

Market feedback collection is handled in OS projects by archiving the email feedback from users and setting up online message boards and discussion lists for the users. The other stakeholders (like system administrators, training personnel, etc.) have to use the same feedback channel as the end-users as there are no separate feedback mechanisms for different stakeholder groups.

Also analyzing trends in users is commonly omitted in OS projects – at least it is not done in a systematic way.

To sum it up, if the OS project does not have any usability professionals employed by an external organization to participate in the OS project, this fundamental phase of ensuring adequate and timely usability engineering gets too little attention in OS projects. Therefore participation of usability professionals from the very beginning of an OS project is important.

### *6.2 Plan and Manage HCD Process in OS Projects*

Users of the OS product who are not actively developing the product in question are not adequately consulted at the beginning of the development, but only at the later stages of the project, when the users can try the alpha and beta versions and submit feedback to the online forums of the OS development team. Of course user-developers, specifically those working on products targeted mostly to the developers, are better served in this regard.

The resources available severely limit the HCD methods and techniques, which can be selected for use in an OS project. Not only the time and work force is limited, but also the skill set of the developing team might not match with the skill required by each tool. The lack of skills can be solved by getting more usability experts to participate in the OS projects. OpenUsability.org provides a marketplace to promote this, as described in section 3.1.1.

Usually OS projects need to settle with those methods suitable for OS development, and which require only limited time and resources. Such methods include for example (UsabilityNet.org, 2003):

- Evaluating existing systems
- Creating scenarios
- Paper prototyping[8]
- Compiling style guides
- Rapid prototyping[9]

Evaluating the products can be challenging, as most of the usability evaluation methods require access to the users. Therefore the projects usually use only bug reports, comments on online discussions, and email feedback from the users.

Ensuring a HCD approach within the project team and planning the HCD activities is challenging, as traditionally most of the developers are engineers, whose skills do not match well with usability engineering.

As the OS projects usually manage the project through meritocracy and virtual project management (Scacchi, 2004), usability engineering aspects can be introduced in the project management if the usability people manage to gather enough authority through their actions. In other works, if they prove necessary to the project or some other project in the OS world, their opinions are listened with more care. If the usability people can achieve this in the OS projects also the evangelizing for (championing) the usability issues becomes much easier.

### *6.3 Specify the Stakeholder and Organizational Requirements in OS Projects*

As noted by Scacchi (2002) the software informalisms are found to play a critical role in the elicitation, analysis, specification, validation, and management of requirements for developing OS systems. These same informalisms should be used also to manage the requirements related to usability.

This kind of approach however makes it quite impossible to fully achieve many of the base practices included in this UMM process.

In most parts, the base practices 3.2, 3.4, and 3.5 (see Table 1) are carried out through threaded online discussions and threaded e-mail lists. In other words the requirements are not gathered systematically, but sporadically in those discussions resulting in usability informalisms.

For those OS projects, which plan to develop a replacement product for a proprietary software solution, the available public material about these proprietary applications could be used to describe for example the use of the planned OS system. As an example, an OS word processing application could utilize the tutorials and manuals of the proprietary solution to define the use of the system.

### *6.4 Understand and Specify the Context of Use in OS Projects*

Even the OS projects need to identify what kind of tasks the user is trying to do with the system. However documenting these tasks can be much more informal in OS projects than what is required in the Usability Maturity Model specification.

[8] Usually paper prototyping is too low-tech to appeal to engineers, and additionally the distributed nature of the development teams might cause this tool to be ineffective.

[9] This is natural extension of “release early, release often” –mantra, if only usability has been taken into account already in the beginning and not only after the interface has been designed.

The task documentation may take the form of threaded and annotated discussions in the online discussion boards, where the task is incrementally finalized.

Identifying the user attributes and various environmental variables (organizational, technical and physical) requires a little more formal approach. One could even argue that this step is not really required, because if the application is targeted to a broad audience the user characteristics should include everyone. And on the other hand, if the application is targeted to a highly specialized group of users, the developers – who most likely are "scratching their personal itch" (Raymond, 1999) – tend to know a lot of the user base, because they are the user base.

The specification of the technical environment is important to some degree. Meaning that the general form factor of the target device and network environment should be known before coding starts. However the operating system of the target device is less important. The project might be initially targeted to one specific operating system (some Linux/Unix variant is the natural choice for a OS developer), but if need arises to have the application running on other operating systems, some developer will raise to the occasion and start the porting project.

*6.5 Produce Design Solutions in OS Projects*

The mantra of many OS projects: "Release early and release often" (Raymond, 1999) fits well in the iterative nature of the usability engineering. Usually there will no problems in an OS project to explore the system design, to develop design solutions, and to create a steady stream of prototypes.

Usually the user training and also to some extent user support is beyond the scope of the OS project as the main focus is the get the application out. The lack of user training is solved in many cases, when a separate company starts providing the support for the OS product and creating the necessary materials (manuals, tutorials, support services, etc.) for it. This approach fits well with the OS approach, as long as the company does not make money with the application itself, but with the services built around it. All the companies providing various distributions of Linux (RedHat and MontaVista to name a few) are examples of this approach.

Small part of the user support is carried out inside the OS projects themselves through the various mailing lists and end-user services provided on the project Web site.

*6.6 Evaluate Designs Against Requirements in OS Projects*

This part of the process model requires the most legwork and usually it is left pretty incomplete or totally omitted in the OS projects. It also usually requires direct contact with the users, which might not fit well with engineering people.

Also, as the requirements in OS projects are usually quite informal, matching the prototypes and design solutions to these vague informalisms is either challenging or way too easy. Meaning that either there are no requirements to measure the prototypes against or then any design solution might fill the informal collections of requirements. If the project has produced a set of guidelines, this makes it easier to create products, which might fit in to the loosely defined requirements.

There has been recently more and more activity on usability evaluations among OS projects. In some cases (e.g. OpenOffice.org, GNOME) the evaluations are done with the support from interested corporations.

In some way part of the evaluation is done by the end-users, who submit bug reports to the project databases. As this is not very organized effort, some usability problems can be left unnoticed, unless there are enough users using the application ("Given enough eyeballs, all bugs are shallow", Raymond, 1999).

*6.7 Introduce and Operate OS Products*

Most of the base practices and work products in this HCD process do not usually belong to the responsibilities of an OS project itself. For example user training is not really applicable to OS development projects. However the companies living around the OS projects (providing user support, manuals, CD distributions, etc.) are better equipped to handle this part of the usability engineering process.

Also the traditional label of software maintenance does not quite fit what one sees occurring in different OS communities (Scacchi, 2004). He describes the management of OS products to be more like mutation, reinvention, or co-evolution of OS systems and their development communities.

*6.8 A Selection of UMM Work Products for OS Projects*

UMM specification lists almost 90 work products to be created during and by the usability engineering process. Most of the work products are not suitable for OS projects and some of them are quite impossible to create within OS development projects. A selection of most suitable work products and their counterparts in the OS projects are listed in Table 3. This table does not include those work products, which are better taken care by third party companies built around the OS projects. Examples of such work products are training related materials and manuals.

**Table 3: Selected usability engineering work products for open source projects.**

| Work products in UMM | Counterparts in OS projects |
|---|---|
| Stakeholder/User requirements specifications | Threaded discussions, e-mail archives |
| Organizational requirements specifications | Threaded discussions, e-mail archives |
| Priorities for different requirements | Informal ratings of usability informalisms on discussions/e-mail lists |
| Task information | Threaded discussions, e-mail archives |
| User interaction specification | Style guides, user interface guidelines |
| Dialogue detail | Style guides, user interface guidelines |
| Look and feel | Style guides, user interface guidelines |
| Usability and ergonomic defects | Bug reports/database |

*6.9 General Findings*

While going through the UMM base practices one by one

and thinking how they could be applied to OS projects, it became evident that the formal UMM model does not fit well in OS projects. However many base practices were already carried out in OS projects to some extent and using less formal methods.

A number of the UMM work products can be found in less formal format spread through the threaded discussions, which previous research has already listed being very valuable to OS projects (a place to gather informalisms).

## 7. Discussion

This study has approached the usability in OS products from a formal angle, using a rigid process perspective to view the development projects. This kind of approach was doomed already from the beginning, taken into account the very informal nature of open source projects. However, as the usability of OS applications seems to be getting better and better without using the formal usability engineering processes, one could argue that in the long run the OS communities will find the most effective OS usability practice – not from the UMM world, but creating new practices. At least that is what happened with the OS development model as a whole.

This has been also elaborated in an essay on advocato.org (Salmoni, 2004). Salmoni for example suggests that the discussion stating that OS usability will fail *"reminds me of similar criticisms of the open source method when it first started becoming famous"*. Salmoni also elaborates that user testing might be over-rated and that other usability testing methods (like empirical assessments) can be very well use to replace user testing at least in the early phases of OS development process.

### *7.1 Future Research*

One solution to enhance the usability of OS products is that usability experts working for corporations work in OS projects. This is very challenging task for the usability experts as described by Benson et al. (2004b). The decentralized and engineering-driven approach of OS source projects can be at odds with usability engineering methodologies. In one hand the experts need to meet the corporate needs, like consistency, documentation, support, and conformance to development processes. On the other hand they need to interact with engineers worldwide and evangelize usability to an unseasoned community.

However as the experiences with GNOME, NetBeans, and OpenOffice.org show, professional usability experts can help the OS projects as their more structured approach allows them to produce specific user interface specification documents, interface guidelines, and conduct laborious and time-consuming usability evaluations, which usually do not exist in OS projects, but as a single source of information can be really helpful to the projects.

Therefore an area of further study could be to validate if the participation of usability professionals and supporting organization helps to enhance the usability of OS applications. One way could be to compare two OS projects working on similar applications, but having different levels of corporate support. The author suggests someone to compare the usability approach, participation levels of usability professionals, and usability engineering processes in general between KOffice[10] and OpenOffice.org[11] projects. Both projects develop OS office application suite, but OpenOffice.org has a strong backing from Sun, whereas KOffice does not have similar amount of corporate backing. However as the usability related material within these projects is spread among many sources – mailing lists, bug databases, and web sites – the study is not simple to carry out. The sources for KOffice usability information include at least koffice, koffice-devel, kde-usability, kde-quality, and kde-developer mailing lists, the bug database, and KOffice and KDE Web sites. For OpenOffice.org one has to go through at least dev@openoffice.org, allbugs@openoffice.org, dev@qa.openoffice.org, issues@ui.openoffice.org, and dev@ui.openoffice.org mailing lists, the UI issue tracker database, as well as OpenOffice.org and UI.OpenOffice.org Web sites.

## Acknowledgements

The author would like to thank professor Marko Nieminen (Helsinki University of Technology) for support and for helping to organize and focus the paper.

## References

Benson, C. Elman, A. Nickell, S. Robertson, C., Z. (2004a). *GNOME Human Interface Guidelines 2.0.* The GNOME Usability Project. Available at http://developer.gnome.org/projects/gup/hig/2.0/ Accessed November 29th, 2004.

Benson, C. Müller-Prove, M. Mzourek, J. (2004b). Professional Usability in Open Source Projects: GNOME, OpenOffice.org, NetBeans. *Proceedings of CHI 2004.*

Bevan, N. (2001). International Standards for HCI and Usability. *International Journal of Human-Computer Studies*, 55, 4.

Bevan, N. (2003). UsabilityNet Methods for User Centered Design. *Human-Computer Interaction: Theory and Practice (Part 1), Volume 1 of the Proceedings of HCI International*. Lawrence Erlbaum. pp. 434-438.

Earthy, J. (1999). *Usability Maturity Model: Processes, version 2.2.*

Gasser, L. Scacchi, W. Ripoche, G. Penne, B. (2003). *Understanding Continuous Design in F/OSS Projects.* 16th. International Conference on Software & Systems Engineering and their Applications.

Golden, B. (2004). *Succeeding with Open Source*. Addison-Wesley Professional. pp. 61-87.

ISO/IEC 15504 (1998). *Software Process Assessment, Parts 1-9.* Available at http://www.sei.cmu.edu/iso-15504/resources/part<x>100.pdf (replace <x> with Part number). Accessed November 28th, 2004.

[10] KOffice (http://www.koffice.org/) is a free, integrated office suite for KDE, the K Desktop Environment (a graphical desktop environment for Linux and Unix).

[11] OpenOffice.org (http://www.openoffice.org/) is an open, multi-platform (Linux, Mac OS, and Windows), and integrated office suite, which is used as a basis for Sun's commercial StarOffice office suite product (http://wwws.sun.com/software/star/openoffice/index.html).

Jokela, T. Lalli, T. (2003). Usability and CMMI: Does A Higher Maturity Level in Product Development Mean Better Usability? *Proceedings of CHI 2003*.

Jokela, T. (2004a). *Different Usability Maturity Models*. Available at http://www.tol.oulu.fi/~tjokela/umm%20list.htm. Accessed October 26th, 2004.

Jokela, T. (2004b). Evaluating the user-centredness of development organisations: conclusions and implications from empirical usability capability maturity assessments. *Interacting with Computers* (Article in Press, Corrected Proof). Available at http://www.sciencedirect.com. Accessed Nov 28th, 2004.

Mockus, A. Fielding, R. Herbsleb, J. (2002). Two Case Studies of Open Source Software Development. Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, Vol 11, no. 3, July 2002*. pp. 309-346.

Mozilla.org (2003). *Mozilla UI Sector*. Available at http://www.mozilla.org/projects/ui/ Accessed November 29th, 2004.

Nichols, D. M., Twidale, M. B. (2002). The Usability of Open Source Software. *FirstMonday.org, issue 8_1*. Available at http://www.firstmonday.org/issues/issue8_1/nichols/index.html. Accessed October 13th, 2004.

Nielsen, J. (1993). *Usability Engineering*. Boston, Academic Press. pp.23-37.

OpenOffice.org (2003). *Specification Proposals for OOo 1.1.x* Available at http://ui.openoffice.org/proposals/index.html Accessed November 29th, 2004.

Raymond, E., S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates, Sebastopol, CA. Also available at http://www.tuxedo.org/~esr/writings/cathedral-bazaar/. Accessed November 29th, 2004.

Royce, W.W. (1970). Managing the Development of Large Software Systems. *Proceedings of 9th International Conference on Software Engineering, IEEE Computer Society, 1987*. pp. 328-338. Originally published in Proc. WESCON, 1970.

Salmoni, A. J. (2004). *Open source usability*. Available at http://www.advogato.org/article/780.html. Accessed November 28th, 2004.

Scacchi, W. (2001). Process Models in Software Engineering, version October 2001. Available at http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf. Accessed November 15th, 2004. Final version to appear in, J.J. Marciniak (ed.), *Encyclopedia of Software Engineering, 2nd Edition* John Wiley and Sons, Inc, New York, December 2001.

Scacchi, W. (2002). Understanding the Requirements for Developing Open Source Software Systems. *IEEE Proceedings Software, volume 148, number 1*. pp. 24-39.

Scacchi, W. (2004). Free/Open Source Software Development Practices in the Computer Game Community. *IEEE Software, 21(1). January-February 2004*. pp. 56-66.

Smith, S. Engen, D. Mankoski, A. Frishberg, N. Pedersen, N. Benson, C. (2001). *GNOME Usability Study Report.* Sun GNOME Human Computer Interaction (HCI), Sun Microsystems, Inc. Available at http://developer.gnome.org/projects/gup/ut1_report/report_main.html Accessed October 12th , 2004

SourceForge.net. (2004). People page available at http://SourceForge.net/people. Accessed November 28th, 2004.

Thomas, M. (2002). *Why Free Software usability tends to suck.* Available at http://mpt.phrasewise.com/discuss/msgReader$173 Accessed October 12th, 2004.

UsabilityNet.org (2003). Methods Table. Available at http://www.usabilitynet.org/tools/methods.htm. Accessed November 18th, 2004.