Towards Open Source Software Adoption

Educational, Public, Legal, and Usability Practices

OSS 2006 tOSSad workshop proceedings

Como, Italy, June 10, 2006

Bülent Özel
Can Burak Çilingir
Kaan Erkan (Eds.)

First published 2006 by
TÜBİTAK (The Scientific & Technological Research Council of Turkey)

Cover design
Akinon Design Studio
www.akinon.com

Printed and bound by
TÜBİTAK MAM Matbaası, 41470, Kocaeli, Turkey

Towards Open Source Software Adoption
Educational, Public, Legal, and Usability Practices
OSS 2006 tOSSad workshop proceedings
Como, Italy, June 10, 2006
by Bülent Özel, Can Burak Çilingir, Kaan Erkan (Eds.)

ISBN: 975-403-373-0

Preface

This book contains papers presented at tOSSad 2006 workshop. The workshop was hosted by The Second International Conference on Open Source Systems (OSS 2006) June 10th 2006, Grand Hotel, Como, Italy.

The main objective of tOSSad workshop at OSS 2006 meeting has been to integrate, and to discuss already formed methodologies, strategies, skills and technologies in F/OSS domain in order to help public bodies, educational institutions and SMEs to share research results, establish synergies, build partnerships and innovate in an enlarged Europe.

tOSSad[1] (towards Open Source Software adoption and dissemination) is an EU FP6 project funded under IST-3. It aims at improving the outcomes of the F/OSS communities throughout Europe through supporting the coordination and networking of these communities by means of state-of-the-art studies, national program initiations, usability cases, curriculum development and implementation of collaborative information portal and web based groupware. By conducting these actions on an international European level, with inclusion of the ACC and NMS countries the tOSSad project acts for a general acceptance and coordinated boost of F/OSS development. The project consortium consists of 19 partner organisations from 15 different EU countries.

The workshop proceedings consist of technical reports, research papers, case analysis, and demonstrations. The papers address following issues:

1. F/OSS Training/Educating Needs and Methods
2. F/OSS Training Experiences
3. Legal Issues in F/OSS Adoption
4. F/OSS Adoption Cases and Experiences
5. Usability in F/OSS

[1] www.tOSSad.org

Bülent Özel, `bulent@cs.bilgi.edu.tr`
Can Burak Çilingir, `canburak@cs.bilgi.edu.tr`
Kaan Erkan, `kaan.erkan@uekae.tubitak.gov.tr`
Görkem Çetin, `gorkem@uekae.tubitak.gov.tr`

June, 2006

Contents

Adoption of Open Source Software on the Enterprise Desktop

Michael Amberg and Steffen Möller

Friedrich-Alexander-Universität Erlangen-Nürnberg, Department of Business Administration, Economics, and Social Sciences, Business Information Technology, Lange Gasse 20, 90403 Nürnberg, Germany
{amberg,steffen.moeller}@wiso.uni-erlangen.de
WWW home page: http://www.wi3.uni-erlangen.de

Summary. Open source software (OSS) has recently become more and more interesting for companies, organizations and public administration due to opportunities like the independence from single software vendors and its availability mostly free of charge. However, the realization of those opportunities strongly depends on the successful implementation of the OSS within the organization and thus, the adoption and the (continued) use of the system by the employees. Researchers as well as practitioners agree on employees' acceptance being the key determinant on information systems' use and thus implementation success. However, only very little is actually known about the acceptance of OSS on the desktop. In order to fill this gap, we perform a case study examining a real-world OSS implementation project.

1 Introduction

Open source software (OSS) has recently become more and more interesting for companies, organizations and public administration due to opportunities like the independence from single software vendors and its availability mostly free of charge. However, the realization of the benefits of OSS strongly depends on how effectively the software is implemented within an organization. Researchers as well as practitioners agree on employees' acceptance being the key determinant on information systems' use and thus implementation success (e. g. see [6] or [15]). Neglecting employees' acceptance can cause refusal among employees, which in turn can lead to resistance which finally results in project cost overrun and even project failures (cf. [11]). However, only very little is actually known about the acceptance of OSS on the desktop. In order to fill this gap, we perform a case study research [17] examining a real-world OSS implementation project affecting about 400 employees spanning different destinations, functional areas and hierarchy levels.

We proceed as follows: we first specify a suitable acceptance model for our research (section 2). Subsequently, we detail on the application of the model in our dedicated OSS implementation project and analyze the results (section 3). Finally, we summarize the key findings in section 4.

2 Specification of the acceptance model

2.1 Review of extant acceptance models

Generally, two main classes of acceptance models can be identified: basic models and integrated models. The basic models denote approaches which are mainly founded in intentional models originated in social sciences research. Common basic models are for example the Theory of Reasoned Action (TRA; [8]), Technology Acceptance Model (TAM; [6]), Theory of Planned Behavior (TBP; [1]), Task-Technology-Fit model (TTF; [9]), and Diffusion of Innovation (DoI, [13]) theory.

The second class, integrated models, denotes models which are built mostly on combinations of basic models, e. g. the acceptance model of Taylor and Todd, which is build on the integration of TAM and TBP [14], De Carvalho et al., (TAM and TTF; [7]), the Unified Theory of Acceptance and Use of Technology (UTAUT) by Venkatesh et al. [15], which is an integration of eight acceptance models, among which are TAM, TRA, TBP, and TTF, and finally the Dynamic Acceptance model for the Reevaluation of Technologies (DART) by Amberg et al., an integration of concepts of five existing acceptance models including TAM and TTF [2].

Regarding the explanatory power of existing acceptance models, recent analysis show the superiority of integrated models over basic models (cf. [15]). Thus it seemed to be a good starting point for our analysis to rely on the experiences with existing integrated models. Although most of the depicted models claim to be employable for the evaluation of information systems in general and therefore also seemed to be suitable for the evaluation of OSS on the desktop, it is a common approach to adjust those universal models to the requirements of a particular research item (cf. [12]). Consequently, a flexible approach providing adaptability to individual requirements of the research item, while ensuring a balanced consideration of those factors was needed.

According to design criteria of the DART-approach, being based on a meta-structure in order to identify a balanced set of individually measurable acceptance determinants, DART is expected to be particularly useful to support this requirement. As a result, the DART approach was selected as a basic framework for our analysis.

2.2 Model Specification

DART is a highly flexible model, designed for the analysis and evaluation of employee and user acceptance and has already been applied in a variety of different research areas, e.g. situation-dependent mobile services, web-based aptitude tests, and process innovations [16, 2, 3, 4]. DART is based on the fundamental idea of the balanced scorecard (BSC) using a meta-structure in order to identify a balanced set of individually measurable acceptance criteria.

Consequently, no complete set of acceptance determinants is given in advance, they rather have to be specified according to the concrete research item. According to the BSC approach, the intention is to find a set of precise determinants with high significance, meeting the requirements of sustainability, measurability, achievability, reasonability and timeliness [12].

3 Application of the Acceptance Model in an OSS implementation context

3.1 Project Setting

Because of the character of the acceptance evaluation, to provide insights within a complex (organizational) environment, an appropriate research design was necessary. As shown in [5], case study research is the predominant research approach considering information systems adoption within an organizational context. Consequently, case study research was adopted to guide this research.

The case presented in this paper focuses on the implementation of OSS on the desktop in a organization of the IT industry. In particular, the research item is a company-specific OSS desktop environment based on a standardized Linux distribution and including several open source desktop applications (office, web browser) as well as some business critical closed source applications (ERP, CRM).

Prior to the analysis itself it was necessary to identify a set of suitable acceptance determinants for the evaluation of OSS on the desktop. By performing an in-depth literature review according to the DART-approach, a total of 13 key acceptance determinants were identified and taken as a foundation for the subsequent analysis:

- **Perceived Usefulness:** Motivation, Task Adequacy, Relative Advantage
- **Perceived Ease of Use:** Complexity, Usability, User Compatibility, Technical Compatibility
- **Perceived Network Effects:** Training, Communication, Information
- **Perceived Costs:** Individual Migration Effort, System Quality, Implementation Effort

In the subsequent step to the determinant identification, the determinants were used to develop a standardized questionnaire utilized for the acceptance survey. On basis of each determinant a number of suitable questions were derived taking existing scales into consideration.

3.2 Results

Subsequent to the preparation, the questionnaire was transferred into a web-based survey system and an invitation to complete the questionnaire was emailed to all employees concerned. About 30% of all the employees completed our survey.

Following [15] and [10], we performed three key analysis on our data: a correlation analysis to identify highly correlating acceptance indicators, a factor analysis to reveal underlying dimensions of the indicators and finally a regression analysis to evaluate the direct influence of the dimensions on the actual system usage.

Analyzing the correlation of our indicators with the actual OSS usage, we found that eleven of our thirteen determinants correlated significantly with the actual usage of the OSS desktop. Only two determinants, Communication and Training, were found to be correlating less significantly. Regarding the correlation of the indicators among each other, the results suggest that several acceptance determinants were seen as different aspects of the same dimension and thus can be grouped together.

Consequently, we performed an explanatory factor analysis, and found eight underlying dimensions: System Performance (grouping Task Adequacy, Motivation, and System Quality), Employee Integration (Communication and Information), Relative Advantage (Relative Advantage), Training (Training), Compatibility (Technical Compatibility, User Compatibility), Individual Adequacy (Complexity, Individual Migration Effort), Implementation Effort (Implementation Effort), and Usability (Usability).

To further analyze our findings, we performed a multiple regression analysis to identify the (direct) influence of each dimension on the actual usage of the OSS desktop. This yielded to four dominant acceptance dimensions which show a significant influence on the usage of OSS on the desktop: System performance (.41***), Individual Adequacy (.35***), Relative Advantage (.33***), and Compatibility (.11). These dimensions as well as the eight underlying indicators can be considered as the most important factors which should be carefully reviewed to ensure OSS acceptance and usage.

Taking the explanatory power of the resulting model into consideration, we found that the model was able to explain a total variance of .40 on the actual usage of the OSS desktop (adjusted R). This could be considered as slightly superior to the majority of existing acceptance models which report an average variance of about .35 (for a recent review see [15]). Finally, the validity of the model was confirmed by performing additional model tests (F-test, multicollinearity, heteroscedacity, autocorrelation, residual analysis).

4 Conclusion

The purpose of this research was to analyze the adoption of OSS on the enterprise desktop in order to identify the key influencing factors on the employee's adoption decision. By performing a case study research examining a real-world OSS implementation project, we identify four key aspects: System performance, Individual Adequacy, Relative Advantage, and Compatibility.

We expect that an OSS implementation strategy, which aims at the improvement of these four aspects could significantly increase the actual use of the OSS and therefore contribute to a successful implementation of OSS within an organization.

However, even if the results suggest the general applicability of our approach, researchers and practitioners should be aware of several limitations mainly originated in our research approach. Further analysis will be required to test and extend the boundaries of the model presented. For instance, the general applicability, the validity and reliability of our results need to be further analyzed in one or more longitudinal field studies including a higher number of projects, different migration strategies, varying implementation levels and even with a different impact on the employees daily business. Our future work yields into that direction.

References

1. Ajzen, I. (1991): The theory of planned behavior, Organizational Behavior & Human Decision Processes, 50, 2, 179-211.
2. Amberg, M., Hirschmeier, M. and Schobert, D. (2003): DART - Ein Ansatz zur Analyse und Evaluierung der Benutzerakzeptanz, in Uhr, W. et al. (Eds) Proceedings of the 6^{th} Internationale Konferenz für Wirtschaftsinformatik, Heidelberg, Physica, 573-592.
3. Amberg, M., Möller, S. and Remus, U. (2005): Employees' Acceptance of Process Innovations: An Action Research Approach, in Zhao, X., Liu, B. (Eds.) Proceedings of the 5^{th} International Conference on Electronic Business, Hong Kong, 401-403.
4. Amberg, M., Fischer, S. and Schröder, M. (2005): Analysis of User Acceptance for Web-based Aptitude Tests with DART, in Proceedings of the 11^{th} Americas Conference on Information Systems, Omaha, 2005, 2428-2433.
5. Choudrie, J. and Dwivedi, Y. K. (2005): Investigating the Research Approaches for Examining Technology Adoption Issues, Journal of Research Practice, 1, 1, 1-12.
6. Davis, F. D. (1989): Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology, MIS Quarterly, 13, 3, 319-339.
7. De Carvalho, R.B., Ferreira, M.A.T. and Choo, C.W. (2005): Towards a Portal Maturity Model (PMM): Investigating Social and Technological Effects of Portals on Knowledge Management Initiatives, in Vaughan, L. et al. (Eds.) Proceedings of the 2005 annual CAIS/ACSI Conference, London, Ontario.
8. Fishbein, M. and Ajzen, I. (1975): Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research, Massachuttes, Addison-Wesley.
9. Goodhue, D. L. and Thompson, R. L. (1995): Task-Technology Fit and Individual Performance, MIS Quarterly, 19, 2, 213-236.

10. Heijden, H. v. d. (2004): User Acceptance of Hedonic Information Systems, MIS Quarterly, 28, 4, 695-704.
11. Lorenzi, N. M., Riley, R. T. (1994): Organizational Aspects of Health Informatics: Managing Technological Change, New York, Springer.
12. Möller, S. and Remus, U. (2006): User Acceptance of Enterprise Portals, in Tatnall, A. (Ed.) Encyclopaedia of Portal Technology and Applications, Hershey, Idea Group.
13. Rogers, E. (1995) Diffusion of Innovations, 4^{th} Ed., New York, The Free Press.
14. Taylor, S. and Todd, P. (1996): Understanding information technology usage: A test of competing models, Information Systems Research, 6, 2, 144-176.
15. Venkatesh, V., Morris, M. G., Davis, G. B. and Davis, F. D. (2003): User Acceptance of Information Technology: Toward a Unified View, MIS Quarterly, 27, 3, 425-478.
16. Wehrmann, J. (2004): Situationsabhängige mobile Dienste, Berlin, WiKu.
17. Yin, R. K (1994): Case Study Research - Design and Methods. 2^{nd} Ed., Sage Publications, Beverly Hills, CA.

GPL v3: A Legal Analysis

Andrés Guadamuz González

AHRC Research Centre for Studies in IP and IT Law
University of Edinburgh, Old College, South Bridge
Edinburgh EH8 9YL, UK
a.guadamuz@ed.ac.uk

Summary. Scheduling this paper offers a first-look legal analysis of the draft version 3 of the GNU General Public License, and will also look at the debate that it has generated in the Free and Open Source community. The paper will answer the following vital questions: Is the GPL v3 different in any fundamental way from GPL v2? What are the main differences? What will be the impact of the new GPL to the problem of software patents? Is there an incompatibility problem with previous versions of the licence? How does it compare with other existing licences?

1 Introduction

The draft of the version 3 of the GNU General Public License (GPL) was released in January 2006, causing quite a stir in the Free Software and Open Source (hereby referred to as FOSS) communities. The draft is still under discussion at the time of writing, but it is becoming clear that the new GPL could prove to be a monumental split the non-proprietary community by making developers choose between different types of development ideologies. The purpose of the present paper will be to provide a legal analysis of the clauses contained in the new draft in order to answer whether the existing GPL requires an update.

2 GPL basics

It is not an exaggeration to state that the GNU General Public License (GPL) is the most important licence in the FOSS movement. At the time of writing, 67% of all projects listed in the Sourceforge open source repository are released under the GPL[1]. But why is it so important? What makes the GPL the licence of choice?

[1] Out of 73,978 projects, 50,013 were released with the GPL. See: http://sourceforge.net/softwaremap

The GPL was first drafted in 1985[2] by Richard Stallman of the Free Software Foundation (FSF) in order to accommodate the ideas of free distribution of source code espoused by their development philosophy[1]. The licence was designed to protect software released by the FSF with something that Stallman called copyleft[3]. This concept was to allow the further distribution and modification of software released under its terms, but it would not allow proprietary developers to come and "close" the released software[3]. The version 2 (and current) version of the licence was released in 1991, and it was drafted by Stallman and by law professor Eben Moglen[3].

The GPL can be described as a form contract[4] that ensures the source code is passed on, but anyone who redistributes it, with or without changes, must pass along the freedom to further copy and change it. This places a burden on the person transferring the software; the burden is that the software must remain "free", as defined by the FSF and the GPL[5]. This is different from just placing software in the public domain because the person making use of the free code can subsequently copyright it and release it in proprietary format[6].

The GPL is the legal framework that sustains the copyleft system[6]. It reads as a mixture of a legal contract and an ideological manifesto. The preamble to the work restates clearly some of the most common beliefs of free software and the non-proprietary approach, with several admonitions about the meaning of the word "free". The main point is that, as mentioned before, the source code must be made available to the users. The preamble states:

> "For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights."[7]

The licence specifies that this is achieved by two means: by protecting the software by means of copyright; and by providing the users with a licence that gives them the freedom to use and modify the software in any way they see fit if they meet the stated conditions. The main body of the licence reiterates these ideas. Section 1 for example states:

> "1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate

[2] Although the official dating of version 1 is 1989. This is because the GPL was known originally as the EMACS General Public License.

[3] For more about the concepts of copyleft, see: Guadamuz A. [2]

[4] Some people do not believe that the GPL is a contract. See: Jones P.[4]. For an excellent rebuttal of this argument, see: Rosen L.E. [5]

[5] Or as they like to repeat, free as in freedom, not as in beer.

[6] The full text of version 2 of he licence can be found at: `http://www.fsf.org/licenses/gpl.html`

[7] Free Software Foundation: GNU General Public License v2, Preamble.

> copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program."[8]

This section also mentions that the user can make monetary charges when distributing the copy as long as the charges are for expenses in making copies of the software, which is also consistent with the general free software characteristic that does not discriminate against commercial software as long as it is not proprietary[9] commercial software.

Many of the provisions of the GPL can be found in other non-proprietary software licences. What makes the GPL unique is section 2(b) of the licence, as this is where the restrictions against using the software to create commercial software are specified. The section reads:

> "2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions: ... b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License." [7]

What this means is that any software developed by using the open source code of the copyleft program must not charge for the derivative product, and most importantly, must ensure that the GPL is transferred to further users of the derivative software. This type of licence has been aptly named a "viral contract" by Professor Radin, defining them as "contracts whose obligations purport to 'run' to successor of immediate parties" [8][10]. These contracts would then spread in a viral form, as the licensee must include the terms of the GPL in any subsequent licence they will include to their derivative work because that obligation is part of the contract, and then those subsequent licensees will have to impose the same contractual terms in further licences that they perform, ad perpetuam.

Despite the fact that copyleft licences tend to promote the free software principles and the definitions drafted by the FSF and Stallman, it must be pointed out that some of the contractual restrictions existing in licences such as the GPL have prompted some criticism from enterprises and commercial users[11]. Despite these criticisms and an incessant campaign from some proprietary software developers against it, the GPL has stood well against most

[8] Free Software Foundation. GNU General Public License v2, Preamble.

[9] Proprietary is generally equated to "closed" source[7].

[10] Some people in the Free Software community do not like the term "viral", but it has become more prevalent in the literature. See [9].

[11] For some criticisms, see: Guadamuz A.[10]

challenges, as exemplified by its longevity and the relatively small litigation that it has generated[11].

3 GPL v3

While the longevity and success of the GPL have been a good testament as to its legal validity[12], it was felt at the FSF that it was time for an updated version to fine-tune some legal issues and to respond to developments in the Free Software community. Prior to the release of the draft, Stallman and Moglen identified key issues that they believed needed to be addressed by the licence[13]. These were:

1. Internationalise the licence: While GPL v2 was drafted by with American law in mind, it is purported to adhere to the international copyright principles present in the Berne Copyright Convention[12] While the only ruling in favour of the GPL has taken place outside of the United States,[14] it has been felt that it requires tightening of its international compliance.
2. Standard setting: Stallman and Moglen argue that the GPL has become an international standard for the FOSS industry, so any update has to take this into consideration.
3. Responding to changing circumstances: The FOSS community, and in particular the principles of Free Software, are faced with new threats, such as patentability of software, trusted computing and digital rights management. The GPL must change to answer those threats[13].

So, what is contained in the new licence? The first draft of the version 3 of the GPL is longer, and unfortunately less clear than version 2[14].This is a real problem, as it has been often pointed out that the existing wording of the GPL is not the clearest example of copyright licensing, its interpretation and broad generalisations are the subject of endless legal interpretation[15].

The aforementioned document stating the need for a redraft prompted expectations that the new draft text would attempt to overhaul the protection of GPL software against software patents and it would perhaps update the copyleft principle present in the existing section 2(b). Although the actual draft mentions these topics, the real surprise came with regards to digital rights management (DRM), in particular technical protection measures (TPM). The new GPL could have a lot of consequences for the future of open source usage in the entertainment industry. One of the most contentious sections is section 3 on DRM:

> "Regardless of any other provision of this License, no permission is given to distribute covered works that illegally invade users' privacy,

[12] Berne Convention for the Protection of Literary and Artistic Works 1886.
[13] Stallman and Moglen, supra note 19.
[14] The draft can be found here: `http://gplv3.fsf.org/draft`.

> nor for modes of distribution that deny users that run covered works the full exercise of the legal rights granted by this License."[15]

In other words, the distribution of derivatives with works that contain certain restrictive types of DRM will be prohibited. While it must be commended that this clause is very specific against specific types of technical protection measures, the provisions are still hamfisted when read in conjunction with the next paragraph.

> "No covered work constitutes part of an effective technological protection measure: that is to say, distribution of a covered work as part of a system to generate or access certain data constitutes general permission at least for development, distribution and use, under this License, of other software capable of accessing the same data."[16]

This paragraph specifically excludes all works distributed under the GPL from the anti-circumvention measures in the WIPO Copyright Treatiy[17] (WCT) by specifically stating that the licensed software shall not constitute "an effective technological protection measure", and it would therefore not apply for such protections. This seems a much more effective way of dealing with TPMs than the broader and more difficult to implement version of the first paragraph.

Surprisingly, the new version is not as adamant against software patents as it was expected as a reflection that some of the biggest open source and free software players in the United States are acquiring patents as well[18] Stallman and Moglen seem very pragmatic regarding software patents, and recognise that FOSS developers may be involved in complex patent licensing transactions, therefore their implicit recognition of the status quo[19].

The new draft expands on the implicit patent licensing of the existing GPL and make it an explicit patent grant in section 11 of the draft. The licence prohibits distribution of the software by people who have initiated or brought patent suits arising from the software. This is a clever attack on software patents, it does not forbid people to apply for them, but it frowns upon enforcement. Strangely, the licence allows developers to add a stronger patent retaliation clause that is not incompatible with the licence. This clause is spelled out, but it is not really a part of the licence, it is just not incompatible with it. The compatible patent clause states:

[15] s3 of GPL v3.
[16] Ibid.
[17] Specifically, Art. 1 of the WCT, which states that "Contracting Parties shall provide adequate legal protection and effective legal remedies against the circumvention of effective technological measures..."
[18] Apache, IBM and Novell have applied for thousands of software patents in the United States. For more about this, see: Guadamuz A.[16].
[19] See their draft rationale here: `http://gplv3.fsf.org/rationale`.

e) They may impose software patent retaliation, which means permission for use of your added parts terminates or may be terminated, wholly or partially, under stated conditions, for users closely related to any party that has filed a software patent lawsuit (i.e., a lawsuit alleging that some software infringes a patent). The conditions must limit retaliation to a subset of these two cases:
   1. Lawsuits that lack the justification of retaliating against other software patent lawsuits that lack such justification.
   2. Lawsuits that target part of this work, or other code that was elsewhere released together with the parts you added, the whole being under the terms used here for those parts"[20]

This may prove to be terribly confusing for the average reader in my opinion, and a bad drafting practice.

Another interesting feature of the new draft is that it revamps the old copyleft clause 2(b). As explained earlier, under the current GPL the copyleft aspects only apply to derivative works that are distributed to the public. In other words, you take a work under the GPL, change it and then distribute your own adaptation. Then you have to re-distribute it under the GPL. This simple rule would generally provide clear-cut cases in which the GPL would apply and where it would not. For example, imagine that a hardware developer creates a driver module that interacts with the Linux kernel (distributed under the GPL). It is not part of the kernel, but it interacts with it. Under GPL v2, it is clear that this module is not a derivative, and therefore it does not need to be distributed under the GPL.

What happens with GPL version 3? The situation with derivatives is less straightforward. The copyleft clause in the new GPL has been given a tremendous boost. Users and developers still have the right to install and use GPL software without restrictions. Developers also have the right to privately modify the software, unless they have initiated a patent suit "against anyone for making, using or distributing their own works based on the Program." The problem is that of modifications that are distributed. Consider section 5(b) (the old section 2.b):

> "b) You must license the entire modified work, as a whole, under this License to anyone who comes into possession of a copy. This License must apply, unmodified except as permitted by section 7 below, to the whole of the work. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it."[21](Emphasis mine).

This would certainly apply to anyone who includes any sort of modification of code into their work. Imagine for example that you include modified

[20] S7(e) draft GPL v3.
[21] S5(b) draft GPL v3.

modules from the Linux kernel in your work in order to provide compatibility with the kernel. It seems like you would definitely have to licence the entire program under the GPL! This would not apply if the work can be clearly identified as not being a derivative. The new GPL says:

> "These requirements apply to the modified work as a whole. If identifiable sections of that work, added by you, are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works for use not in combination with the Program. But when you distribute the same sections for use in combination with covered works, no matter in what form such combination occurs, the whole of the combination must be licensed under this License, whose permissions for other licensees extend to the entire whole, and thus to every part of the whole."[22]

This seems like overboard protection. Even if a work is not based on GPL software but it is distributed in combination with covered works, then it must be licensed under the GPL. What will happen with open source projects that do not choose to use the GPL, such as Mozilla and Apache? And what about interfaces? At first glance, it seems that this would also affect drivers and modules that interact with the kernel. However, the draft further confuses this by stating:

> "A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. Mere inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate."[23]

This paragraph seems to generate more legal uncertainty, something that has not been lost on commentators of the draft[24]. This paragraph tries to rationalise specific cases in which the revamped GPL copyleft clause will apply by inventing a new definition to what a compilation is. They try to determine that the software distributed in the same distribution medium has to be GPL if it is a "compilation", but not if it is an "aggregate". Why create the new terminology?

It seems like the drafters of the new GPL realised that the new copyleft clause would be controversial when they included an excuse within it:

[22] S5 draft GPL v3.
[23] Ibid.
[24] For a series of comments, see: `http://gplv3.fsf.org/comments/`.

> "Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program."[25]

Why excuse the clause in the licence if there is nothing to excuse?

4 The verdict so far

It is difficult to ascertain yet what will be the final result of the draft GPL v3. Stallman, Moglen and the FSF must be commended by the inclusive and open process with which they have undertaken the drafting process. The GPL v3 has been open for discussion to the public since January 2006, a process that will continue with specific working groups (both open and closed to public participation), and a further draft can be expected soon. By reading some of the comments, it seems that the FOSS community has been finding specific areas of special concern, particularly the revamped copyleft clause and the DRM provisions. Take for example a comment to section 4 with regards to DRM and software that violates privacy:

> "Frankly, this clause seems off-topic. Why not, for instance, deny distribution to covered works that are components of weapons of mass destruction? The whole issue of what constitutes privacy is something of an open issue these days, when everybody submits their corporate shopping ID at the grocery store checkout counter and thinks nothing of having their most everyday habits and activities tracked, profiled, and distributed to anyone willing to pay for them."[26]

The DRM issue has become a real sticking point with the wider developer community. The most controversial indictment of the draft, and perhaps the most damaging, was delivered by Linus Torvalds, the father of the Linux kernel. Torvalds has declared that the Linux kernel will not be released using GPL v3, but that it will continue to use version 2. His problem lies specifically with the strong prohibition on the use of DRM found in section 3. Torvalds commented that the new GPL would require people to give control of their private keys, which he will not do. He stated that:

> "Sure, DRM may mean that you can not _install_ or _run_ your changes on somebody else's hardware. But it in no way changes the fact that you got all the source code, and you can make changes (and use their changes) to it. That requirement has always been there, even with plain GPLv2. You have the source. The difference? The hardware

[25] S5 draft GPL v3.

[26] Comment by user 'kop' in the Comments section, see: `http://gplv3.fsf.org/comments/`.

> may only run signed kernels. The fact that the hardware is closed is a _hardware_ license issue. Not a software license issue. I'd suggest you take it up with your hardware vendor, and quite possibly just decide to not buy the hardware. Vote with your feet. Join the OpenCores groups. Make your own FPGA's."[17]

This should come as no surprise to those who follow open source debates. Back in 2003 Torvalds had already expressed that he thought some DRMs were useful, as there seems to be a distinction between the more general DRM and the more restrictive TPM[18]. There are already open source projects that intend to allow artist control[27], while at the same time provide users the tools to use general DRM to monitor and track their works, but not to restrict access, which is what a technical protection is. DRM is a very broad term, and licence management can be considered in that light.

To make matters worse, Stallman has mentioned that Torvalds has no say in what happens to the Linux kernel, and that the kernel developers are the ones who will decide over the licence[28]. This could prove tricky, Torvalds owns the trade mark to Linux, and he is one of the co-authors of the Linux kernel and therefore not the sole proprietor, so he may not be able to make unilateral decisions about licensing.

The problems about licensing only serve to stress the controversial nature of the new draft. As the GPL has become the de facto constitution of the FOSS community, any change to that constitution will generate considerable problems.

5 Conclusion

The final verdict on the draft will undoubtedly come soon, but it seems clear that the draft in its present form may have some problems, and will require further adjustment to make it as widely accepted as GPL v2.

It is clear that the new version of the licence has lost clarity and gained length. This would not be a problem as such if this was followed by a strengthening of the existing provisions, but this does not seem to be the case at the moment. While there are some elegant and clever clauses in the new draft, others seem to be filled with explanatory paragraphs that seem to be out of place in a legal document. The draft could pick some drafting tips from the smaller and more elegant licences that solve some of its issues in a more efficient manner.

[27] For example, Authena, see: `http://authena.org/`.

[28] See Vaughan-Nichols, S.: "Torvalds hasn't ruled out GPL 3 for Linux", Linux-Watch, February 7 (2006), Available @ `http://www.linux-watch.com/news/NS3301105877.html`

References

1. Stallman, R.: "The GNU Operating System and the Free Software Movement", In DiBona C, Ockman S and Stone M, Opensources: Voices from the Open Source Revolution, London: O'Reilly (1999), p.59.
2. Guadamuz, A.: "Viral contracts or unenforceable documents? Contractual validity of copyleft licenses", 26(8) European Intellectual Property Review 331-339 (2004).
3. Moody, G.: Rebel Code: Linux and the Open Source Revolution, London: Penguin (2002), pp.26-29.
4. Jones, P.,:"The GPL Is a License, not a Contract", Linux Weekly News, December 3 (2003).
5. Rosen, L.E.: Open Source Licensing: Software Freedom and Intellectual Property Law, Upper Saddle River, N.J.: Prentice Hall PTR (2004), pp.59-66.
6. Lambert, P.: "Copyleft, copyright and software IPRs: is contract still king?" 23(4) European Intellectual Property Review, (2001), pp.165-171.
7. O'Sullivan, M.: "Making Copyright Ambidextrous: An Expose of Copyleft", The Journal of Information, Law and Technology (JILT) 2002 (3), @ `http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2002_3/osullivan/`.
8. Radin, M.: "Humans, Computers, and Binding Commitment", Indiana Law Journal, Vol.75, Fall 2000, p.1125.
9. Wichman, T.:"Firms' Open Source Activities: Motivations and Policy Implications", Report for the Free/Libre Open Source Software: Survey and Study (2002). @: `http://strategicadvice.net/floss.pdf`.
10. Guadamuz, A.: "Legal Challenges to Open Source Licences", 2(2) SCRIPT-ed 301-308 (2005).
11. Miller, J.S.: "Allchin's Folly: Exploding some myths about open source software", 20 Cardozo Arts & Entertainment Law Journal 491 (2002).
12. Gomulkiewicz, R.:"De-Bugging Open Source Software Licensing", 64 University of Pittsburgh Law Review 75 (2002).
13. Moglen, E. and Stallman, R.: GPL Version 3: Background to Adoption, June 5 (2005), @: `http://www.fsf.org/news/gpl3.html`.
14. Höppner J, "The GPL Prevails: An Analysis of the First-Ever Court Decision on the Validity and Effectivity of the GPL", 1(4) SCRIPT-ed 662 (2004) `http://www.law.ed.ac.uk/ahrb/script-ed/issue4/GPL-case.asp`.
15. Gomulkiewicz R.: "General Public License 3.0: Hacking the Free Software Movement's Constitution", 42(4) Houston Law Review 1015 (2005), pp. 1034-1036.
16. Guadamuz A.: "The Software Patent Debate", 1(3) Journal of Intellectual Property Law & Practice 196-206 (2006).
17. Barr, J.: "Torvalds versus GPLv3 DRM restrictions", NewsForge, February 2 (2006), @: `http://trends.newsforge.com/article.pl?sid=06/02/02/1636216`.
18. Bechtold, S.: "The Present and Future of Digital Rights Management: Musings on Emerging Legal Problems", Becker E; Buhse W; Günnewig D; and Rump N (eds.), Digital Rights Management: Technological, Economic, Legal and Political Aspects, Springer, Berlin (2003), pp. 597-654.

Too many Open Source Licenses! But do the existing licenses adequately encompass the diverse needs and concerns of particular stakeholders?

Darren Skidmore

University of Melbourne, Department of Information Systems.
University of Melbourne, VIC 3010 Australia, d.skidmore@unimelb.edu.au
WWW home page: http://www.dis.unimelb.edu.au

Summary. When are too many Open Source Licenses enough? There is a need to reduce the number of Open Source Licenses that are available, in the Open Source Software license weltanschauung. However there are arguments for the need to add to the suite of licenses because of the failure of the available licenses to deal with contemporary issues of Intellectual Property, such as patents or where there are specific jurisdiction or software practice issues which are required to be addressed, a growth in the number of stakeholders in Open Source Software also demands attention for addition to the available licenses. This article discusses the briefly the issues of definitions of Open Source Licenses, a brief history of the dominant licenses. Then details issues that exist for software such as Intellectual Property, in particular Patents, consumer warranty, changes in jurisprudence, choices of law and forum. Stakeholders of Open Source software are then examined, along with specific examples of concerns particular to those stakeholders, such as fear of effects of consumer warranty and patent infringement, loss of control over where to sell goods and where the issues may be in mandate of a Government.

1 Too many Open Source Licenses

Scheduling the number and diversity of Open Source licenses available, is large and varied, the Commonwealth of Massachusetts compares 52 different licenses [1] the Open Source Initiative (OSI) lists over 58 [2], the Free Software Foundation (FSF) lists and comments on almost 100[3], and the ifrOSS lists over 180 [4]. Each of the Licenses has had its origins in different needs, different times, and / or different agendas of the authors and owners of the source code associated with the license. There has been comment by the OSI to reduce the number of Open Source Licenses that are available [5, 6], companies such as Sun Microsystems retiring licenses [7] and vendors such as Hewlett Packard have chosen to use existing licenses rather than to add a HP branded Open

Source License to the current increasing numbers of licenses in the Open Source weltanschauung.

License proliferation, was also explored in a brief article for the Open Source Development Labs, by Lawrence Rosen, a lawyer and Open Source legal expert, one of only a few people to have written books specifically on Open Source Licenses [8, 9, 10]. Although arguing that there is a genuine need for multiple different licenses, there should be some thought of consolidation and effort put into the development of the licenses listing out nine points to consider [11]:

1. Create an agreed set of wordings for specific concepts.
2. Use proper legal terms in the licenses, including International differences.
3. Encourage simple language.
4. Agree how to translate into languages other than English.
5. "Agree on a Standard Grant of Patent License provision".
6. "Agree on standard patent defense provisions".
7. Decide on attribution definition.
8. Decide on how to deal with jurisdiction.
9. Accept and embrace the diversity of OSS and therefore the many licenses.

Amongst domain literate people in Software development, generally the word "free" is associated with the Free Software Foundation, its philosophies and licenses particularly the GNU GPL and GNU LGPL, and the term "Open Source" is means that the License which governs access to the Source Code is compliant with the Open Source Initiatives' Open Source Definition. Especially with Open Source, this is not always applied in usage by all, however, these are the definitions which will be explained and then used by this paper.

In the initial stages of what would later become Open Source, the late 1960's, there were two styles of what would become to be called Open Source Licenses, which have been implemented by the two licenses which have traditionally dominated the Open Source World, the licenses are the GNU General Public License (GNU GPL), and the BSD style licenses, just purely in terms of the number of available Open Source applications available, with the largest majority being licensed under the GNU GPL. In an analysis of the licenses used by the projects in SourceForge, using the FLOSSmole repository 66.8% of the projects were licensed with GNU GPL, 10.7% the GNU Lesser General Public License (GNU LGPL), which is a slightly modified version of the GNU GPL for use in linking libraries, and 6.9% for the BSD, with the remaining 22 licenses collectively 15.6% of the Total, for 60 297 projects [12]. Using the FLOSSmole query tool , roughly, with the April, 2006 data, the results were 66.9% GNU GPL, 10.3% GNU LGPL, 7.0% BSD, with the remaining 57 licenses totaling 15.8% of the licenses used in the SourceForge projects.

The conditions of the GNU GPL require that any subsequent or derivative works are to be licensed under the GNU GPL which is a license, termed a CopyLeft license, built to the philosophy of the Free Software Foundation and the four freedoms declared in the Free Software Definition [13]. The basis of the four freedoms is that anybody should be able (termed “free” by the FSF) to view, use or modify the source code of any program, for any purpose and to achieve this philosophy the GNU GPL mandates that the source code has to be available for such a purpose, and that this freedom must be continued into any works using the source code.

The BSD style licenses, differ from the GNU GPL, because they allow for derivative works to be licensed under any terms that the licensor desires, apart from the requirement to give attribution for the source of the source code, following in the tradition of the academic community from which it rose [8]. Although there are now many licenses with various definitions, provisions, restraints, requirements and effects, there are two main types as typified by the GNU GPL and the BSD, which this paper will refer to as Reciprocal and Non Reciprocal. Other usages are Reciprocal, Academic, Standards, and Content [8] or licenses with strong / limited CopyLeft, without CopyLeft, restricted options, privileges, content, and fair use of Immaterial goods [4]. A Reciprocal license such as the GNU GPL, requires that subsequent works are licensed under the same conditions, whereas a Non-Reciprocal license such as the BSD allows for another license to be specified for any derivative work.

Software, currently, in general is not actually sold, but is licensed under specific conditions of use [14], these usages can restrict who, how many or what the usage maybe, or in the case of Open Source, allow for others to view and use the source code. The source code and / or application can be licensed to different persons under different conditions (it can also be licensed to the same person under different conditions). Using the ability to license source code to persons under different conditions has created several Dual Licenses, where the application can either be open source, or closed source, depending under what conditions the Licensee wishes to use the software. However for a dual license, to work, the licensor has to own the intellectual property in the software to be permitted to offer a dual license type [15].

The Open Source Definition [16], was created in 1998 by the OSI [17], partially to make Open Source a more business friendly term, pursing the idea that Open Source was a pragmatic and best practice engineering methodology for creating software. This is in opposition to the FSF’s philosophical belief in the fundamental freedom of software usage, and their enforcement of it via the GNU GPL. The Open Source Definition has ten principals which are used to assess if a license if compatible with the OSD, these include access to source code, redistribution rights, no restrictions on where or what the software may be used for and no restrictions on types of software with which the application must be combined.

The term Open Source is not a trademark, nor is it a restricted term, an OSI compatible license is merely a certification mark, given, by the OSI, to a license, which has been assessed by the OSI to meet the Open Source Definition. Therefore a license can be claimed to be Open Source, but not be endorsed by either the FSF or the OSI. Examples of this are the HESSLA license [18], which specifically disallows the code to be used by programs which do not respect human rights, or a Microsoft license such as the MS-LPL [19], which restrict use to the Microsoft Windows platforms only, although the code is open and usable by any. In both of these examples apart from a single restriction, they would meet the OSI definition. For the purposes of this paper, the term Open Source will use the definition as created by the OSI, and if required the term Free to be that as defined by the FSF, but it must be understood, that although the software community in general understands the domain specific meanings of the terms, outside of the domain this is not well understood.

As the use, development and participation in Open Source grew, the number of licenses grew, some to deal with specific organizational or personal aims. An analysis detailing the reasoning behind licenses has not been found, although the FSF does give a brief comment on many licenses, their purpose, and why it is not aligned with the FSF's views [3]. Beyond the needs of the corporation or personal ideology has also been the development of licenses that either are drawn to clarify a legal area or to respond to changes in jurisprudence. The original licenses were written for United States law, and were built upon assumptions of the legal structure of the United States, and the technological and engineering thinking of that time. The licenses have in the main, stood the test of time well, although have never really been exhaustively tested in the Courts. The GNU GPL has never been tested in a US court, but it has been upheld in a German court [20]. It is not a criticism of the authors, in that this paper asks if the original licenses have contemporary issues, but a reflection upon both the changes in the legal environment and the success of their ideas. Of course, these developments have been no different to that of the Non-Open Source Licenses, which have changed frequently in response to changes in both technological usage, and the legal issues of the day.

Although the licenses were created so that the authors could use the protection of copyright, to further their aims in relation to how others could access and use the source code, it is other intellectual property issues which are affecting the licenses and their ongoing development which has and is forcing changes to the wordings and requirements of the licenses. A minor, but important issue is one of trademark, where the license allows free access to the source code, but prevents a derivative work from using a derivative name of the original project, such as the Apache HTTPd and associated projects [21]. However, there is now a major issue with how the licenses manage inclusive code that would implement a process or idea which infringes a patent. In Intellectual Property, a Patent is different from Copyright. There are several significant differences between the two, but for software, there are three important differences. Simplistically: One, a patent is a monopoly on the idea itself, whereas copyright is just over a specific expression of an idea , so if someone has a patent, only they can approve another to use that idea, in copyright someone cannot copy a specific expression, but can come up with their own. Two, patents are not worldwide, a patent may exist in one jurisdiction but not in another, or may be owned by a different person, or be in a different patent application. A persons copyright to a work exists in any nation which has signed the TRIPS agreement [22], (i.e. any nation in the World Trade Organization treaties). Three, a patent claim must be registered and approved by the national patent office in the jurisdiction which the patent is being registered in, whereas copyright does not have to be registered and exists at the point of creation of the work. Even though this creates a semi-central register to check against, the wording of the patent claims is often written in obscure and non normative wording, making a search or claim ambiguous, and tracking extremely difficult [23].

More modern licenses cannot fully protect an open source project from a patent infringement claim, but are more explicit about permissions to patentable ideas if they are introduced into the code. Other aspects of protection against patent infringement are dealt with at the project management of an Open Source application, or via pledges of Patent amnesty or protection by organizations [24, 25, 26, 27]. Issues with Patent infringement is not limited to Open Source, any software can infringe a patent, the largest software vendors are constantly fighting Patent claims, there is the major example of the almost ubiquitous BlackBerry communications device which was almost turned off in the major market of the United States because of a Patent infringement claim. A patent is a monopoly on an idea access to the code base is not always, actually required, if the program caries out a process, this infringes the patent, so being Open Source or Closed Source may not matter. Because of initiatives such as the Open Invention Network, and the patent pledges of vendors with major Patent portfolios Open Source Software is arguably in a safer position in regards to patents than other software developers using closed source licenses.

Because software is a digital good, and can be easily moved around the world, this has created a need to deal with issues which are required, when a product can be supranational. Some legal jurisdictions do not have definitions of "derivative work", distribution, or consumer warranty laws which may make the entire license invalid due to the wording of the license [28]. Although less of an issue currently, there has been concerns with transferring software around the world, particularly in the case of exporting encryption algorithms and the applicable code, because this was seen as exporting weapons, under the Wassenaar Arrangement [29], therefore some licenses have include clauses specifying compliance to a specific countries export laws ([30], Clause 6). Also of importance is determining in which jurisdiction or forum any dispute would be heard by the courts, this was of particular importance in the discussion about deciding on the creation of an Open Source License in countries other than the US[31, 32]. Separately to the international issues, maybe the determining or specifying under which type of law will be used to interpret the license, options include contract law, sale of goods law, intellectual property law [8].

Besides changes in jurisprudence, there has also been technological progress and innovation, which has affected the nature and interaction of software application usage since the inception of the first open source licenses. At the very lest, this includes the ease of both transferring source code and ideas around the globe as well as the ability to interact seamlessly has changed vastly since the late 1960's. The communications infrastructure, the collaboration infrastructure, the documentation and the range of persons and organizations contributing have all changed dramatically in the last 25 years. Besides changes in contribution and collaboration, there are issues such as that of virtualization and use of the Internet to distribute not just the application program, but the results of the algorithms has allowed some organizations to exploit Open Source applications, but not share or return their own innovations to the community. Currently the GNU GPL is going through a process of revision (to version 3) to deal with issues that the FSF sees as abuse of GNU GPL licensed software [33]. Although individual applications are, generally, licensed under a specific license there is an issue of distributions or combinations of applications where a number of licenses are used, in the case of RedHat 7.1 over 15 different open source licenses were used [34], or distributions have been asked to desist because someone believes they are not adhering to the complexity of the licensing requirements [35].

For those who need to choose an Open Source License, some of these issues may be moot, as they may be forced to use a particular license, simply because if they need to use an existing piece of software or use parts of the source code, they have to comply with a reciprocal license, such as the GNU GPL [12]. There may also be a choice based upon the firms business model for production of software leading them to a choice of license [36, 37]. If vendors or small developers wish to build upon or take code from other Open Source projects they may prefer to use Non-Reciprocally licensed code, since they are free to use it and relicense to suit their own needs [38]. Even for Vendors of software, firms and developers, in situations where it is possible for choices to be made about the licenses there are differing concerns, or in particular the priority of concerns about the effects of use of Open Source and in the particulars of the license that is chosen. In informal discussions with people and vendors this varies with the size, sophistication and type of developer or business.

For smaller vendors or developers, part of the concern about releasing the source code through an Open Source license seems to be the loss of control once the software enters the Open Source world, where anyone can take and use the source code. Although willing to accept their own risks and implications of what they designed or created the code for it is the downstream usage, where they cannot foresee or design for that causes concern. For some, it is the fear of being in breech of consumer warranty laws, (e.g. the goods must be fit for purpose) so they fear being held legally responsible for code used in a situation or a purpose over which they have no control. For others it is fear of a jurisdiction, a person might not wish to offer a product into a country which has a history of litigation, and so might make a business decision not to sell, support or operate in that country because the risks and costs of litigation outweigh the benefits accrued from selling into that jurisdiction. Similar, under patent law, valid and legal algorithms in one jurisdiction may be infringing in another, the author or owner of the code may not have control over the source code if an open source license is used.

The situation is perhaps not as bleak as is portrayed, large vendors such as Red Hat, IBM, Oracle, Novell and Sun, are extending into Open Source, and as previously mentioned have created at least the start of some initiatives to respond to issues of patents that should provide some protection to other developers, although research and care is still very much required. As an issue in dealing with Patents, for software is a much larger issue than this paper can deal with or solve. On the issue of consumer warranty, although no references have been found, it is unlikely that any software vendor especially the large vendors, would want a general legal precedent set on the liability of software, as it would be possibly somewhat disastrous for the entire industry. Therefore if any action was brought, presumably there would be a high chance of participation by the major vendors.

Beyond Vendors, there are choices for those who need to use the software, the consumers and organizations. Some organizations would actively use the nature of Open Source to continually modify and improve their software, depending on the application though; most do not seem to need to modify their software. A reciprocal license may be extremely useful, in that although perhaps costs in support are higher, there is a decrease in risk of using the product. Access to the source code gives some protection against a vendor not being able to support the product, there is leverage against price increases in the future and more subtlety there may be a lowering of business risk as others adopt the same software and processes. It is not to claim that Open Source Software is free of cost, because it is not; nor that Open Source Software is free of risk, these are both due diligence tasks which have to be considered, but that is the same for any software adoption or acquisition, indeed it should be part of any ICT governance decision.

There are also other factors, such as public policy choices, in the example of the European Union. The European Commissions IDABC has initiated discussion [32], consultation and collaboration on the creation of a European Open Source License. This process has advanced the EUPL, which is currently at draft Version 0.2 [39]. One outcome is to ensure that by using a reciprocal license, if the EU has software created then neither the EU nor its citizens will have to pay for the source code a second time, nor any derivatives based upon the work. The jurisdiction of any dispute is to be that of the jurisdictional laws of the licensors courts, although an earlier version specified EU Law. The EUPL also attempts to deal with an issue, not dealt with well by other licenses, which is in terms of mixing of different types of licenses, specifically stating that although the EUPL is a reciprocal license, that it will publish a list of like styled licenses which derivative software may be relicensed under if this is needed. Therefore making it possible to add EUPL licensed code which by the reciprocal terms of the license, should be licensed under the EUPL be incorporated or mixed with GNU GPL licensed code with the resultant codebase licensed under the GNU GPL.

Open Source Software licenses are useful for the licensing of source code, but there is also a need to consider supporting associated Intellectual Property, such as documentation and the standards associated for the transfer of data. Licenses certainly exist for documentation, such as the GNU Free Document License [40], or the range of Creative Commons licenses [41]. As do licenses for the standards which Open Source Software is based or uses to communicate [8].

The Creative Commons (CC) initiative, although built for content, rather than software, has a good approach for people who wish to share content. CC have tackled this problem with three aspects. The first was to create multiple license types, where the author selects from a limited variation of choices to suit their needs, such as reciprocal licensing, allowing modifications, and / or commercial use. Along with being able to quickly select the license conditions, CC also has created infrastructure to assist in the management of the licenses, by giving three explanatory documents based upon the license; one, a plain language statement of the license, two, a proper legal license, and three a digital version of the license that can be understood by a computer program (properly coded). The third aspect was to have the license written in multiple jurisdictions, so that they would comply with local law [41].

Further research is needed in many areas. Perhaps deeper investigation of the types of users of Open Source to determine their views and needs, the purpose of this paper was to raise the issues, a more detailed examination is needed. Most of the discussion has been about reciprocal type licenses, only briefly discussing the advantages for non-reciprocal licenses. The practicality and / or methods of being able to use mixed licensing requires investigation, the EUPL at least attempts to deal with this issue, but this is only for EUPL to other reciprocal licenses, not for existing ones, [11] although briefly mentioning the issue does not provide any guidance on solutions. Similarly for Dual licenses are another area for research. Disputes where there is a licensor in one jurisdiction and the application is used perhaps by a third party in another could be a complex area of academic jurisprudence. In the area of Patents, the author is unsure of where to start, certainly the issue of whether software or business process patents should be allowed has been argued in the European Parliament, but even if this was disallowed there are other major jurisdictions which allow them, so possibly a solution involving better articulation or changes in software engineering are required, although unlikely. Development of a Creative Commons style approach might also be investigated.

In conclusion, similar to Rosen, it is not to say that the current available licenses should be removed or are inadequate to the tasks of modern software engineering. The existing licenses, particularly the dominant ones are well used, have at least some legal protection, and are the basis of functioning software all over the world. To unravel or relicense the existing code bases would be a Herculean task, and of doubtful return on the energy expended.

Perhaps some of the more obscure or obsolete could be retired, which is already happening as organizations move to other licenses. However recognition does need to be made that the changes in the composition of the participants, the jurisprudence, and globalization of software usage has created the need make changes to the text of the licenses or the creation of more modern ones. The EUPL is a very good example, applicable internationally, but it is only a reciprocal license, at the very least a non-reciprocal version is also needed.

References

1. Open Source Licenses - Quick Reference Chart, @ `http://www.mass.gov/itd/legal/quickrefchart.xls`, Editor. 2004, Commonwealth of Massachusetts. p. Chart in Spreadsheet format (.xls) of ˜50 Open Source Licenses and their attributes.
2. Open Source Initiative, Open Source Initiative OSI - Licensing. 2004, Open Source Initiative.
3. Free Software Foundation, Licenses, in Licenses, Free Software Foundation, Editor. 2005, Free Software Foundation.
4. ifrOSS, License Center, I.f.r.d.f.u.O.S. Software, Editor. 2005, Institute für rechtsfragen der freien und Open Source Software.
5. Open Source Initiative, Charter for License Proliferation (LP) Committee of the Open Source Initiative (OSI). 2005, Open Source Initiative,.
6. Open Source Initiative, License Proliferation. 2005, Open Source Initiative,.
7. Phipps, S.: Addressing Proliferation: Deeds not just Words. 2005, Sun Microsystems. p. WebBlog Post.
8. Rosen, L.: Open Source Licensing Software Freedom and Intellectual Property Law. 2004, Upper Saddle River, NJ: Prentice Hall. xxii,396.
9. St. Laurent, A.M.: Understanding Open Source and Free Software Licensing. 2004: O'Reilly.
10. Välimäki, M.: The Rise of Open Source Licensing - A Challenge to the Use of Intellectual Property in the Software Industry. 2005: Turre Publishing. 263.
11. Rosen, L.: License Proliferation. 2005, Open Source Developers Lab,.
12. Weiss, D.: Quantitative Analysis of Open Source Projects on SourceForge. in The First International Conference on Open Source Systems. 2005. Genova, Italy: ECIG.
13. Free Software Foundation, The Free Software Definition.
14. von Krogh, G. and von Hippel, E.: Special issue on open source software development. Research Policy, 2003. 32(7): p. 1149-1157.
15. Olson, M.: Dual Licensing, in Open Sources 2.0 : The Continuing Evolution, C. DiBona, D. Cooper, and M. Stone, Editors. 2006, O'Reilly: Sebastopol, Calif. p. 71-90.
16. Open Source Initiative, The Open Source Definition. 2004.
17. Open Source Initiative, Open Source Initiative OSI - OSI History, Open Source Initiative, Editor. 2005.
18. Hacktivismo, The Hacktivismo Enhanced-Source Software License Agreement, Hacktivismo.

19. Microsoft, Shared Source Licenses, in Microsoft Shared Source Initiatve. 2005, Microsoft.
20. Harald Welte vs Deutschland GmbH. 2004, District Court of Munich.
21. Apache Software Foundation, Apache License, Version 2.0. 2004, Apache Software Foundation,.
22. World Trade Organization, TRIPS (Trade-related aspects of Intellectual Property rights). 1994, World Trade Organization.
23. Cohen, J.E. and Lemley, M.A.: Patent Scope and Innovation in the Software Industry. The California Law Review, 2001. 90(1).
24. Open Invention Network, Open Invention Network formed to promote linux and spur innovation globally through access to key patents, Open Invention Network, Editor. 2005, Open Invention Network,. p. Press Release.
25. Red Hat, Statement of Position and Our Promise on Software Patents, Red Hat, Editor. 2004, Red Hat,.
26. Novell, Novell Statement on Patents and Open Source Software. 2004, Novell.
27. IBM, New IBM Initiative Advances Open Software Standards In Healthcare and Education. 2005. p. Press Release.
28. Fitzgerald, B.: Legal Issues Relating to Free and Open Source Software, ed. B. Fitzgerald and G. Bassett. Vol. 1. 2003, Brisbane, Queesnland, Australia: Queensland University of Technology School of Law. viii, 128, 1.85Mb.
29. Wassenaar Arrangement Secretariat, Wassenaar Arrangement on Export Controls for Conventional Arms and Dual-Use Goods and Technologies: Guidelines & Procedures, including the Initial Elements, in Wassenaar Arrangement, W.A. Secretariat, Editor. 2004, Wassenaar Arrangement: Vienna. p. 35.
30. Apple Computer Inc, Squeak License, Squeak.org.
31. National ICT Australia, Australian Public Licence B Version 1-1. 2004, National ICT Australia.
32. Dusollier, S., P. Laurent, and Schmitz, P. E.: Open Source Licensing of software developed by the European Commission, in IDA/GPOSS Encouraging Good Practice in the use of Open Source Software in Public Administrations, UNISYS, Editor. 2004, European Commission. p. 67.
33. Free Software Foundation, GPLv3 First Discussion Draft Rationale. 2006, Free Software Foundation.
34. Wheeler, D.A.: More Than a Gigabuck: Estimating GNU/Linux's Size. 2002.
35. Smart, C., Kororaa Accused of Violating GPL, K. Forum, Editor. 2006, Kororaa. p. Web Page Post.
36. Bonaccorsi, A. and Rossi, C.: Licensing schemes in the productionand distribution of Open Source software. An empirical investigation. 2003, Institute for Informatics and Telematics.
37. Bonaccorsi, A. and Rossi, C.: Open Source Software, intrinsic motivations and profit-oriented firms. Do firms practice what they preach? in The First International Conference on Open Source Systems. 2005. Genova, Italy: ECIG.
38. Michaelson, J:, There's no such thing as a Free (software) lunch. ACM Queue, 2004. 2(3).
39. European Community, European Union Public Licence. 2005, European Commission.
40. Free Software Foundation, GNU Free Documentation License (GFDL), Version 1.2. 2002.

41. Creative Commons, Licenses Explained: Creative Commons. 2005.

Free & Open Source Software, Human Development and Public Policy Making: International Comparison

Mehmet Gençer[1], Bülent Özel[2], Harald Schmidbauer[3], and Vehbi Sinan Tunalıoğlu[4]

[1] İstanbul Bilgi University, Dept. of Computer Science mgencer@bilgi.edu.tr
[2] İstanbul Bilgi University, Dept. of Computer Science bulento@bilgi.edu.tr
[3] İstanbul Bilgi University, Dept. of Business Administration harald@bilgi.edu.tr
[4] İstanbul Bilgi University, Dept. of Computer Science vst@bilgi.edu.tr

Summary. Despite the growing body of research on inner workings of FOSS development, there are few studies on its relation with broader developments in society. In this study we have attempted a preliminary investigation of (1) how FOSS prevalence is related to economic and human development indicators of countries, and (2) whether public policies regarding FOSS emerge in a consistent relation with these indicators in several clusters of countries constructed from the United Nation's human development index. Our results point to relative significance of non-economic factors in FOSS adoption and lack of consistent policies among public agencies.

Keywords: Free and Open Source Software (FOSS) adoption, human development index, public policies

1 Introduction

Widespread use of Free and Open Source Software (FOSS) is a key strategic target not only within the FOSS community, but also for public policy makers, among others because of its advantages in promoting competition in the software market and reducing IT costs in public spending. Despite the growing body of research about the merits of FOSS in producing software and its advantages in providing an alternative governance mechanism in production, factors which affect its adoption are not well known. The development of related strategies (or policies whether public or private) requires a better understanding of how FOSS adoption interacts with various aspects of society.

Our method consists of an international comparison of the prevalence of FOSS, in particular: We seek to explain the variability in the use of FOSS across countries on the basis of several variables related to human development, economics and transparency, and the Internet infrastructure. Furthermore, we look into the server software (free or proprietary) used in several public organizations in countries all over the world, and relate it to a classification of countries we develop on the basis of cluster analysis.

We relate FOSS prevalence to human development, since the FOSS mode of production requires skills and abilities, rather than capital assets. Human development, which, according to the first Human Development Report [1], is "a process of enlarging people's choices", has to be quantified in our context. Despite some criticism (see, for example [2, 3]), we choose the human development index (hdi), which is published along with Human Development Reports by the United Nations Development Program (UNDP).

It is straightforward and plausible that there may be a relation between the access to information technology and the prevalence of FOSS. High-quality Internet infrastructure is expensive. This leads us to the necessity to consider also the impact of economic reality, with gross domestic product (gdp) per capita and the transparency index [4] as proxies, on the prevalence of FOSS.

This paper is organized as follows. Section 2 attempts to explain the prevalence of open source across countries on the basis of selected variables. Countries are clustered with regard to selected variables in Section 3. Section 4 looks into the server software variety used by public organizations of countries, which leads us to some remarks about possible FOSS policies in Section 5.

2 Explaining Open Source Software Prevalence

We operationalize the prevalence of FOSS in a country as the number of hits in the country-specific Google search of the keyword "open source". This variable will obviously be correlated with the size of a country in terms of its population. We hypothesize that the prevalence of open-source software is also related to

- economic characteristics: gdp per capita, transparency index;
- other factors such as the level of education.

Our investigation of the prevalence of open source software is based on regression models with doubly-logged Google counts of "open source" as dependent variable, which we determined in a survey in April 2006, and a selection of independent variables from the following list (which includes also information on the transformation applied):

variable	source
log(log(population))	CIA World Factbook 2006
log(gdp per capita)	CIA World Factbook 2006
human development index (hdi)	Wikipedia; keyword: human development
transparency index (tri)	Transparency International, Annual Report 2004
log(internet usage)	CIA World Factbook 2006
log(internet hosts)	CIA World Factbook 2006

In addition to the variables mentioned in this table, it was found plausible to include also the variable *relative Internet usage*, defined as Internet usage per 1000 population, in the analysis.

A full regression model, with all variables included as regressors, and fitted to 116 cases (countries) for which the entire data set was available, has an explanatory power (i.e., R^2) of 67%, but leaves many variables insignificant[5]. Using stepwise procedures to fit a reduced model with fewer, and significant variables, leads to the following model, fitted to 129 cases:

$$\begin{aligned}\texttt{log(log(count.open.source))} = \underset{(-2.13)}{-1.546} &+ \underset{(3.27)}{0.8155} \times \texttt{log(log(population))}\\ &+ \underset{(4.11)}{0.0485} \times \texttt{log(internet.hosts)}\\ &+ \underset{(5.44)}{1.3070} \times \texttt{hdi}\end{aligned}$$

(The numbers in parenthesis are the t values of the estimates.) This model also has an R^2 of about 67%, while retaining only significant regressors. A comparison of the regressors in this model with the list of variables in the table above reveals that Internet usage, relative Internet usage, the transparency index, and gdp per capita are not significant in explaining the variability of Google counts of “open source” across countries.

It may come as a surprise that gdp per capita has so little explanatory power on the prevalence of open source software. To reassess this result, another regression model, with gdp per capita substituted for hdi, can be fitted to the data set. The explanatory power of such a model is substantially smaller than the model cited above; its R^2 is reduced to 61%. We can thus conclude that economic well-being can indeed explain a certain share of variability contained in the dependent variable, but the explanatory power is further enhanced by considering non-economic factors. This gives a clue about the importance of non-economic factors in accounting for the prevalence of open-source software.

[5] A full account of our estimation results can be found in Gencer, Özel, Schmidbauer, and Tunalıoğlu [5].

3 Clustering Countries

The goal of this section is to classify the countries with respect to the variables discussed earlier in Section 2. This will enable us to obtain insight into the way the countries are different, and provide us with a means to determine the cluster membership of a country. The results explained below are based on a k-means cluster analysis with standardized variables. A cluster number of 5 was found optimal to account for differences between the countries. The five clusters can be verbally described as follows:

- Cluster 1: Low relative Internet usage, mid-range hdi; mid-range open-source count reflecting the mid-range size (i.e., population) of the country. This cluster contains 24 countries, among them Bulgaria, Latvia, Croatia, Vietnam, Armenia, and Iran.
- Cluster 2: Low relative Internet usage, mid-range hdi; low open-source count reflecting the smaller size of the country as compared to Cluster 1. This cluster contains 33 countries, among them Belize, Samoa, Tonga, the Bahamas, Panama, and Puerto Rico.
- Cluster 3: Low relative Internet usage, mid-range hdi; higher open-source count reflecting the larger size of the country as compared to Cluster 1. This cluster contains 18 countries, among them the Russian Federation, Poland, Brazil, China, Thailand, and Romania.
- Cluster 4: Highest relative Internet usage, highest hdi; very high open-source count, almost irrespective of the population size. This cluster contains 35 countries, among them the United States, Germany, France, the United Kingdom, Canada, and Italy.
- Cluster 5: Very low relative Internet usage, very low hdi. This cluster contains 30 countries, among them Sierra Leone, Sudan, Pakistan, Myanmar, Laos, and Kenya.

The countries mentioned in each cluster are those with the highest Google counts of “open source” in their respective cluster[6]. Figure 1 shows a classification tree. It illustrates how the cluster membership of a country is determined. If the condition on a fork is fulfilled, the left branch applies, and vice versa.

4 Server Software Varieties in Organizations Across Country Clusters

Is there a cluster-specific pattern in the server software variety used by governmental organizations across countries? To investigate this question, we will look into an additional variable, namely the server software variety (f = free,

[6] For a full list of the cluster analysis results, see Gencer, Özel, Schmidbauer, and Tunalıoğlu [5].

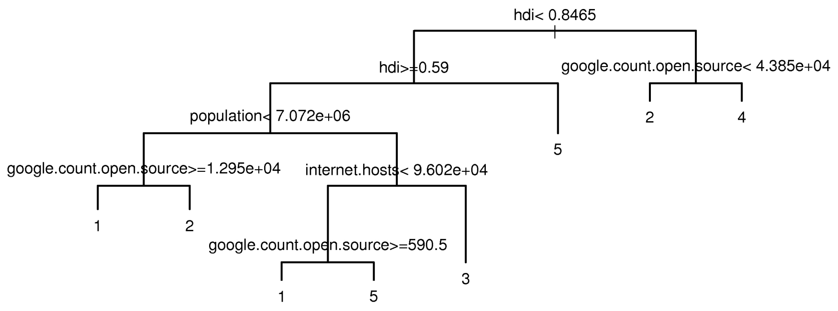

hdi< 0.8465
hdi>=0.59
google.count.open.source< 4.385e+04
2
4
population< 7.072e+06
5
google.count.open.source>=1.295e+04
internet.hosts< 9.602e+04
1
2
google.count.open.source>=590.5
3
1
5

Fig. 1. Classification tree

p = proprietary) used in several prominent public organizations / facilities of a country for which the data was readily accessible; these are the bureau of statistics, the central bank, the finance regulator, the foreign ministry, and the postal services[7].

The cross-classification of the country data in Table 1 allows to test the null hypothesis that cluster membership and the server software variety are independent. The table gives the numbers of countries for which the organization mentioned is in each combination of cluster membership and server software variety. Countries for which no data were available were omitted in this table. As explained in Section 3, the main difference between Clusters 1, 2 and 3 is in population size. In order to increase the power of the χ^2 test and thus facilitate detecting deviations from the null hypothesis of independence, it is justified to collapse Clusters 1, 2, and 3 into a single cluster and contrast the three remaining clusters. The result of this procedure is shown in the last row of Table 1. The conclusion from this table is that the null hypothesis of independence is rejected only in the case of the bureau of statistics. An inspection of this part of the table reveals that the cluster of low-developed countries has more free server software than the other clusters. This is in contrast to the generally lower average open-source count in this cluster.

	central bank		bureau of statistics		finance regulator		foreign ministry		postal services	
cluster	f	p	f	p	f	p	f	p	f	p
1, 2, 3	21	39	6	17	12	16	11	7	9	11
4	9	12	6	18	13	11	16	6	14	10
5	7	9	4	1	3	2	3	3	3	2
p-value	0.72		0.04		0.63		0.52		0.64	

Table 1. Cluster membership and server software variety of organizations

[7] Own survey; April 2006.

A related question is: Is there evidence that the variety of server software used in one organization of a country is associated with that used in another one? We found no case in which the hypothesis of independence of server software variety in different organizations could be rejected. In this sense, we found no evidence of a server software policy within countries.

5 Towards Policy Making

As explained in the previous section, we found no evidence that governments pursue a server policy. This raises the question whether a top-down policy is desirable at all. Based on macro-economic models, Comino and Manenti [6] have recently shown that subsidization of FOSS adoption is welfare-decreasing, whereas mandatory adoption (e.g. in schools) may succeed, depending on the society. Information campaigns, on the other hand, were found by Comino and Manenti to enhance welfare.

This is in line with our findings, which provide empirical evidence that non-economic factors are an important determinant in FOSS prevalence. We interpret our empirical results such that information campaigns and educational incentives enhance welfare more efficiently than subsidization or mandatory adoption approaches.

6 Summary and Conclusions

In an effort to assess the determinants of the prevalence of Free and Open Source Software (FOSS), we tried to account for the variability in Google counts of the keyword "open source" across countries worldwide using a regression model with several regressors characterizing human development, the economic situation, and the Internet infrastructure of the countries considered. It turned out that the human development index, population, and the number of Internet hosts are significant, while Internet usage is not.

A cluster analysis of the country data pointed at the interplay of human development and FOSS prevalence. We found that the bureau of statistics is the only example among several prominent organizations of a country whose server software variety (free or proprietary) is significantly related to the cluster membership of countries.

We conclude from our models that gdp per capita is less important in accounting for prevalence of FOSS, and that focusing only on increasing Internet usage will not promote the use of FOSS. It is rather information campaigns and educational incentives that may be conducive to widespread use of FOSS.

We mentioned earlier that FOSS introduced its own mode of production, in which human capital plays the most important role. The study by Streeten [7] emphasizes that "human beings are both ends in themselves and means of production". With respect to that, we suggest that FOSS be further discussed under the aspect of its contribution to human development.

References

1. UNDP: Human Development Report 1990. Oxford University Press, New York, Oxford (1990)
2. Kelley, A.C.: The human development index: “handle with care”. Population and Development Review **17** (1991) 315–324
3. Srinivasan, T.N.: Human development: A new paradigm or reinvention of the wheel? The American Economic Review **84** (1994) 238–243 Papers and Proceedings of the Hundred and Sixth Annual Meeting of American Economic Association.
4. Transparency International: Annual Report 2004 - The Coalition Against Corruption. Transparency International (2004) Available at `http://www.transparency.org`.
5. Gencer, M., Özel, B., Schmidbauer, H., Tunalıoğlu, V.S.: Free and Open Source Software, Human Development and Public Policy Making: International Comparison. (2006) Research Report, İstanbul Bilgi University.
6. Comino, S., Manenti, F.M.: Open source vs closed source software: Public policies in the software market. (2003) Available at: `http://opensource.mit.edu/papers/` cominomanenti.pdf.
7. Streeten, P.: Human development: Means and ends. The American Economic Review **84** (1994) 232–237 Papers and Proceedings of the Hundred and Sixth Annual Meeting of American Economic Association.

F/OSS in European National Programmes

Bülent Özel[1], Julia Velkova[2], and Manon van Leeuwen[3]

[1] İstanbul Bilgi University - The Scientific & Technological Research Council, Turkey `bulento@bilgi.edu.tr`
[2] Bulgaria Information Society, Bulgaria `julia@isoc.bg`
[3] FUNDECYT, Extremadura, Spain `manon@fundecyt.es`

Summary. Deriving upon (1) a study on current state of F/OSS in more than 10 European countries, reported by tOSSad (towards Open Source Software adoption and dissemination) Project partner institutes and (2) the discussions from the report of a tOSSad workshop on F/OSS national programmes, this position paper gives a brief on status of F/OSS adoption in European Public Administrations.

1 Introduction

This study is an outcome of the EU FP6 funded project - tOSSad (towards Open Source Software adoption and dissemination) - and it is produced within, "F/OSS in National Programs" workpackage of the project. This workpackage aims to start up national programmes for improved usage of F/OSS in some (at least one) of the project target countries and develop guidelines that will be used for F/OSS adoption in the public sector.

The 'tOSSad F/OSS in National Programmes Workshop' took place at the Second Free Software International Conference - International Meeting about Free Knowledge that was held in Merida, Extremadura province, Spain on 25th and 26th of October, 2005. Prior to that workshop a study on current state of F/OSS adoption in more than 10 European countries, reported by tOSSad Project partner institutes.

In the following section findings of aforementioned studies and the workshop is given rather in a summarized manner. However, a full report [1] of the workshop along with F/OSS status on covered European countries is available online[4].

[4] Please see `http://www.tossad.org/tossad/publications`

2 On F/OSS in European National Programmes

- F/OSS is easier to be accepted and adopted in the non-governmental, educational and business sectors than in the Public Administrations. The business is more flexible, more cost-saving oriented, and more open to innovations from which they can benefit relatively quickly and directly.
- The unawareness about F/OSS and the not-knowing-that-they-use F/OSS users do not want to be the first to adopt F/OSS unless other organization or administration adopts it first. This is due to the lack of will to change and fear of who will support the software.
- Users, and administration staff/employees should be involved in decision making process. Otherwise, a high resistance to change is observed.
- Vendor lock-in is a significant barrier for the adoption of F/OSS in Public Administrations. What is more, the users that do not want to use F/OSS do not think they are locked, as well as the opposite statement is also true. The cost for changing a system/software becomes extremely high and the organizations/administrations therefore often stay with the old system.
- Due to the lack of profesional ICT companies running on F/OSS mostly it is not possible to e.g. offer the the same software and service support as proprietary software vendors can. This means that the public administration bodies themselves have to train and educate their staff in order to keep the systems running.
- The one-year budget cycle in Public Administration is a very serious problem because it leads to the impossibility to evaluate the further savings from the migration, while the cost for this, which is also hardly estimable, has to be paid immediately.
- There exists lack of good or enough cooperation between the governments/public administration and the private sector in order to provide the best F/OSS solutions.
- One major problem is to convince a structure, like Government or Public Administration, on why to migrate to a completely different platform. The problem becomes more complicated due to the lack of enough specialized software for doing the management and accounting in the administrations which could be a good reason to migrate to F/OSS.
- The German practices can be considered as good approach, where the majority of initiatives on F/OSS adoption in Public Administrations, although being made mainly at regional and local levels, try to make migration to F/OSS of basic software components on server and desktop systems. In this manner, wider adoption of F/OSS can be more easily achieved than if doing a complete migration.
- Books on F/OSS have to be published and promoted in order to keep and raise awareness which is a serious problem for the wider adoption of F/OSS.
- The public sector needs a decision model that will assist this sector when upgrading or purchasing new software. This model should help them in

calculating the cost savings (support, educational needs, training, number of errors etc.) and the benefits from using a particular type of software.
- The central ICT-departments seem to show little interest towards a shift to F/OSS systems (primary when it came to office related software or servers).

3 Conclusion

In short,current tOSSad studies on F/OSS adoption in European National Programmes conclude that there seems to exist will in some of the European Public Administrations to use F/OSS, but it is limited to regional and local administrations, and still a clear public policy on wider usage of F/OSS is missing.

References

1. Velkova, J., Özel, B.: F/OSS National programmes Workshop Report (Deliverable-D06). (2005) Available at `www.tossad.org`.

Using a WikiWikiWeb to support Open Source Adoption in Small and Medium Sized Companies

Oliver Strauß, Oliver Höß, and Anette Weisbecker

Competence Center Software Management
Fraunhofer Institute for Industrial Engineering (IAO)
Nobelstr. 12, 70569 Stuttgart, Germany
Tel. +49 (0) 711 / 970 {- 2406 | - 2409 | -2400},
Fax. +49 (0) 711 / 970 - 2401
E-Mail: {Oliver.Strauss,Oliver.Hoess,Anette.Weisbecker}@iao.fhg.de

Summary. In this paper we describe the use of a Wiki [1], which is a special kind of light weight content management system, in the management of information and knowledge that arises during the adoption of an Open Source Software (OSS) system. Wikis allow users to easily create web pages and links between those pages. The fact that all pages can be edited by everyone encourages a collaborative approach to content creation. Under the right circumstances and with a collaborative corporate culture this participative approach actively involves the users to participate in the adoption process and to help each other by providing their knowledge in the Wiki. We describe the role that Wikis can play in OSS adoption projects and propose some measures that help to successfully adopt Wikis for this purpose. Finally we give a short overview over the ReqMan Wiki [2] which is a Wiki engine especially suited for the semi-structured information that arises in OSS adoption projects.

1 Introduction

The landscape of high quality Open Source Software (OSS) is steadily growing. Open Source gains more and more importance in the application domain (e.g. Open Office) as well as in software development where OSS is available in the form of frameworks, libraries and components. The adoption of OSS in a commercial environment requires careful planning of the whole application lifecycle from evaluation and selection [3] of a product or library, to examination of legal issues, migration from other systems, training and roll-out, operation, maintenance and support. During this lifecycle a lot of information and knowledge is produced that can be valuable to users in their daily work, to developers and to management. Elicitation, management and dissemination of this body of knowledge are important success factors for the adoption and acceptance of OSS in a company. Wikis, most of them being Open Source as well, can be used in the context of OSS adoption to collect and distribute support information and knowledge about new software systems and motivate users to actively participate in the process of knowledge creation.

After a description of the classical Wiki functionality in section 2 we discuss the use of Wikis in the context of OSS adoption with its benefits and dangers in section 3. For the adoption of a Wiki system in a company we propose measures that help to create the circumstances that make the successful use of a Wiki possible (section 4). Finally in section 5 we introduce the ReqMan Wiki [2] which offers features useful for the purpose of OSS adoption and OSS support. A summary in section 6 concludes the paper.

2 Collaborative content management with Wikis

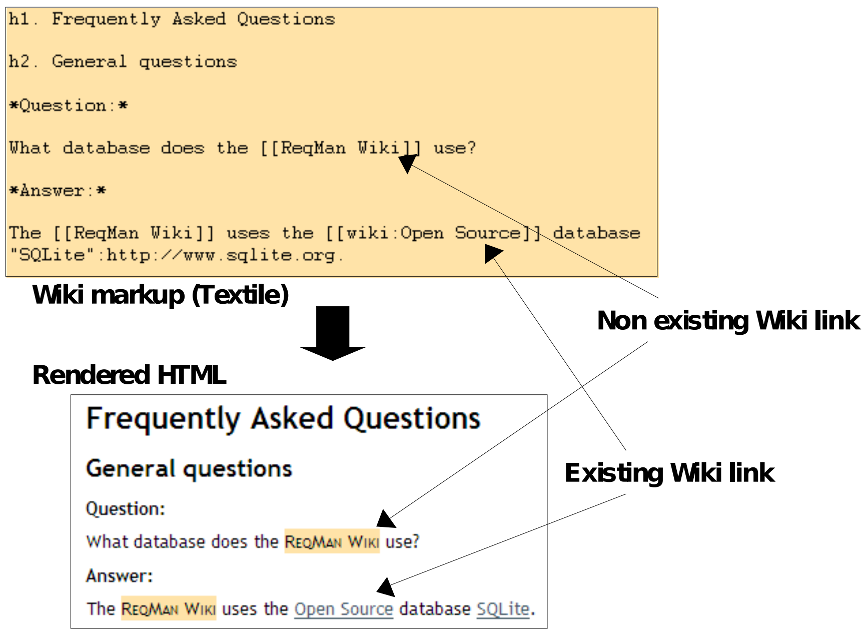
A Wiki is a web-based application that allows its users to produce and interlink content in an easy to use way. The first Wiki [4] was developed by Ward Cunningham in 1995 and in the meantime numerous variants and extension have emerged that expand on the basic idea of a Wiki: enable the users to easily create web pages and links between these web pages in order to collectively create a growing and interwoven body of content. The most prominent example of a public Wiki is Wikipedia [5], a community effort to create a huge online encyclopedia.

The content of Wiki pages is typically entered in plain text that is enhanced with a limited set of markup commands. The user thus can create headings, paragraphs or emphasize parts of the content without the knowledge of HTML. The markup syntax is individual to each Wiki implementation, but there are also standard markup formats like Textile [6] or Markdown [7] that are used by some Wikis (see Figure 1).

Each web page inside a Wiki has its unique name that can be used to reference the page from within other pages. Page references are indicated either by using a special way of spelling the page name (e.g. by using camel case words like in “ThisIsAPageName”) or by using special markup (e.g. “[[This is a page name]]”, see Figure 1). If a reference points to a non existent page, that page can be created by following the link to the missing page. This enables users to create incomplete content that can be augmented later and that clearly indicates the missing information that needs to be added.

Easy markup and page links make up the basic functionality of a Wiki and provide users with the means to create a self organizing web of information. In addition to these basic functions many Wikis provide additional functionality such as:

- searching via a full text search engine
- navigational aides like a list of pages that link to the current page (‘backlinks’)



```
h1. Frequently Asked Questions

h2. General questions

*Question:*

What database does the [[ReqMan Wiki]] use?

*Answer:*

The [[ReqMan Wiki]] uses the [[wiki:Open Source]] database
"SQLite":http://www.sqlite.org.
```

Frequently Asked Questions

General questions

Question:

What database does the ReqMan Wiki use?

Answer:

The ReqMan Wiki uses the Open Source database SQLite.

Fig. 1. Example for Wiki markup and Wiki links

- version history of each pages' content
- means to categorize pages

There are a large number of Wiki implementations available today that vary in the features they provide. Most of them are Open Source projects. For a comparison of various Wiki implementations see for example [8].

The light weight approach to the collaborative management of content provided by Wikis is especially suitable for small and medium enterprises (SME) that oftentimes are hesitant to make big investments in centralized knowledge management systems.

3 Wikis and Open Source Software adoption

During the adoption of an Open Source system a lot of knowledge and information is being produced. This comprises unstructured information like user experiences as well as semi-structured information like OSS application descriptions, evaluation reports, usage reports, requests for enhancements or lists of frequently asked questions (FAQ). Most of the time, this information is not made explicit. If it is documented, it is often documented in a way (e.g. paper based) that makes searching for specific information and cross linking between related information items difficult. Full blown knowledge management systems can be used to overcome these problems but many of those systems are complicated and hard to maintain.

Wikis on the other hand are well suited for documenting unstructured or semi-structured information, such as FAQs or experience reports. All Wiki users are allowed and encouraged to create new Wiki pages, enhance and add to existing pages and correct the errors they notice. The body of information is not maintained by a few designated knowledge workers but collectively by all participating users.

The Wiki concept idealistically builds on users that constructively work together. In such environments it can show its full potential. If this condition is not fulfilled Wikis can easily be abused (e.g. by deleting other persons pages or by intentionally posting incorrect data) and loose much of their value. The feasibility of the Wiki approach thus depends on the culture in the adopting company.

If the users accept and embrace the Wiki as a form of knowledge sharing while working with a newly adopted software system, they can provide help to each other while they are working and making new experiences with the new system. Experts can use the Wiki to answer open questions and to provide manuals, tutorials and other documentation. The participatory approach used in Wikis enables all stakeholders to play an active part in the adoption process of a new software system. Ideally this can improve the identification with and acceptance of the new software.

4 Adoption of a Wiki in a company

The success of a Wiki depends on the participation of the users and therefore on a successful adoption of the Wiki software and the Wiki concept of collective content creation in a company. The adoption of a Wiki should in our view include the following measures:

- Training of all users in the usage of the Wiki system (text markup, page linking) and the Wiki philosophy (collective content ownership).
- Clear communication of the personal benefits of the Wiki for each user.
- Provide users with immediate initial benefits by supplying meaningful initial content and a useful initial structure with the Wiki. In this way users can learn how to create and structure content by looking at existing examples. They also immediately benefit from the provided content and are more willing to contribute to it.
- Listen to users' needs and suggestions. Regularly collect feedback about problems and experiences and improve the Wiki system and the organization.

After the introduction of a Wiki system, it is important to keep it alive and active and to maintain its value for the company. We propose the following measures to achieve this goal:

- Regularly discuss new content or changes in the structure with the users.
- Maintain the Wikis usefulness. If all users are allowed to add and modify content freely the structure of the Wiki web is likely to deteriorate over time and become inconsistent. A maintenance process can prevent this effect by employing activities such as regular deletion of obsolete pages, addition of links to pages, reorganization of information across multiple pages and improvement of the consistency of keywords.
- Provide incentives to active Wiki contributors to improve the motivation of users to participate.

These measures are aimed to achieve the participation of the users in the Wiki way of content creation. The successful adoption of a Wiki also depends heavily on the culture of a company (see section 3).

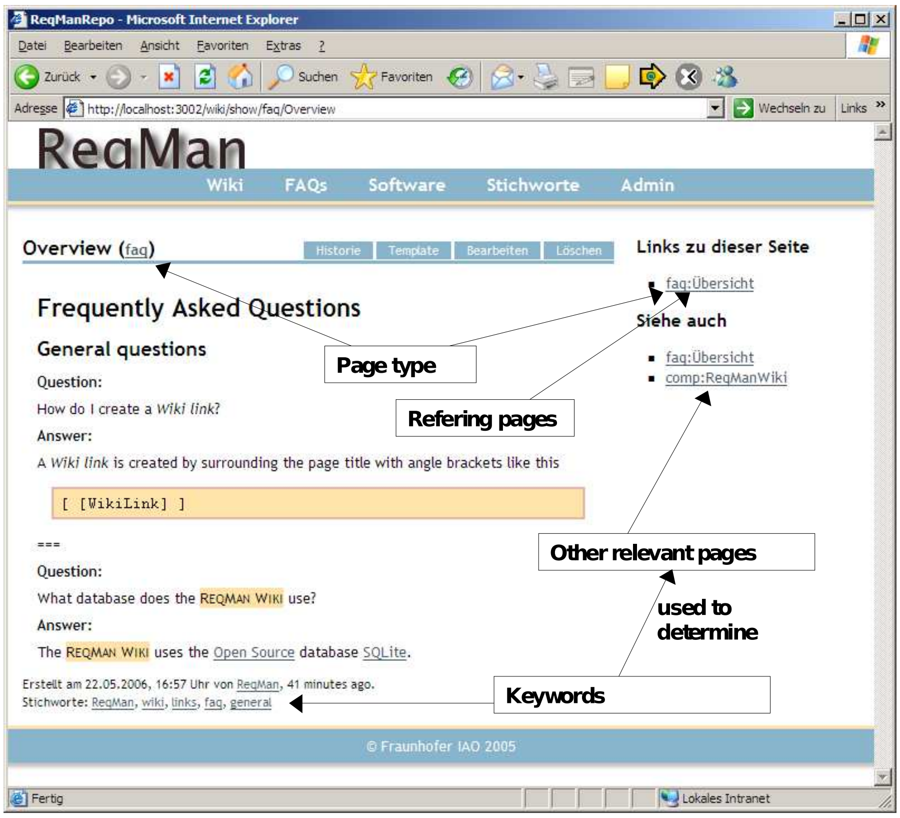
5 The ReqMan Wiki

In section 2 we described the basic Wiki functionality which is primarily suited for unstructured content. The structuring and formatting of content is completely in the responsibility of the user. In order to provide support for semi-structured information, which is produced during OSS adoption (see section 3), we have developed a Wiki that support different page types and content templates for arbitrary content structures.

In the ReqMan Wiki which was developed in the research project ‘ReqMan Requirements management for SME’ [2] each Wiki page has a type that can be chosen freely. Possible types can be for example “faq” or “experience-report” (see Figure 1). For each page type a content template can be defined. Each time a user creates a page of a certain type, the associated content template is provided as a blueprint that the user can fill in or chance. In this way a basic structuring of the content is suggested and encouraged without placing restrictions on the formatting of the Wiki pages.

In order to improve the structuring and navigation of the Wiki content the ReqMan Wiki also offers the possibility to provide tags (i.e. keywords) to attribute pages. These tags are used to identify other pages with related content (see Figure 2).

The ReqMan Wiki was originally developed to document best practices in requirements management and software reuse [9] but the open nature of a Wiki makes it equally suited for OSS adoption. It is realized in the programming language Ruby [10] with the Open Source web application framework RubyOnRails [11] and an SQLite [12] database (see Figure 3). It employs the established Model-View-Controller architecture and uses the object relational mapping infrastructure of RubyOnRails to decouple the application logic from the database.


ReqManRepo - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück Suchen Favoriten

Adresse http://localhost:3002/wiki/show/faq/Overview Wechseln zu Links

ReqMan

Wiki FAQs Software Stichworte Admin

Overview (faq) Historie Template Bearbeiten Löschen

Links zu dieser Seite

- faq:Übersicht

Siehe auch

- faq:Übersicht
- comp:ReqManWiki

Frequently Asked Questions

General questions

Question:

How do I create a *Wiki link*?

Answer:

A *Wiki link* is created by surrounding the page title with angle brackets like this

```
[ [WikiLink] ]
```

===

Question:

What database does the REQMAN WIKI use?

Answer:

The REQMAN WIKI uses the Open Source database SQLite.

Erstellt am 22.05.2006, 16:57 Uhr von ReqMan, 41 minutes ago.
Stichworte: ReqMan, wiki, links, faq, general

© Fraunhofer IAO 2005

Fertig Lokales Intranet

Fig. 2. Screenshot of the ReqMan Wiki

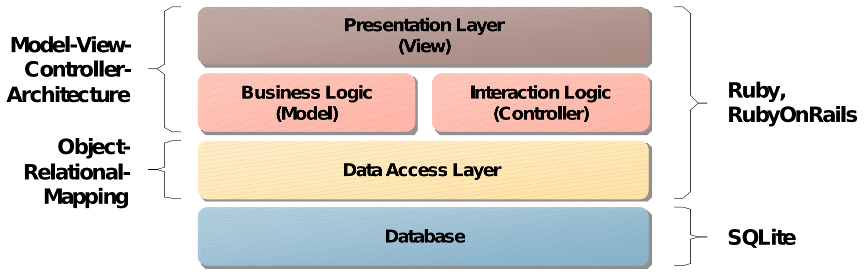

Model-View-Controller-Architecture
Object-Relational-Mapping
Presentation Layer (View)
Business Logic (Model)
Interaction Logic (Controller)
Data Access Layer
Database
Ruby, RubyOnRails
SQLite

Fig. 3. Architecture of the ReqMan-Wiki

6 Summary

In summary we described in this paper, how a Wiki can be utilized to capture, manage and disseminate knowledge and information that is produced during the adoption of an Open Source Software system. We further argue that the collaborative approach that is encouraged by a Wiki infrastructure supports the adoption and acceptance of OSS in organizations. This is especially true for agile organizations and SME. It is to be noted that Wikis can only be of value, if the mindset of collective content creation and knowledge sharing can be incorporated in the organizational culture of the company. We propose a set of measures for Wiki adoption and operation that help to create this mindset. The most important point here is to motivate the users to participate in collective content creation by clearly communicating the rules and the benefits of Wiki usage and to demonstrate these benefits with a working Wiki that already contains a basis of relevant content. We conclude with a short description of the ReqMan-Wiki. This Wiki engine offers three extensions to the basic Wiki functionality (namely page types, templates and tags) that provide support for semi-structured content with is needed for documenting many information that arise during OSS adoption.

7 Acknowledgements

We like to thank the German Federal Ministry of Education and Research for funding the 'ReqMan' project as part of its research initiative 'Software 2006'.

References

1. Leuf, Bo; Cunningham, Ward: The Wiki Way, Addison-Wesley 2001.
2. 'ReqMan Requirements Management for Small and Medium Enterprises', funded by the German Federal Ministry of Education and Research, `http://www.reqman.de/`.
3. Höß, Oliver: Erfolgreicher Einsatz von Open Source Produkten. In: Spath, D. ; Dauner, J. ; Höß, O. ; Wönsche, V.: Strategisches Open Source Symposium: Die Entscheider Konferenz, IAO Stuttgart, 2004
4. Ward Cunningham's original Wiki is accessible under `http://c2.com/cgi/wiki` , last accessed on May 18^{th} 2006.
5. Wikipedia - the free encyclopedia, `http://www.wikipedia.org/`, last accessed on May 18^{th} 2006.
6. Allen, Dean: Textile A humane Web text generator, `http://textism.com/tools/textile/`, last accessed on May 18^{th} 2006.
7. Gruber, John; Swartz, Aaron: Markdown, `http://daringfireball.net/projects/markdown/`, last accessed on May 18^{th} 2006.
8. Wiki-Matrix: A feature comparison of over 30 Wiki engines, `http://www.wikimatrix.org/`, last accessed on May 18^{th} 2006.

9. Strauß, Oliver; Falkner, Jürgen.; Höß, Oliver; Weisbecker, Anette: A lightweight framework for software reuse in small and medium sized companies. In: Proceedings of the 18th international conference on Software & systems engineering and their applications ICSSEA 2005, Vol. 1, CNAM Paris, 2005
10. Thomas, David; Fowler, Chad; Hunt, Andy: Programming Ruby 2nd Edition, The Pragmatic Bookshelf, USA, 2005
11. Thomas, David; Heinemeier Hansson, David: Agile Web Development with Rails, The Pragmatic Bookshelf, USA, 2005
12. SQlite Project Homepage, `http://www.sqlite.org/`, last accessed on May 21th 2006.

Challenges and obstacles: Usage of Free and Open Source Software in local government in Macedonia

Bardhyl Jashari and Filip Stojanovski

Metamorphosis Foundation
"Naum Naumovski Borche" 88-a, Skopje, 1000, Macedonia,
info@metamorphosis.org.mk
WWW home page: http://www.metamorphosis.org.mk

Summary. Survey of 14 municipalities in Macedonia highlights the challenges and obstacles to usage of Free and Open Source Software (FLOSS) within local government units. Municipalities provided answers related to their software usage and budget plans, clearly indicating the lack of funds to satisfy the needs of the citizens they serve by purchasing proprietary software available from vendors associated with the central government. TOC of using FLOSS compared to proprietary software in Macedonia is comparatively lower, ranging from one-third to one-half of the price needed for purchase of the software and training IT staff to provide maintenance and support for it.

1 Introduction

Local governance in Macedonia is organized into 85 local government (LG) units: 84 municipalities, and the City of Skopje, a special LG unit encompassing and sharing certain competences with the 10 municipalities comprising the capital. The new Law on Decentralization modeled according to the EU standards and directives took effect in July 2005, transferring powers and responsibilities from central to local level.

This study used a representative sample of 20 LG units (24%), out of which 14 (16%) provided information through the standardized questionnaire, under condition that individual municipality data would not be disclosed. Moreover, the LG units covered by the study (Fig.1) provide a cross-section of the various kinds of municipalities present in the country, with diverse geographical characteristics, ethnic and religious background, political affiliation, and level of economic development.

In regard to level of ICT implementation, the sampled municipalities include some of the leaders in this field (Veles, City of Skopje), and most of

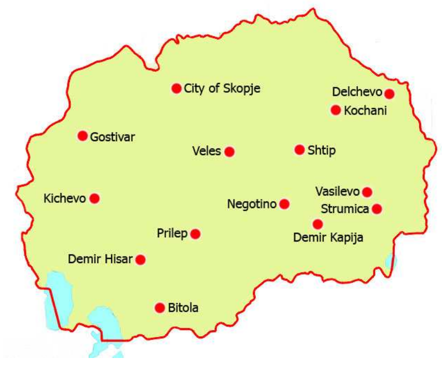


Fig. 1. Macedonian municipalities covered by this study

them have shown higher than average level of interest for further IT development, expressed through commitments and participation in various projects, such as the Project for development of citizen-oriented e-Government services on a local level run by USAID and Metamorphosis Foundation. This research does not cover municipalities which have not expressed interest in their IT development.

The main ICT policy document in Macedonia is the National Strategy for Information Society Development (NSISD), adopted in 2005 by both executive and legislative branches of the government, which is technology-neutral and brand-free.

The e-Citizens section of the NSISD defines the LG units as primary institution for enabling G2C communication, and states (section 2.6.1 of NSISD) that it "is necessary to support the use of the FLOSS as a public good accessible to all citizens, especially through the freedom of choice of a platform in the formal and informal ICT education." One of the objectives (06.03) of the same section also requires enabling "accessible e-services irrespective of the software platform, which shall use open standard formats of documentation and exchange of information."

1.1 Known Results

The UNDP survey of 2004 identified low level of IT capacity within Macedonian municipalities, especially pointing out the lack of qualified personnel as major obstacle, but also lack of appropriate equipment.

In 2003 the Government of Republic of Macedonia signed a $4 million Strategic partnership contract with Microsoft, licensing all unlicensed Microsoft software used by the central government and public educational institutions, with Microsoft agreeing to provide localized versions of its OS and office suite, and its local partners receiving contracts to develop several high-profile of e-Government applications. This in effect sealed the almost exclusive use of Microsoft-based products within the central government, to the point that even free, closed source applications for submitting reports to Pension Fund are provided for Windows only, without alternatives.

Use of FLOSS within central government remains sporadic and is mainly limited to web servers and web applications such as CMS (`http://www.finance.gov.mk`), resulting from actions of individual IT experts, not from wide-ranging policy.

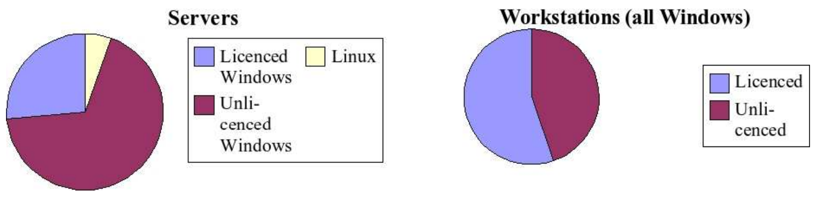


Fig. 2. Operating systems used within surveyed municipalities

LG units were not included in the MS licensing scheme, and have been largely left to their own devices to acquire IT equipment and licenses. Donations by international organizations, such as UNDP and, USAID, some done in cooperation with Microsoft, remain key source in this area.

1.2 Our Results

According to our research, **lack of support** for OSS is emphasized as one of the most important obstacle for OSS widespread use in public administration. Most of the LG units do not have their own IT person. Using the services of local IT companies to provide support for OSS, could foster the development of the local OSS community but also of the IT sector in general.

Additional reason could be the unavailability of open source applications that include all of the functionality of the proprietary software applications they would replace. Especially GIS application and specific tailor-made applications (ex. Finance software).

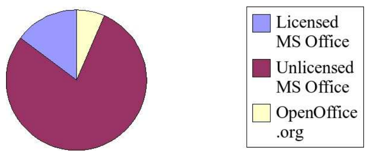


Fig. 3. Office Software used by surveyed municipalities

The second problem is **compatibility** with past, mostly proprietary, software application. Local governments have thousands of documents created with various versions of Microsoft and similar applications over the years. OSS application still need to be able to open those documents and read them. Perhaps they also need to be maintained so that people must have the possibility to edit them.

Another issue is **training**. Although most OSS application user-interface and functionalities are similar to the Microsoft (or other) applications they're replacing, there are still a lot of differences that hamper the usage of OSS. Training is needed in order to overcome this obstacle and for new users to become more familiar with the new working environment. Most of the LG units are willing to migrate to OSS if they would be provided appropriate training and support.

2 Preliminaries

The findings of the survey in more detail comprise of accumulated responses to the standardized questionnaire, filled in by the persons responsible for IT within the responding LG units.

Table 1. Operating systems used by the municipalities

Number of PCs with	Licensed OS		Unlicensed OS	
Windows Server (2003, 2000, NT)	**5**	2,4%	**13**	7,4%
Windows (XP, 2000, 98, ...)	**200**	95,7%	**163**	92,6%
Linux (servers)	**1**	0,5%	**0**	0,0%
Linux (desktop computers)	**0**	0,0%	**0**	0,0%
Other	**3**	1,4%	**0**	0,0%

Table 2. Office software used by the municipalities

Number of PCs with	Licensed software		Unlicensed software	
Microsoft Office (XP, 2003, 97)	**51**	53,1%	**271**	67,9%
OpenOffice.org, other FLOSS	**20**	20,8%	**2**	0,5%
Other commercial software	**25**	26,0%	**126**	31,6%

Most of the repsodents expessed reluctance about readiness to **migrate** to FLOSS, with 64,3% replying "maybe" when asked whether they are willing to migrate. On the other hand, the number of respodents who do not consider such an option (14,3%) is almost two times smaller than those who are willing to migrate to FLOSS (21,4%).

Table 3. Reasons for not using FLOSS

Lack of technical support	**6**	33,3%
Lack of sufficient information	**3**	16,7%
Lack of knowledge of implementation and maintenance	**6**	33,3%
Other: lack of training, insufficient FLOSS affirmation.	**3**	16,7%

Table 4. Conditions for migrating to FLOSS

Availability of appropriate training and support.	**8**	44,4%
Availability of local vendor and/or consultant.	**2**	11,1%
Lowering the total cost of ownership.	**7**	38,9%
In any case.	**0**	0,0%
In case of compatibility with current software solutions.	**1**	5,6%

Most of the respodents (78,6%) are aware of the existence of FLOSS office applications localised in local languages (Macedonian, Albanian, Turkish, etc.), while the reamianing 3 (21,4%) are not.

Most municipalities express interest in using GIS software, with 35,7% using the ESRI Arv View, 28,6% planning to purchase such software soon, and 35,7% not using GIS software.

Table 5. Other software used by municipalities

Accounting & budget (proprietary, locally developed)	**7**	41,2%
Lotus Notes-based archive software	**3**	17,6%
None	**3**	17,6%
Document management system	**1**	5,9%
SW for e-submission of application in Pension Fund	**1**	5,9%
Archive, issuing and approving building permissions (implementation phase)	**1**	5,9%
LTAS – taxes and communal taxes (test phase)	**1**	5,9%

Comparing the current market prices for Microsoft software licenses, the licensing of the unlicensed OS and basic office software they currently use would cost the surveyed municipalities up to EUR 109,265 (OS + Office SBE) or EUR 180,636 (OS + Office Prof.).

The total budget for software purchasing deployed by the surveyed municipalities is EUR 45738, less than half of the cheaper option. Moreover, 50% of all the municipalities plan no budget for this purpose.

Only four of the surveyed municipalities (28,6%) have empolyeed system engineer. Of the remaining 71,4%, most plan to employ one in near future

Table 6. Kinds of software the municipality needs and plans to purchase in the near future, ranked according to the priority

	Ranking
Document management system (DMS) and document digitisation	**1**
Security software (firewall, network antivirus)	**2**
Financial software	**3**
Front/back office solution for forms by citizens and businesses	**4**
Backup software	**5**
Information management system	**6**
Workflow management (Workflow, BPM)	**7**
Intranet groupware and project management solutions	**8**
Client relations management (CRM)	**9**
Resource management and planning (lgRP)	**10**
Other software: Microsoft Small Business Server (ISA, Exchange..), GIS software (ArcView), ect.	**11**

Table 7. Annual municipal budget designated for software purchase

None	**4**
No reply, does not know	**3**
EUR 820	**1**
EUR 984	**1**
EUR 2459	**2**
EUR 2951	**1**
EUR 3279	**1**
EUR 32787	**1**

Table 8. Usual prices of licenses for Microsoft software in Macedonia

Windows Server	**EUR 836**
Windows XP	**EUR 159**
Office Small Business Edition	**EUR 268**
MS Office Proffesional	**EUR 531**

(70%), and others either do not plan such action (20%) or provided no reply (10%).

Very few of the surveyed municipalities (2 out of 13, or 14,3%) have local strategy and action plan for implementation of ICT development in the municipality. Majority of 71,4% does not have such policy documents developed, but would appreciate additional information and consultations about the opportunity for strategic planning of the local ICT development in the

Table 9. Annual municipal budget for information system maintenance

	External consultant (Outsourcing)	In-house (municipal employee)	No reply
None		**2**	
No reply		**3**	**1**
EUR 820	**1**		
EUR 984	**2**		
EUR 1475	**1**	**1**	
EUR 1639	**1**		
EUR 3279	**1**		
EUR 29508		**1**	
Total:	**6**	**7**	**1**

municipality. The remaining 14,3% does not have a strategy nor plans ICT policy activities in near future.

Prices of training of IT professionals in Macedonia range from EUR 2,079 for each of the Microsoft-based MCSA and MCSE certifications, to EUR 1,000 for a standard course for Red Hat administrators. Taking into consideration the licence prices and the training prices, it is obvious that satisfying the stated needs of the surveyed municipalities will be more effective by using FLOSS.

References

1. Gusev, M. (2004). Local e-Governance, ICT Assesment Report of municipalities in Macedonia. UNDP, Skopje, Macedonia. `http://www.undp.org.mk/e-governance/`
2. The National Strategy for Information Society Development (2005), and abridged version of the Strategic Partnership Agreement between the Government of the Republic of Macedonia and Microsoft Corporation (2003) can both be downloaded from the website of the Committee for IT of the Government of RoM, `http://www.kit.gov.mk`
3. Kostadinova-Daskalovska, K. (2005). Usage of Information and Communication Technologies, 2004: Results of the pilot survey. State Statistical Office of the Republic of Macedonia, Skopje, Macedonia
4. (2004). General Data About the Situation Regarding the ICT in Macedonia. Metamorphosis Foundation, Skopje, Macedonia, available on the website of the publisher `http://www.metamorphosis.org.mk/` in the Research section.

A F/OSS Usability Study: tOSSad Case

Damiano Verzulli[1], Uros Jovanovic[2], and Görkem Çetin[3]

[1] PDA Communication Srl, Italy
damiano@verzulli.it
[2] XLAB d.o.o
uros.jovanovic@xlab.si
[3] The Scientific & Technological Research Council, Turkey
gorkem@gorkemcetin.com

Summary. The EU funded project tOSSad (Towards Open Source Software Adoption and Dissemination) project has an aim to improve the outcomes of the Free / Open Source Software (F/OSS) communities throughout Europe. One of tOSSad's mission is to share usability aspects and requirements among usability experts in order to help developers complete their F/OSS projects on time and under budget by producing howtos, tutorials, guidelines and carrying out F/OSS usability tests. This work tries to solve some of the problems F/OSS faces, by conducting a survey and analysing the outcoming results. Within this scope, tOSSad project partners first designed an online survey with a focus on three major F/OSS component, namely OpenOffice.org, Linux and Firefox. We then analysed the data collected and as a result, majority of the respondents admitted that Linux, OpenOffice.org Firefox have enough features and that they are user-friendly. In other words, they have got a high assessment of usability. However, it should be pointed out that, Linux desktop should advance so that it can be properly used on the desktop by the newcomers.

1 Introduction

Usability is a term describing the effectiveness, efficiency and satisfaction [1] with which users can achieve tasks. Today, a large number of developed software does not meet users' needs or does not perform as expected. In order to address this problem, users must be involved in all stages of the development process - a user-centred design (UCD).

Usability is still not the first priority of open source developers. Developers working on the same project are usually geographically distributed and culturally diversed, therefore having different priorities and aesthetic demands. For this reason much open source software, and particularly the desktop environments, has been claimed to be unusable, not giving satisfaction of using, presenting a steep learning curve and an overall low usability. The developers had mainly produced software for high-end technical people, with a focus on attributes and features. It seems that at present the very nature of open and distributed development results in a lack of implementing user-centred design (UCD) process for the software project managers.

2 Previous works

In order to provide an overview of the history and the field at present, this research on F/OSS usability studies started with an analysis of projects and initiatives taken in the past dealing with the most current issues in the usability field. Also, the concept of "extreme usability", a methodology that incorporates usability in a highly iterative and agile development process, has been reviewed.

There are a vast number of initiatives, projects and articles about open source and usability. Below we have listed the most significant initiatives currently registered at a worldwide scale. We have tried to find the ones that deal with the most current issues in the field today.

The quality and quantity of F/OSS usability projects have been dramatically increasing since 2000. Among the projects that has considerably helped F/OSS to be adopted by end users are Better Desktop [2], Extreme Usability [3], GNOME Usability Project [4] and KDE Usability Project [5].

Generally the attention for usability in F/OSS has increased over the past years and thanks to this, much progress has been made. Firefox is a good example were an open source application successfully competes with proprietary software, though it requires effort from the F/OSS community to face the challenges that still exist.

3 Methodology

In order to collect empirical information about F/OSS applications a survey was performed. The objective of the survey was to collect user opinions about the usability of Linux desktop as a desktop operating system, Firefox and OpenOffice.org. These products were chosen since they represent three categories of products and are among the most well known F/OSS products on the market today.

The survey performed was self-administered. The most common ways of distributing self-administrated surveys are through the use of mail, fax, newspapers/magazines, and the Internet. The questionnaire, conducted under the activities planned within work package 3 of the tOSSad [6] project, was made available on the tOSSad web site and performed on-line.

The planning and preparation of the survey took place during telephone-meetings and mail communication between partners. The first issue was to decide for the applications that were to be handled in the survey. Consensus about Linux, OpenOffice.org and Firefox was established almost instantly. It was also decided that each section would consist of about 15 questions, hence not exceeding 15 minutes for a respondent to finish the survey. The design of the questions was divided between Viewrope (Linux), University of Stuttgart (OpenOffice.org) and Internet Society Bulgaria (Firefox). A first draft was rapidly created and circulated among the partners, based on the comments a second draft was created. After a number of iterations and discussions, the final version came to consist of 17 questions about Linux, 15 questions about Firefox and 18 questions about OpenOffice.org.

4 Analysis and results

The survey results have been analysed using eZ publish survey tool and more than 20 figures and charts have been produced. Detailed results have been included in the tOSSad deliverable D10, available on-line on the tOSSad website. Below is a shortened form of each analysis results.

4.1 Results of the Openoffice.org survey

In this section of the survey, 196 respondents took part. The number of respondents for each survey section is different because the usage of OpenOffice.org and Mozilla Firefox is not limited to the operating system. Therefore, the participants in the survey could have experience with both products but not with the Linux OS.

OpenOffice.org survey results showed that 67% of the respondents have a big working experience and habits built with Microsoft Office. Thus we can consider that their answers compare the usability of these two products in a correct way.

Moreover 36% and 34% of the respondents admitted high usability and sufficient number of features, which is an indication that the product is being developed in the right direction regarding making it more user friendly. However, the fact the the percentage of such answers is still far below the 50% means that more work has to be done. Nevertheless, 66% of the users admitted that they have got a positive impression of OpenOffice.org and another 86% considered the user interface and functionality to be good.

An interesting fact that appears is that 10% use it because it is their company policy, 7% more declare that they did not switch from MS Office and 8% started with OpenOffice.org (they have not used other similar product). This leads to the conclusion, that a major increase in such type of users and companies can be expected in the next few years due to the appearance of new releases that incorporate more features and improve usability.

As a final comment, a really minor percentage (4%) declare to face difficulties with the installation process which can be considered a big success towards making the product more user friendly

4.2 Results of the Firefox survey

205 respondents took part in the Firefox survey. Mostly “old” Firefox users from half year and more (83%) took part in the survey. From the last 6 months only 16% of the respondents became new Firefox users. Its nearly two times less then in OpenOffice.org. That is why information promotion is essential for the Firefox but again, like in OpenOffice.org it can also be concluded that a critical mass of users is reached.

The main reasons for switching to Firefox are that it is free and the product has some additional features such as tabbed browsing, which is used and liked by more than 90% of users. Stability, user-friendly interface and security are also estimated as positive items of the Firefox usability.

In terms of usability, 97% of respondent found in Firefox a sufficient amount of functionalities and for another 97% the installation process was wasy: two important factors in Firefox popularity.

In general 95% of the respondents gave a positive assessment of the usability aspects of Firefox.

4.3 Results of the Linux survey

The Linux survey was mainly directed to research the Desktop usage and usability issues. In terms of user interface, even if GNOME or KDE are the default choice for lot of distributions, due to the flexible nature of any Linux systems it’s easy, for the user, not only to adopt a third, different, user interface but also to continuously switch between them. Hence, we decided to not “stick” on a certain interface but, on the contrary, to prepare a set of “general” questions.

274 respondents took part in the Linux survey. The majority of Linux users (90%) have been using this operating system for more than half a year and only 9% of users switched to Linux in the last six months. From this appears the conclusion that most of the respondents are well experienced users of Linux that have passed the initial adaptation period of working with a new OS and can objectively evaluate overall usability.

The survey has shown that SUSE/Novell, Debian GNU/Linux, Gentoo and Ubuntu are the most popular among Linux distributions, quickly followed by Mandrake/Mandriva and RedHat/Fedora.

For some users the first start with Linux was not so easy (37%). This indicates that the developers should put more emphasis on a good “Start up help” and documentation. Although this is a very important area of program development and should by no means be neglected, it is not F/OSS specific, the same rules apply to proprietary programs. However, in our opinion, it is more likely that someone (an advanced user) writes a tutorial for a free program than for the program owned by another company.

In contrast to the common opinion that it is difficult to install the Operating System and hardware under Linux, for the majority of the respondents

(64%) this is not that difficult. This means that one of the important gaps of Linux desktop usability is being worked on.

As it was marked by the majority of the respondents, Linux operating system is a stable (88%) and fast (76%) system, which is important for usability. In general 95% of the respondents evaluate Linux highly.

5 Conclusion

In this research, we have tried to identify main problems around usability issues of F/OSS, with possible mitigation and recruitment ideas derived from the survey responses. The online survey of Linux, OpenOffice.org and Firefox shows that, after the last few years of usability improvements of F/OSS products, many users have come to the conclusion that all three products have a user friendly interface. While “Linux desktop” is a broad metaphor which can be pulled elsewhere and perceived differently, we see a consensus among the survey respondents that Linux desktop is easy to use with clear abbreviations, user-friendly interfaces and logical navigation. However there is room for improvement regarding hardware installation and a reduction in the learning curve needed in the process of migration from the MS Windows interface.

6 Survey Questions

6.1 Linux Questions

1. **What Linux distribution do you use?**
   (Mandriva (formerly known as Mandrake) - SuSe/Novell - Debian GNU/-Linux - Gentoo - TurboLinux - Slackware - Knoppix - Red Hat Enterprise Linux - Fedora Core - Ubuntu - Pardus - Other)
2. **Would you recommend your version of Linux to your friends and colleagues?**
   (Yes No)
3. **How long have you been using Linux?**
   (less than 6 months - 6 to 12 months - more than 12 months)
4. **Was it easy to start using Linux?**
   (1 very difficult, 6 very easy)
5. **Is the way that Linux presents information clear and understandable?**
   (1 very unclear, 6 very clear)
6. Is it easy to find information and instructions for tasks that you want to perform? (1 very difficult, 6 very easy)
7. **Is the navigation logical when performing a task?**
   (1 very illogical, 6 very logical)

8. **Is your operating system fast?**
   (1 very slow, 6 very fast)
9. **Is it easy to install hardware under Linux on your computer?**
   (1 very difficult, 6 very easy)
10. **Are the error messages clear and informative**
    (1 very uninformative, 6 very informative)
11. **Is it easy to install/uninstall software under Linux on your computer?**
    (1 very difficult, 6 very easy)
12. **Is your Linux system stable?**
    (1 very unstable, 6 very stable)
13. **How would you rate the performance of the Linux operating system?**
    (1 very good, 6 very bad)
14. **What is your overall impression of the Linux operating system?**
    (1 very negative, 6 very positive)
15. **What do you like best about your Linux distribution?**
16. **What do you dislike most about your Linux distribution?**
17. **What could, in your opinion, be done to make your Linux distribution more user-friendly?**

6.2 OpenOffice.org Questions

1. **Which OpenOffice.org version do you use?**
   (My version is older than 1.0 - OpenOffice.org 1.0.X - OpenOffice.org 1.1.X - OpenOffice.org 2.0 alpha/beta - Derivatives and ports like NeoOffice, StarOffice, etc - I don't know / Not applicable)
2. **How long have you been using OpenOffice.org?**
   (less than 3 months - 3 to 6 months - 6 months to 1 year - more than 1 year)
3. **Why did you switch to OpenOffice.org?**
   (I started with OpenOffice.org - Because security is better due to general - availability of source code - It has lots of features - It is user-friendly - It does not crash - It is free - I like the F/OSS (free/open source software) idea - I think the quality of F/OSS is better - F/OSS is more open to standards than commercial software - There are no constraints due to legal commitments (licenses) - I use both Microsoft Office and OpenOffice.org - OpenOffice.org is our company/organisation policy - I did not switch from Microsoft Office - Other)
4. **If you did not switch from Microsoft Office, please specify from which application you switched.**
5. **If you have other reasons for switching to OpenOffice.org, please specify them here.**
6. **What are your skills regarding Microsoft Office**
   (1 - unskilled, 6 - very skilled)?

7. **What is your overall impression of OpenOffice.org**
   (1 - very bad, 6 - very good)?
8. **Are the shortcuts good enough in order to use OpenOffice.org with the keyboard only?**
   (1 - very bad, 6 - very good)
9. **Rate the quality of OpenOffice.org translation in your language**
   (1 - very bad, 6 - very good)
10. **Please select the language of OpenOffice.org you use.**
    - A list of languages was given in the survey and omitted here due to space considerations

11-18 **Please rate the following aspects of OpenOffice.org:**
- Ease of use (1 - very complicated, 6 - very easy to use)
- Comprehensibility of terms, names, abbreviations or symbols in forms / menus, etc. (1 - completely incomprehensible, 6 - easy to understand)
- User interface (1 bad interface, 6 - excellent interface)
- Functionality, provided by OpenOffice.org (1 bad/lacking functionality, 6 good/satisfying functionality)
- Stability in comparison with Microsoft Office (1 completely unstable, 6 very stable)
- Ease of installation (1 - very difficult to install, 6 - very easy to install)
- Overall performance (1 - very bad performance, 6 - excellent performance)
- Features of functions you know from Microsoft Office of other products (1 lots of missing; 6 none missing)

6.3 Firefox Questions

1. **How long have you been using Firefox?**
   (Less than 3 months - 3 to 6 months - 6 months to 1 year - More than 1 year)
2. **Why have you switched to Firefox?**
   (It is more secure - It has lots of features - It is user-friendly - It does not crash - It is free - It has tabbed browsing - It is the best option on my operating system Other)
3. **I use the tabbed browser option of Firefox**
   (Yes No - Don't know / Do not use it)
4. **I like tabbed browsing.**
   (Yes No - What are they?)
5. **Rate the quality of Firefox translation in your language**
   (1 - very low quality, 6 excellent quality)
6. **Please select your language:**
   - A list of languages was given in the survey and omitted here due to space considerations

7. **Are the shortcuts good enough in order to use Firefox with the keyboard only?**
   Please rate them from 1 (worst) to 6 (best)
8. 8-15 Please rate the following aspects of Firefox:

   — Ease of use (1 is very complicated, 6 is very easy to use)
   — Comprehensibility of terms, names, abbreviations or symbols in forms / menus, etc. (1 is completely incomprehensible, 6 is easy to understand)
   — If applicable, provide an example for the previous question
   — Graphical user interface in comparison with that of Internet Explorer (1 is bad interface, 6 is excellent interface)
   — Functionality, provided by Firefox (1 is bad/lacking functionality, 6 is complete/satisfying functionality)
   — Stability in comparison with Internet Explorer (1 is completely unstable, 6 is very stable)
   — Ease of installation (1 is very difficult to install, 6 is very easy)
   — Overall performance of Firefox (1 is very bad performance, 6 is excellent performance)

References

1. International Standards Organization (ISO) 9241-11
2. Better Desktop Project, `http://www.betterdesktop.org`
3. FLOSS Usability, `http://www.flossusability.org`
4. GNOME Usability Project, `http://www.developer.gnome.org/projects/gup/hig`
5. KDE Usability Project, `http://usability.kde.org`
6. tOSSad project, `http://www.tossad.org`

Integrating Usability with Open Source Software Development: Case Studies from the Initiative OpenUsability

Ellen Reitmayr[1], Björn Balazs[2], and Jan Mühlig[3]

[1] relevantive AG, Saarbrücker Str. 38, 10405 Berlin
ellen.reitmayr@relevantive.de
WWW home page: www.relevantive.de

[2] Apliki GbR, Marktstr. 7, 10317 Berlin
bjoern.balazs@apliki.de
WWW homepage: www.apliki.de

[3] relevantive AG, Saarbrücker Str. 38, 10405 Berlin
jan.muehlig@relevantive.de
WWW home page: www.relevantive.de

Summary. When getting involved with Open Source Software (OSS) projects, usability specialists frequently face conditions that make their work difficult. While most of these conditions are known from commercial software development, the combination with certain organisational structures found in OSS projects pose new challenges. In this paper, factors influencing usability work in Open Source Software projects as experienced by members of the initiative OpenUsability[4] are summarised and illustrated in case studies. The sketch of an integrated user-oriented design process for OSS is meant to facilitate future collaboration.

1 Introduction

Up to today, professional usability support and defined User Interface Design processes are rare in Open Source Software (OSS) projects. Even if usability becomes more and more of an issue, little knowledge exists among the developers about the scope, the methodology and the application of an iteratively optimising User-Oriented Design process in terms of ISO 13407 (ISO, 1999). Usability is often perceived as "yet another feature" which can easily be implemented as an add-on to the next software release.

The initiative OpenUsability[5] aims to overcome those assumptions by bringing Open Source developers and usability experts together. The idea

[4] www.openusability.org
[5] www.openusability.org

is to establish a long-term relationship between usability and OSS development, and to foster the establishment of User-Oriented Design processes in terms of ISO 13407 (ISO, 1999).

In the following, factors influencing usability work in Open Source Software projects as experienced by members of the initiative OpenUsability during its three years of activity are presented. In case studies, those factors are set into relation with successful and less successful collaboration between usability and Open Source, and practical suggestions are given how to overcome limitations. Finally, the model of an integrated development and usability process for Open Source development cycles is sketched.

2 Factors Influencing Usability Work in OSS Projects

2.1 Target Users and User Feedback Mechanisms

There are diverging theories about the ecology of Open Source Software projects and how people get involved with it. One of the most well-known and controversially discussed is "The Cathedral and the Bazaar" by Eric S. Raymond (1998) which suggests that individuals start projects to "scratch an itch". Later research has shown that while the actual reasons are more complex, demanding a tool oneself and gaining reputation among fellow developers (Lerner and Tirole, 2000) were important factors to get involved with OSS development.

With regard to usability work in OSS projects, it has shown that such projects often face problems when the user base is extended outside the original community. User innovation cycles[6] as described by Hippel (2001) often do not apply any more. Developers tend to disregard feedback from "common users" in favour of source-aware user comments (Yeats, 2006). As the latter is usually given by fellow developers, little may be known about the user base outside the own community.

But even if projects are explicitly meant to serve a more extended user base, wishes of "common users" as identified in forums or wish lists are often randomly added or neglected, and hastily judged by means of technical measures (Yeats, 2006). Aggregated or directed user feedback mechanisms usually are not existing.

The advantages of using aggregated feedback channels, in contrast, are illustrated in the case study "TV-Browser". All in all, the OpenUsability work has shown that a clear definition of the target users, as well as an understanding of the users' tasks and requirements is missing in the majority of OSS projects.

[6] User innovation cycles are for example merely accountable for the success of the Apache server over proprietary competitors.

2.2 Feature- versus User-Orientation

As a consequence of an insufficient understanding of the user base, a traditional focus on feature- rather than user-oriented design is often observed in OSS projects. Implications and consequences are below illustrated by the example of the image manipulation application "GIMP".

2.3 Hierarchical Structure

In their investigation of the ecology of OSS projects Healy and Schussman (2003) suggest that the success of Open Source Software and hierarchical structure within the project are intercorrelated: Successful projects in terms of commit or download frequency showed to have a stronger hierarchical organisation.

In usability work in the scope of OpenUsability, the hierarchical factor has also shown to be relevant: Missing hierarchical decision paths pose a problem in larger projects. Here a usability expert needs to convince each relevant developer instead of a single project "leader" about the fact that technical expansion and interaction design should go hand in hand. A "leader" in terms of Healy and Schussman (2003) is a key developer who is respected and followed by the rest of the project.

As there is typically a lack of manpower on the usability side, the establishment of a continuous User-Oriented Design process is hindered significantly in projects with extremely flat hierarchies, or where the "leader" is not the primary contact of the usability specialist. In projects with a clearly defined "leading" contact instead, it is also easier for usability experts to keep track of the activity and current developments in the project. The effects of hierarchical structures and finding influential project members to collaborate on usability is pictured in the case study "KDE Project".

3 Case Studies from the Initiative OpenUsability

3.1 KDE Project

Compared to the majority of Open Source Software projects, the Linux desktop environment KDE has an exceptional position: Rather known from the development of frameworks, it is made up of a core development base, but additionally comprises a high number of smaller, "free-floating" software projects. A consistent hierarchical structure can not be observed - while some KDE projects established a strong hierarchy with strict regimentation for code submissions, others are less structured.

When OpenUsability got involved with KDE, the usability experts started to collaborate with the projects Kontact, Kivio, DigiKam and Kuroo. Here, influential project members ("leaders" in terms of Healy and Schussman, 2003) had explicitly requested usability input. Suggestions were mostly fruitfully discussed and implemented.

In the project AmaroK instead, a single developer had asked for usability input, obviously without telling others about it. While a usability specialist soon did an analysis, the AmaroK core developers were embarrassed about criticising their usability, and claimed they had never asked for a report - it was neglected.

Analysing the conditions leading to that disappointing reaction, one aspect showed to be the missing verbal agreement with the project "leaders". The other, equally important aspect, was that the usability specialist did not consult the project about its goals and current problems before doing the analysis: While the report followed general guidelines, AmaroK in those days had a special concept of what their software should be like. Identifying those project goals and working in a directed way towards those goals might have been a better start than a general usability report. The strategy of identifying goals has proved to be very effective - successful examples in KDE are the Gentoo package manager Kuroo or the flowcharting application Kivio.

Still, such punctual and rather general reports are the "cheapest" way to provide usability feedback to OSS projects. To the usability specialist, they offer the opportunity to invest a calculable amount of time in doing the analysis and creating the report. Considering some extra time for discussion, the usability feedback is "done". In practice it has shown that developers set different priorities in fixing issues than usability experts: Instead of seeing them in the context, they implement the "quick fixes" first even if severity and priority values are provided in the report. In Kontact, the KDE Personal Information Manager, this lead to an inconsistent menu structure among mail, calendar and address book as suggestions for one part of the application were implemented, but not for the other parts. All in all it has shown that writing punctual reports along common usability heuristics resulted in a number of quick fixes, but that major issues regarding interaction design or information architecture were often neglected. Still, punctual reports following general guidelines are preferable to no usability feedback at all.

With the growing involvement of usability specialists in single KDE applications, the need for general guidelines and a common direction for the whole KDE desktop became apparent. Here, the usability specialists faced a new challenge: It was difficult to determine a clear structure and obvious "leaders" in the KDE core development team. The distribution of responsibilities significantly hindered (and still hinders) the establishment of a consistent user experience of the KDE desktop. Current efforts concentrate on finding a vision and target user base that is accepted by the majority of the project possible strategies are voting, discussions and/or implementing prove of concepts in hierarchically structured sub projects. The effectiveness of the suggested methods still needs to be proven.

Lessons learned from the KDE project:

1. Check if the project is willing and has the time for a *long-term collaboration* with usability experts.
2. Identify *"leaders"* and compile project goals with them.
3. If no "leader" can be detected, analyse *common goals* in the project, and manifest them in a vision that is shared by the whole project.
4. Apply usability methods *appropriate* to *achieve these goals.*
5. Report results and suggestions *early and often.*

3.2 TV-Browser

TV-Browser is a java-based TV guide which is designed to look like a paper TV guide. While the key developers had first asked for usability support on their settings dialog only, the usability specialist soon realised that there were inadequate theories in the development team regarding the users' expectations and needs in terms of the whole application. Those theories had probably evolved from insufficient user feedback mechanisms and from not considering users outside the community.

In order to clear up the misconceptions and answer questions relevant for the design of the application, the usability specialist performed an online user survey. The most important result of the survey could be observed when following the ongoing discussion on the projects forum during the first hours the survey was publicly announced: While developers were at first downplaying the problems described by single users, the growing number of participants made them understand that the majority of users shared the same problems and expectations. Aggregating directed feedback from a representative number of answers here gave the developers a reality check and strongly enforced the future collaboration.

Lessons learned from TV-Browser:

1. Identify (diverging) *theories about user habits* and *expectations* among the developers.
2. Do a *survey or user observations* to learn about the actual habits and expectations of the users, and by this establish an understanding of the user base outside the own community.

3.3 GIMP

The GIMP is a freely distributed program for such tasks as photo retouching, image composition and image authoring and has for a long time been the only valuable image manipulation tool for Linux. Due to this status, its user base is widespread, and tasks accomplished with the tool range from simply flipping an image to professional arts production.

In the past, this widespread user base had lead to an uncontrolled growth of the application, especially with regard to filters and script extensions. While the functionality grew, the information architecture was not adjusted to the new demands, and finding relevant functions became more and more difficult even for expert users. The feature-oriented design style actually complicated the usage.

In order to facilitate the switch to a user-oriented development style, the GIMP team has recently decided to limit its target user base to professional usage. Based on this new focus, relevant use cases will be identified and the GIMP will be optimised to meet those requirements.

In other projects, a limitation of the target user base is not appreciated. EGroupWare, for example, an online groupware application, aims at a wide user base. In order to still support different groups of users, user profiles are identified and the system is tailored to optimally support each of these profiles individually.

Lessons learned from the GIMP:

- If possible, limit the *target user base* to be *feasible for the given project.* Just because its source is free to everybody does not mean it needs to be used by everybody.
- *Identify user profiles* by surveys or task observation and support them in an appropriate way.

4 Model of an Integrated Development Process

4.1 Preliminaries

While the actual usability process should not much differ from the User-Oriented Design Process described by ISO 13406 (ISO, 1999), some preconditions need to be met in order to allow for a seamless integration of the usability process into the OSS development cycle.

1. *Find the right people.*
   On both sides, competent and influential partners need to be identified. While the usability specialist should know what he is talking about, on the development side there needs to be a common attitude towards usability and a willingness to collaborate in a long-term relationship. Finding an influential primary contact for the usability specialist further supports the collaboration.
2. *Identify the project goals.*
   No matter in what development state the usability specialist gets involved with the project, the first step should be to identify the goals of the project, and which users they want to address independently of the actual current user base.
3. *Establish communication channels.*
   While Open Source Software projects usually possess a valuable infrastructure to communicate about technical details, communication channels between usability and development need to be established. In these

channels, technical language should be kept to a minimum, still usability specialists should get a frequent update of current activities on the development side.

4.2 Integrating Usability with the Development Cycle in OSS Projects

As mentioned in the previous section, an optimal User-Oriented Design process in Open Source Software projects itself much resembles the proposed ISO 13407 (ISO, 1999) standard: An iterative process of optimising design solutions along evaluations in real world scenarios.

Apart from the above preliminaries, we suggest two additions to the User-Oriented Design process as illustrated in Figure 1: First, beforehand the requirements phase, place an online survey among current and potential users that aims at clearing up misconceptions about user habits and expectations. This survey is not meant as a method to collect requirements, but as a reality check for the developers and a tool to enforce the integration of usability work.

Second, even in hierarchically structured OSS projects the likeliness of activities and additions not considered in the initial requirements plan is very high. By means of the above described communication channels, the usability specialist should keep an eye on current activities in the project and react flexibly by measuring them against the user requirements.

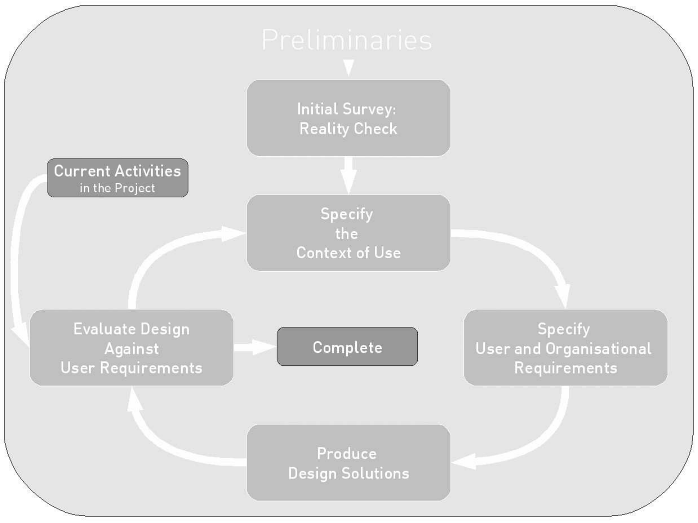

Preliminaries
Initial Survey: Reality Check
Current Activities in the Project
Specify the Context of Use
Evaluate Design Against User Requirements
Complete
Specify User and Organisational Requirements
Produce Design Solutions

Fig. 1. User-Oriented Design process as described in ISO 13407 (ISO, 1999), extended by two factors suggested to be relevant in integrated OSS development processes.

5 Conclusion

Due to the high number of passionate technical and non-technical contributors, Open Source Software has a high potential to gain not only technically excellence, but to offer users an optimal use experience. Still, usability contributors face difficulties when collaborating with OSS projects. The experiences of members of the initiative OpenUsability described in this paper form a first base of suggestions to overcome those difficulties and implement user-oriented development styles in Open Source Software projects.

References

1. Healy, K. and Schussman, A. (2003). The Ecology of Open Source Software Development. `http://opensource.mit.edu/papers/healyschussman.pdf`, 2003.
2. Hippel, E. v. (2001). Open Source Shows the Way: Innovation by and for Users No Manufacturer Needed! `http://opensource.mit.edu/papers/evhippel-osuserinnovation.pdf`, 2001.
3. International Organization for Standardization (1999). ISO 13407:1999. Human-centred design processes for interactive systems.
4. Raymond, S. E. (1998). The Cathedral and the Bazaar. `http://www.firstmonday.org/issues/issue3_3/raymond/`, 1998.
5. Yeats, D. (2006). Open-Source Software Development and User-Centered Design: A Study of Open-Source Practices and Participants. Ph.D. diss., Texas Tech University.

Experiences in Building a Free and Open Source Software Training and Certification Programme

Anas Tawileh

School of Computer Science, Cardiff Univeristy,
The Parade, Cardiff CF24 3AA, UK
anas@tawileh.net
WWW home page: http://www.tawileh.net/anas

Summary. The Free and Open Source Software (F/OSS) has achieved an enviable growth during a short period of time. However, that rapid growth in adopting F/OSS was not complemented with appropriate development of capacity building and educational programmes that focus specifically on F/OSS. This imbalance of supply and demand of F/OSS skills and competencies has to be corrected in order to remove any limitations to F/OSS adoption and to promote higher levels of F/OSS acceptability. In this paper, we present the experiences of designing, developing and delivering an integrated skills development programme for F/OSS. The programme is intended to create the core of a highly capable F/OSS community of practice to realise the benefits afforded by F/OSS in contributing to the development of the information society in Syria and the Middle East.

1 Introduction

Undoubtedly, the Free and Open Source Software (F/OSS) movement has succeeded significantly in securing a respectable position within the international software landscape [1]. Companies and governmental organisations are increasingly relying on F/OSS applications to empower their IT infrastructures while realising better value and returns on their investments. However, this widespread adoption of F/OSS was accompanied by a relatively slow development of educational programmes and services to back its success by enabling IT professionals to deal effectively with this new, rapidly evolving technical environment.

This paper will present the experiences of introducing an integrated skills development and certification programme for F/OSS developed with the support of the Network of Syrian Scientists, Technologists and Innovators Abroad (Nosstia) [2], an NGO in Syria aiming to accelerate technological development in the country to fulfill the requirements implicitly mandated by the international progression towards the global information society. The paper highlights the essential success factors and the obstacles encountered during the programme's design and implementation in order to assist organisations and communities intending to introduce similar programmes and initiatives to focus their efforts on the critical elements and avoid the problematic aspects.

2 Programme Inception

The accelerating progression towards the global information society has placed substantial demands on developing countries to improve their technological capabilities in order to bridge the digital divide separating them from their developed counterparts [3]. However, such improvement requires significant investments in different fields. Most developed countries suffer from shortages in financial and human resources, and can not afford to divert funds dedicated to essential development projects to build their technological capabilities.

The Free and Open Source Software (F/OSS) offers an attractive option to solve this dilemma [4]. F/OSS is developed in an open environment, under transparent development processes based on contributions from developers spread all over the world [5]. Licensing agreements for F/OSS grants the user more freedoms than of the proprietary software applications. F/OSS users have the right to access the source code of the software, study it, modify it and redistribute it for any purpose without obtaining permission from the original author of the application [6]. These features enabled F/OSS to attain unmatched growth rates and attracted many talented programmers to contribute to its development.

It can be safely argued that the characteristics of F/OSS are specifically relevant to the capacity building initiatives in developing countries [4]. Most of the applications released under F/OSS licenses are distributed free of charge. This will eliminate the financial burden associated with software costs from developing countries willing to initiate major technological development projects. More importantly, the freedoms granted for the user under F/OSS licenses facilitates the acquisition of skills and competencies required to raise the technical level of knowledge workers. The accessibility of the source code facilitates learning and localisation of application software, leading to higher adoption of F/OSS in particular, and technology in general.

The F/OSS Certified Web Development Professional Programme (CWDP) [7] was designed and launched to realise the benefits of F/OSS for the development of Syria. The prorgamme aims to build appropriate capacity for understanding, developing and implementing F/OSS in Syria and the Middle East.

The programme was introduced at Nosstia's Center of Excellence, and supported by two universities and a dedicated industrial committee. This multi-stakeholders approach was adopted to represent the different interests and perspectives of all stakeholders in the educational process, and to reflect the real needs and requirements of the marketplace. The process started with a detailed market needs analysis to determine the highest priority skills and competencies that should be covered by the programme's curricula. By selecting the constituent modules based on valid analysis of the industry, relevance of the materials will be guaranteed and job prospects for graduates will be greatly enhanced.

The LAMP platform for web-based application development (GNU/Linux [8], Apache [9], MySQL [10], PHP [11]) was an excellent candidate. Justification for our choice of LAMP was based on the significant adoption of these technologies in building dynamic web applications, and the global shift from client-server architectures to web-based applications. In addition, the platform is solid and reliable, and is completely based on Free and Open Source components.

3 Content

The F/OSS Certified Web Development Programme was developed to provide an effective learning environment to teach F/OSS-based web development principles and practices to would-be application developers and web masters in an engaging and interactive environment. The programme consists of 5 modules covering the areas of Software Engineering, the GNU/Linux Operating System, MySQL Database Management System, PHP Scripting Language and Apache Web Server. Each module is taught over 40 hours of instructor-led sessions, and assessed through a formal examination process and practical assignments that reinforce what students have learnt during the course. The coursework and assignments are specifically designed to stimulate collaboration and teamwork among participants in an environment that highly resembles the actual software development process in F/OSS over the Internet. Such encouragement for collective learning and development is hoped to foster the establishment of a core community of practice for F/OSS professionals and promote a more open attitude towards knowledge sharing and mutual support. By demonstrating the power and benefits of knowledge sharing and collaborative development, participants will be more likely to contribute to the technological development in their localities, and to the global F/OSS community at large.

The first delivery of the programme attracted 24 participants from different backgrounds and widely varying skills and expertise levels. Participants included governmental employees, private sector employees, technology consultants and university students. Evaluation so far indicates a remarkable success and a significant increase in the F/OSS competency of participants. These first attempts created a momentum for additional courses and programmes on other areas of F/OSS both within and outside Syria, and development of new and revised modules is currently underway. Programmes under development include: Information Security and F/OSS, F/OSS Content Management Systems and Advanced Networking.

4 Lessons Learnt

During the programme development process, many issues were raised that required special attention. Firstly, the scarce availability of qualified trainers and curriculum developers disrupted the smooth delivery of the course modules as scheduled. The high demand and the limited supply of F/OSS professionals which can also be attributed to the lack of appropriate capacity building programmes was the most serious obstacle. The difficulties of finding appropriate trainers resulted in the programme spanning over a longer period of time than speculated in the inception phase. However, participants of the programme were highly understanding and appreciated the difficulties, which eventually led to the smooth, though late, delivery of the programme.

Secondly, development of the training materials required more effort than what we perceived at the outset of the project. In spite of the significant volume of materials freely available online about the taught applications, they were highly scattered and disorganised. Our teaching approach required a concise, easy to follow supporting materials to augment the classroom course delivery. Furthermore, the training materials had to be integrated between the different course modules. For example, notes on MySQL database creation should complement the database design techniques presented in the Software Engineering module. Therefore, the development of high quality materials required extra effort from the materials' developers, and led to significant increase in cost.

On the other hand, the adopted approach of balancing assessment across coursework assignments and formal examination has received positive feedback from the course participants. According to collected evaluations, the coursework provided participants with a unique opportunity to share their experiences and to collaborate with their peer students. This has resulted in higher appreciation of the values of sharing and contribution within the F/OSS community and stimulated students to participate more actively in the development of F/OSS projects.

The improvements of skill levels of the programme participants will be demonstrated by showcasing the applications they developed during the programme, and will be evaluated by external reviews by experts in the field. An interesting observation is the effect of background diversity of participants. Participants were asked to identify their expectations from the course from the early beginning. The statements clearly showed the bias of participants towards knowledge areas they feel comfortable dealing with. This is an essential problem to the assignment of job roles where candidates possess specific required competencies while lack other, equally important, skills. Progressively during the course, participants developed better appreciation for the other skills and started to bridge the gaps they have which may inhibit them from effectively undertaking the activities of a professional web developer.

We believe that the skills development aspect of the F/OSS phenomena is still evolving, and an area of great potential for progress and improvement. Ultimately, for F/OSS to be widely adopted by businesses and governmental organisations, its technical superiority should be supplemented by robust and comprehensive skills development and assessment programmes to secure the availability of sufficient supply of skills capable of dealing with these emerging technologies. We believe that the F/OSS Certified Web Development Programme is a small contribution towards the realisation of this aim. We are willing to share all our experiences and encounters with other initiatives and programmes on order to promote the merits of F/OSS and to build a network for constructive collaboration and mutual progression to achieve excellence in F/OSS training and education.

References

1. Wheeler, D. A.: Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers! `http://www.dwheeler.com/oss_fs_why.html`, 2005, accessed March 15, 2006.
2. Network of Syrian Scientists, Technologists and Innovators Abroad (NOSSTIA) `http://www.nosstia.org.sy`, accessed May 15, 2006.
3. Bates, T. (2001). The continuing evolution of ICT capacity: The implications for education. The Changing Face of Virtual Education. (pp. 29-46). G. M. Farrel. Vancouver, Canada, The Commonwealth of Learning.
4. James, J. (2003). Free software and the digital divide: opportunities and constraints for developing countries. (pp. 25-33). Journal of Information Science, Vol. 29, No. 1.
5. Raymond, E. (2000), "The Cathedral and the Bazaar", `http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/`, (accessed September 4, 2005).
6. Stallman, R. (2002) Free Software, Free Society, GNU Press, Boston MA, USA, ISBN: 1-882114-98-1.
7. Nosstia Center of Excellence (2006) "Certified Web Development Professional Programme", `http://www.nosstia-nce.org`, (accessed May 10, 2006).
8. The GNU/Linux Operating System (2006), `http://www.linux.org`, (accessed May 15, 2006).
9. The Apache Web Server Project (2006), `http://www.apache.org`, (accessed May 15, 2006).
10. MySQL Database Management System (2006), `http://www.mysql.com`, (accessed May 15, 2006).
11. PHP Scipting Language (2006), `http://www.php.net`, (accessed May 15, 2006).

An MSc Programme in Open Source Information Systems

Bülent Özel, Mehmet Gençer, and Chris Stephenson

İstanbul Bilgi University, Department of Computer Science
Kurtuluş Deresi Cad. No:47 34440 Dolapdere, İstanbul, Turkey
{bulent,mgencer,cs}@cs.bilgi.edu.tr

Summary. This paper outlines the rationale and programme of the MSc. Programme in Open Source Information Systems to be started in at İstanbul Bilgi University, Social Sciences Institute, Turkey. The programme proposes a Masters of Science degree in Information Systems with an emphasized focus on Open Source Systems. A fundamental goal of this programme is to create a MSc degree which students will wish to pursue, which will be well regarded by industry and recognized by the relevant research institutes.

1 Introduction

Open Source Computing is a unique technological phenomenon of the 21st century. As the research and business opportunities in this field unfold, the need for education programs become more visible. This work presents a draft for the MSc program which provides a study path that accepts students from various backgrounds and career focuses.

It is evident that there has been growing interest in study of open source systems. Open Source modules, courses, and degrees in some higher eduction programs show that students are also becoming more interested in the topic[1]. In addition, open source phenomenon has been the focus of many researches for last four-five years.

Experiences and discussions elsewhere[2, 3] hints us the necessity to combine technical, social and business aspects of open source computing in a multidisciplinary framework which is expected to enable graduates of this programme gain a broad understanding of the issues and develop skills of their choice for various technical or managerial positions in e-commerce, software industry, public services, etc.[1] Thus, the focus of the program is not solely on

[1] It is noteworthy to follow discussions of MoLOS, which is a European group interested in the design and implementation of postgraduate studies on OSS. All materials produced by the group and mail list archives are available for public access. For further information please see http://www.nongnu.org/masterlibre/.

FOSS as a software technology, rather as a contextualized phenomenon. In other words, it is not solely limited to the aspects, such as, how to administer and use FOSS products.

Within this framework, we have deployed a two-year graduate level program presented below. The abbreviation FOSS is used throughout for 'Free Open Source Software', in accordance with EU terminology of the field.

2 Some of the Relevant Graduate Level Courses, MSc Programmes and Research Centers

It is necessary to give a brief review of existing graduate level courses, degrees, focus of research centers on FOSS. The range of course offerings on FOSS varies from a single term seminars to a full degree in Masters of Science.

2.1 Graduate Level Courses

Open Source Software Engineering Course at the Department of Computer Science at the University Victoria, CA

University of Victoria offers a graduate course in open source software engineering at the Department of Computer Science[2]. The course introduces students FOSS research area and aims to unveil the aspects of the FOSS so that they can make a contribution to it. As such, the course provides an overview of FOSS, intellectual property and software licenses, the communities around FOSS, software engineering practices in major FOSS projects, the economics of FOSS, and FOSS in industry[1].

This single course offering gives its graduate students an opportunity to examine FOSS itself, its history, its tenets and merits, methods, and other abstract aspects rather than focusing on relatively more technical aspects.

Open Source Systems Course at School of Informatics, City University of London, UK

The City University of London offers an Open Source Systems as an elective module within School of Informatics[3].

The module aims to evaluate the usefulness of open source systems to solve real world problems, by drawing upon known case studies. The course suggests a critical analysis of the open source software development process, its opportunities and problems in developing IT systems. To compare and to contrast open and closed source software development the module refers to established opportunities and difficulties in developing IT systems. In addition, proposed module aims to assess open source systems from the legal, ethical and ideological perspectives.

[2] `http://www.csc.uvic.ca/`

[3] `http://www.soi.city.ac.uk/pgcourses/module_list.html`

2.2 MSc Programmes

Some universities in United Kingdom, Italy, and Spain already offer or full Masters programmes in Open Source.

MSc in Open Source Systems, at Sheffield Hallam University, UK

In previous years Sheffield Hallam University proposed one year Masters programme in Open Source Systems. The programme offered a technical background in Software Engineering with an open source emphasis. The programme was targeting "people with a background in computing, science or engineering who want to start a career as independent consultants in open source solutions[4]" However, the programme is currently inactive for the academic year 2005-2006.

MSc in Open Source Computing at University of Lincoln, UK

Department of Computing and Informatics at University of Lincoln offers one year full time programme towards a MSc degree in Open Source Computing.

The gradute students of the programme are expected to profess in a variety of open source business models, design methods, and technologies through planned practical experiences and implementation based research projects.

Offered courses within the programme is as follows[5]:

- Introduction to Open Source Software and OS Case Studies
- Object Oriented Design and Development (UML and Java)
- Advanced Software Engineering for the Internet
- Distributed Computing with Open Source components
- Project Preparation and Professional Issues
- Open Web Technologies and Service Development
- HCI, Multimedia, Graphics, and Human Factors in OS
- Open Source Business Models
- Project and Dissertation

The programme not only addresses the development of open source systems from the technical aspects but also targets to examine FOSS's wider impacts on society in terms of business models and social change.

[4] See `http://www2.shu.ac.uk/prospectus/op_pglookup1.cfm?id_num=CMS030`

[5] See `http://www.lincoln.ac.uk/home/courses/dci/postgraduate/open_source/index.asp`

International Masters Programme in Free Software, Open University, Catalonia

The Open University of Catalonia (Universitat Oberta de Catalunya) offers an International Master programme in Free Software. The master programme that has began in 2003, offers a distance education of the degree using the Internet as the basis of the academic activity[3].

This master degree targets to cover and examine various aspects related to the FOSS technology, such as philosophy and the history of the FOSS movement, GNU/Linux, computer networking, web management, database management, software development, laws and licenses, etc. The programme offers four different sub-tracks. The student can choose one of these four possible specializations:

- Network and Systems Administration: Focuses on configuration and administration of network services, and security in GNU/Linux.
- Software Development: Focuses on design and development applications with FOSS tools, such as different virtual cooperative environments for FOSS programming.
- Web and e-Business Administration: Focuses the components of widely used FOSS web servers and their possibilities, along with management of related data base management systems with web interfaces.
- Information Systems Management: Focuses on managerial and strategy development aspects of FOSS adoption. For instance, this sub-track aims examines the alternatives to proprietary systems, investigates the legal and exploitation aspects related to FOSS and analyzes practical implantation cases of FOSS platforms.

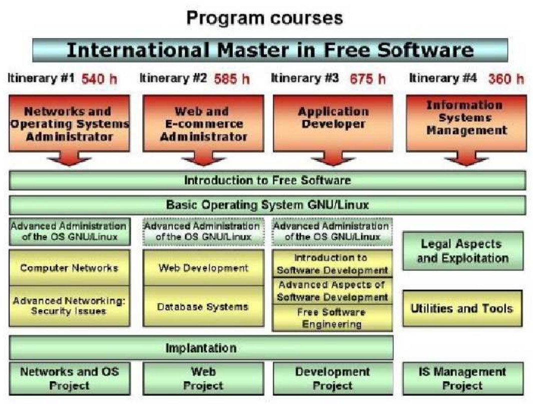

Program courses
International Master in Free Software
Itinerary #1 540 h
Itinerary #2 585 h
Itinerary #3 675 h
Itinerary #4 360 h
Networks and Operating Systems Administrator
Web and E-commerce Administrator
Application Developer
Information Systems Management
Introduction to Free Software
Basic Operating System GNU/Linux
Advanced Administration of the OS GNU/Linux
Computer Networks
Advanced Networking: Security Issues
Web Development
Database Systems
Introduction to Software Development
Advanced Aspects of Software Development
Free Software Engineering
Legal Aspects and Exploitation
Utilities and Tools
Implantation
Networks and OS Project
Web Project
Development Project
IS Management Project

Fig. 1. Open University of Catalonia FOSS Master Programme Design

Fig.1 highlights the details of distant FOSS Programme design offered by Open University of Catalonia.

Masters in Free Software and Open Source, University of Bologna, Italy

University of Bologna offers a full Masters programme on FOSS. Along with software engineering aspects of FOSS, the program also offers courses on Intellectual Proprietary Rights management in FOSS [6].

Masters in Open Source Software Management, University of Pisa, Italy

Department of Informatics at University of Pisa offers a full Masters degree in Open Source Software Management. This multi-disciplinary Masters degree is composed of courses offered by four different departments:

- Department of Informatics
- Department of Corporate Economy
- Department of Economics
- Institute of Trade and Marketing

2.3 Research Centers and Groups

A large number of universities around globe offer PhD seminars or readings on FOSS; run MSc thesis, PhD projects; and encourage academic studies on various aspects of FOSS at established active research centers or groups. Hereby, we will shortly mention only a few of them.

Open Source Software Development at University of California, Irvine, USA

Institute for Software Research at University of California has a research center focused on Open Source Software development. The center "focuses on empirically-based studies of the processes, practices, and communities that develop open source software. Ethnographic and virtual ethnographic research methods are employed in the field studies of open source software development communities[7]."

[6] See `http://www.unibo.it/Portale/Master/Master+Universitari/2004+2005/Tecnologia+del+Software.htm`

[7] Please see '`http://www.isr.uci.edu/research-open-source.html`' for studies and fruitful publications of the center

Research Project on the Free/Open Source Software (F/OSS) Development Phenomenon, Univeristy of Notre Dame, USA

A research group at University of Notre Dame runs a long run research project on FOSS development phenomenon and examine the pattern of growth exhibited by FOSS projects over time. Their approach is primarily based upon social network analysis methodology to derive conceptual models of the FOSS phenomenon[8].

Open Source Research Community, MIT, USA

The MIT Open Source Research Community is basically driven by a group of researchers at MIT Sloan School of Management. The research group mainly investigates FOSS collective development phenomenon as a breakthrough innovation process. The group also maintains database of research papers on FOSS, discussion forums, and news portal on FOSS academic activities [9].

3 Key Features

Deriving upon existing programmes and courses it has been foreseen that for the developed program to be attractive, the students those will graduate from the programme should acquire qualifications which will provide them recognition in both academy and industry. Within this perspective the proposed programme offers a curriculum which goes far beyond the scope of conventional computer science and software engineering Masters degrees. It aims to propose a multi-facet understanding of licensing and copyright law, FOSS business models, socio-political reasoning for FOSS, which of all can provide a basis for both academy and industry.

The developed multi-disciplinary MSc degree in Open Source Information Systems combines following distinct ingredients within the body of offered curriculum:

- Technology Infrastructure Management: Choose, configure and administer network services, operating system and user software. This requires making informed choices when choosing hardware and software regarding continuity and reliability of services, understanding of network concepts and security, and problem solving skills for this kind of problems.
- Software Engineering: Design and develop applications with FOSS tools and methodology, know strategic choices in FOSS code re-usability, understand and deploy cooperative environments for FOSS development.

[8] For details please visit `http://www.nd.edu/~oss/`

[9] `http://opensource.mit.edu/`

- Information Systems Management: Know issues regarding legal aspects of different licenses for software and information, make strategic choices for information system technical superstructure, and deploy flexible teamwork methodologies within FOSS business model.

The mode of study will keep the very tenet of the self learning practice of the FOSS community, where people usually acquire and share knowledge by visiting different web pages and discussing in forums, etc.

It is deemed that offered program should keep the flexibility for both as a professional skill development programme and as an intermediary research programme providing background towards more elaborate PhD studies within this intrinsically multi-disciplinary field. Proposed program opts to offer a non-thesis master degree, where each student is expected to either initiate and support or contribute to an existing FOSS project within their masters project. This very principle of FOSS, 'contribution' back to the community, will also be embodied within the practical courses. Thus, it will also allow the students to experience open source development process.

While introducing FOSS in a Master program, we are aware of the fact that a broad multi-disciplinary curriculum is necessitated, which means a discontinuous break from conventional expectations of an MSc degree. Besides, even though, the benefits of FOSS articulated from its practices and philosophy will be rigorously demonstrated within the programme, the programme will also seek the avoidance of monoculturalism in technology; as FOSS itself inherently is not part of a dominant monoculture and is not even a monoculture to itself. For instance, the forking practices in FOSS projects are a technology of flexibility and differentiation.

The programme expects both computer professionals and new graduates from various technical background, who want to acquire managerial skills in Information Systems with an expertise in FOSS, and non-technical professionals and students such as from Economics, Political Science, IT Law, and Business Administration who want to develop FOSS technical skills along with its managerial and business aspects. The programme also expects students who want to pursue a PhD study in a relevant field. In short, the programme does not strictly prerequisites who should enroll but rather what should be acquired by graduation.

4 Courses and Curriculum

4.1 Courses

The programme offers both newly developed courses, and some relevant graduate level Business Information Systems courses, which of all are already being delivered at the university:

- *OSIS 501 Essential GNU/Linux Usage and Administration*: Open source software systems have different characteristics than closed systems in terms of how they interact with the user and how approaches to administration. This applied course aims to introduce students to methods of evaluating rich choices that are exposed by open source systems to users and administrators, in addition to basic principles and practice of data processing and system management. Upon successful completion of this module, students will be able to use GNU/Linux systems and additional software for office needs and various data processing scenarios, and apply essential system security and administration practices on these systems.
- *OSIS 502 Network and Operating System Management*: Open systems and e-commerce applications require careful implementation of security policies. Additionally, in single or multi-office businesses which rely on computer and network technology, computer networks must be deployed heavily and managed securely, in addition to installation of operating system infrastructure suitable for business processes involved. Subject of this course is equipment, operating system, superstructure software, and application methodology under network integration and security perspective. Necessary mechanisms and policy variants suitable for these mechanisms are covered in an applied manner.
- *OSIS 503 Software Development Using Open Source Tools and Methodology I*: Open source covers both a specific type of licensing and consequent development methodology, and a great number of highly accessible software packages. This course covers basics of using these software, methods and usage patterns, for software development and management/monitoring of the development process.
- *OSIS 504 Software Development Using Open Source Tools and Methodology II*: This course takes subjects covered in the first part further, by deploying more extensive and realistic practice of software development processes.
- *OSIS 505 Social Aspects of Open Source Systems and Public Policy Development*: Open source approach is regarded as a methodology which promotes innovation, accessibility and free competition in inter-organizational processes. This course explores social perceptions and sectoral influences of licensing, and introduces students to open source approaches for promoting development in sectoral, regional or higher levels, by utilizing relevant case studies.
- *OSIS 506 E-business System Development and Infrastructure Management*: E-business and e-government systems have distinct requirements in terms of security, service continuity and compatibility. This course covers strategies and practices for developing, scaling and sustaining e-business systems by using existing open source software base. Upon completion of the course students should demonstrate practical performance in choosing, installing, customizing, monitoring and maintaining software in relevant scenarios.

- *OSIS 507 Legal Aspects of Software and Information Licenses*: Just like their commercial counterparts restricting use of software and information for protecting rights to appropriate, public licenses place restrictions to guarantee free-flow of information and an open and continuous productive activity. This course takes on the similarities and differences of the two licenses in terms of how they relate to productive activity, and introduces students to legal consequences and possible conflicts arising from these licenses with case studies.
- *OSIS 508 Open Source Business Strategies*: Unlike protective licenses which aim to limit flow of information, public licenses are designed to guarantee its freedom. This has been the reason they have arisen from software market which relies on innovation at large scale, and was fit for the nature of this market. This course focuses on new business methodologies shaping around public licenses, within sectors such as Internet and software where inter-dependency of actors is quite high, and analyzes possibilities of applying these to more conventional business practices.
- *BUS 505 Research Methods*: The aim of this course is to develop understanding and skills for conducting qualitative research. Survey development, sample selection, data collection and analysis, use of information technology for research and presentation of results are among the subjects covered. Students will be introduced to information resources, qualitative and quantitative data collection methods, measurement and reliability, basics of information analysis, and reporting techniques.
- *BUS 542 Information Systems Management*: This course approaches information technologies from management perspective. Students are first introduced to New Economy and discuss its effects on society. After covering elements of organizational information architecture and inter-organizational systems, the course proceeds into data stores and their management, data mining, decision support, artificial intelligence and information management, organizational resource planning (ERP), supply chain management, customer relationship management. Finally system development methods, system installation and administration, and transformation management are covered under a decision making perspective. Throughout the course existing and developing technologies are studied for their effects on people and organizations.
- *OSIS 511-512 Masters Project I-II*: The aim of the project is to provide students a chance to practice in a subject of their choosing and apply research methods. While student research on their subject with a critical standing, they can develop analytical skills and and put the methods they have learned in other courses into practice.

4.2 Curriculum

The programme curriculum is given below. Note that the program is devised for a 15 week semester each in 3 hour seminar basis. That is each course has a length of 45 hours in total. In addition, a 50 hour customized intensive non-credit proprietary weekend seminars will be given at the beginning of the first semester to leverage the backgrounds of the incoming student groups.

- Semester 1: *OSIS 501; OSIS 503; BUS 505; BUS 542.*
- Semester 2: *OSIS 502; OSIS 504; OSIS 506; OSIS 508.*
- Semester 3: *OSIS 505; OSIS 507; OSIS 511.*
- Semester 4: *OSIS 512.*

Each student is expected to have a pass grade to earn 3 credits per course (30 credits in total for the degree), plus, they are expected to complete and present an acceptable two-semesters long non-credit Masters Project.

5 Summary and Future Directives

The programme proposes a Masters of Science degree in Information Systems with a emphasized focus on Open Source Information Systems.

The program aims to give its students a broad education and a range of technical and research skills, with an emphasis on Open Source, Open Standards, and Open Content. It presumes a student profile with a background in computer science, natural science, engineering, or business management who wants to start either a career as an independent consultant in open source solutions, or who wants to pursue research towards a relevant PhD program.

Upon the launch, the program will seek to initiate or to adjoin a relevant European Masters Course consortium within Erasmus Mundus[10] programme. The language of conduct of the programme will be in English.

References

1. Daniel M. German, Experiences teaching a graduate course in Open Source Software Engineering, *Proc. of Open Educational Symposium of the 1st International Conference on Open Source Systems*, pp. 326–328. (2005)
2. P. Laird and S. Barrett, On the Dimensions and Issues of a F/OSS Curriculum, *Interim report of tOSSad Project (www.tossad.org) on FOSS Curricula Development*, (2006)
3. D. Megias, J. Serra, and R. Macau An International Master Programme in Free Software in the European Higher Education Space *Proc.of the First International Conference on Open Source Systems*, pp. 349–352. (2005)

[10] Erasmus Mundus is the EU co-operation and mobility programme in the field of higher education which promotes the European Union as a centre of excellence in learning around the world. For further details, please visit `http://europa.eu.int/comm/education/programmes/mundus/index_en.html`

Polaris4OS: a best practice for training and adoption of F/OSS in SME

Giulio Concas[1], Michele Marchesi[1], Andrea Piras[2], Stefano Sanna[2], Efisio Sciola[3], and Selene Uras[1]

[1] DIEE, Dept. of Electrical and Electronic Engineering, University of Cagliari
Piazza d'Armi, 09123 - Cagliari, Italy
agile@diee.unica.it
WWW home page: http://agile.diee.unica.it/home.html

[2] CRS4, Center for Advanced Studies, Research and Development in Sardinia
Polaris, Ed. 1, C.P. 25 - 09010 Pula (Cagliari), Italy
{piras,gerda}@crs4.it
WWW home page: http://www.crs4.it

[3] ApritiSoftware!
Via Monteverdi, 12 09129 Cagliari, Italy
e.sciola@apritisoftware.it
WWW home page: http://www.apritisoftware.it

Summary. The methodical adoption of F/OSS in SMEs raises critical problems related to issues such as technology, organization, culture and business model choices. In this paper, we describe Polaris4OS, a successful experience of F/OSS adoption among CEOs, managers and developers of ICT companies. The lack of F/OSS culture has been identified as the most critical obstacle for the exploitation of the new opportunities offered by the F/OSS communities.

1 Introduction

F/OSS adoption in SMEs introduces some non-trivial drawbacks. For business men and chief technology officer (CTO), F/OSS is critical since there are not case studies and deep analysis about it, making them lost in a huge collection of untrusted reports. On the other side, it is perceived by project managers as a critical resource, for which they have nobody to blame for misfunctioning or troubles. Finally, developers consider F/OSS as a synonymous of "Save as..." command in their browser: F/OSS is the cheapest way to acess example code and get ready-to-run libraries and applications. Both business and technical people miss most potential opportunities offered by F/OSS, like knowledge sharing, collaborative software improvement, large feedback basis, community support. We can consider F/OSS and its philosophy, spirit, methodology, tools and community, as an opportunity which requires skills and expertises outside the official learning path represented by universities, schools and certification centers. A new approach and learning program is needed to take advantages of the F/OSS ecosystem, as consumer and producer actor.

This paper mainly focuses on an "open source curriculum" defined and applied in the Polaris4OS project. It enables developers to effectively work in the F/OSS world and encourages chief executive officers (CEOs) to take advantage of these opportunities: it implies new learning methodologies, new skills to supersede closed and proprietary approaches, new business models to optimize company resources.

2 The Project

The main goal of the Polaris4OS project is to fill the gap between well established proprietary enterprise processes and the upcoming F/OSS model, based on collaboration and knowledge sharing. Eleven locally-based small and medium IT focused enterprises (SMEs) were involved in design and development of extension modules for existing open source frameworks. They had been selected according to the mission of each company and their market.

Polaris4OS consists of a preliminary set-up phase and three executive phases: the training based on a blended learning model, the collaborative design and the open source distributed development.

2.1 The Preliminary Phase

During the preliminary phase, companies were invited to evaluate existing state-of-the-art open source projects, and were asked to select projects to contribute to. They identified an Enterprise Resource Planning (ERP) and a Content Management System (CMS), which were considered as strategic solutions for their core business.

Then the project mentors met the companies to design and plan the next phases. Their core business had been totally based on proprietary software, with both custom solutions and off-the-shelf applications. Moreover, they were competitors in the same geographic area and they never cooperated. They had different experiences with a mixed background of Java and .NET environments, and each had its own software engineering methodology, usually based on practice and not on a formal approach. The preliminary meetings let to set-up a trusted environment in which companies agreed to share their experience, knowledge, resource, code and developers. The mentors identified the skills and the main competence areas needed to participate to selected projects. The key technologies were deemed to be object oriented programming, integrated development environments, collaborative utilities, versioning systems; the most important methodologies were considered to be design patterns, test driven development, continuous integration; the key processes were deemed to be collaborative work and frequent iterations. The project effort was established in two man-days a week per company, for a timespan of nine months.

2.2 The Training

Since the initial developers skills were very different, we decided to adopt a flexible learning method would support a training process mixing classroom lesson and laboratory activities, on-line training and support, collaborative learning among learners, coaching and tutoring [1]. The learning model answers to our necessities is the blended learning model (BLM) [2] [3]. It was supported by the Moodle [4] learning management system (LMS) and by the collaborative knowledge base realized with the learning objects published by the teachers.

Our model consisted of three main learning moments:

- the collaborative learning, based on specific and practical activities during a frontal lesson;
- the slf-learning phase. It gives to developers the possibility to use the material loaded in the LMS by teachers and to interact with coaches and tutors;
- the consolidation, when the developers came back in classroom to demonstrate the achievement of the administered concepts.

The BLM was realized in:

- 4 hours of classroom frontal lessons;
- 8 hours for individual study, collaboration, exercises based on the learning object prepared by teachers and having the support of tutors and coaches;
- 4 hours for classroom knowledge and capabilities assessment.

The LBS monitored developers activities, i.e. the time spent by each one on it and the result of on-line tests. From their analysis, we was able to define a training process tailored on individual necessities, creating a sort of learning customization, and to notice that the developers spontaneously create a community where people helped each other and shared materials and informations[5]. The community created in the first phase represent an important result for the project because it showed that the initial formers diffidence was overcame since the starting phase.

2.3 Designing

In such phase, we analyze the selected F/OSS projects and the definition of two specific application scenarios. To have the execution phases like a real project, we decided to have real customers therefore we selected the local administration of Pula, in Sardinia, and a private breedings company.

Meetings were taken to know customer needs and the software and hardware solution they used. It emerged the interest of the first customer for a tourist event alert system while the second one would like an automatic tracing system of animal growth for a pigs farm.

Tutors and coaches drove developers to evaluate the software solution and they found in Infoglue CMS [6] the most suitable F/OSS software to be extended including the module required by the local administration and in Compiere [7] the ERP to extend and modify in way to implement the tracing system. Once selected the softwares, the developer team worked to define the tasks and a preliminary TO-DO list. Each item list was assigned to a developer.

Design activities were mainly conducted remotely by means of chats, web forum and mails, with weekly interactions for requirement definition, tasks identification and assignment.

2.4 Module Development

The developers implemented the extension modules driving their activities with concepts learned in the first phases and using the results of the design process. The development model was based on continuous integration and frequent releases in order to verify the correctness of the implemented software and the referential integrity of the required changes respect to the original databases structure.

During weekly meetings, the encountered problems had been analyzed and the possible solutions discussed by all developers for the adoption of the best choice. In such meetings, coaches and tutors applied a learning-by-doing model [8] in way to address the developers lacks raised in the course of this phase. A set of problems was related to the inexperience of the development F/OSS model like the remote and distributed collaboration model and the possibility of reusing the source code and the databases of the selected softwares.

For both of modules, the first development task was the installation of the original software and set-up of the integrated development environment (IDE). There were not available automatically procedures for the installation then the developer executed in separately steps the DBMS installation, the database creation, the Compiere or Infoglue installation and the connection between database and these softwares. Regarding to the IDE, the selected one was Eclipse [9].

The following tasks were tailored on the module. The alert system for Infoglue required the mapping of the new tables on the databases structure using Hibernate, the definition of the template for the web pages and the alert system. The system function realized were user registration, session and message management.

The tracking module is related to the European Union directive on the traceability of food products. In the case of a pigs farm the batch is identified by a code related to the week when the pigs born and it lets to identify their diet and medical treatments. Another customer request was to store the production costs. The developers studied the default database then modified

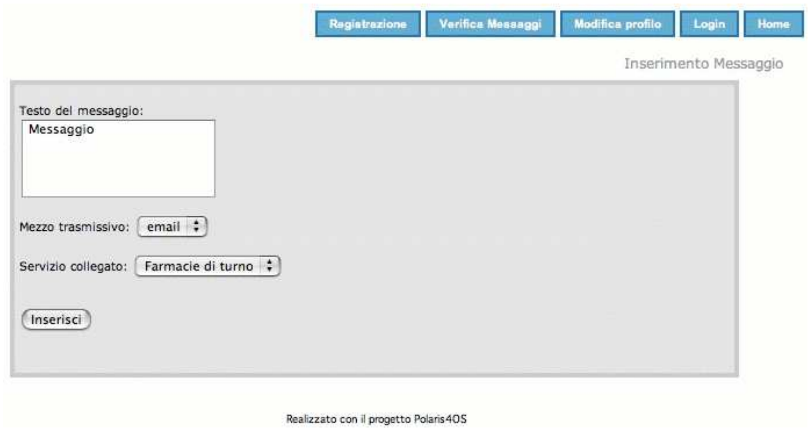


Fig. 1. The web page to insert messages.

the original one. Using the Compiere tools, the mask to insert data was created and integrated in the window management system of application.

2.5 The Companies Participation

Companies involved in Polaris4OS come from different IT market areas and they have different size. Most of them are focused on services for Public Administration, while others work in the Internet context, with web services and application, mobile solutions, digital television and professional training. Average team size is 3-4 developers for each company.

Before starting the project, each company presented itself to other participants, in order to better identify resources provided to the project and expected results. Technical team had experience in database engines, object oriented programming language and some web framework experience. Therefore, there was not an homogeneous expertise nor similar objective for those companies. Most important, there were no previous experience with open source projects.

CTOs have been introduced to F/OSS ecosystems, in order to provide them with all elements to decide what technology to adopt and what kind of OS project to work for. They have been preliminarily introduced to F/OSS licenses, their scope, their limits and their adoption across leading open source projects.

Companies have been encouraged to participate Polaris4OS because they would perceive the training phase as a valuable benefit for their team: they were able to gain new skill at no cost. However, they kept some doubts regarding actual developers effort: it was expected to be two days a week and they considered it too short to achieve a relevant product but, on the other side, they were unable to assign more resources to the project.

3 The Curricula

An important and none evident aspect of the Polaris4OS project was the definition of the developers curricula. The teachers, tutors and coaches guided their actions in the designing and development phases in way to fit it.

The curricula definition started with the training phase and it was adapted to achieve four main goals:

- all developers had to work on the project;
- all developers had to use the selected technologies;
- collaborative approach to all project activities;
- build a developers shared knowledge, practices and learning community for all the following project phases.

We defined a list of fundamental competences for the curriculum and grouped them in four categories:

- information seminar;
- software engineering methodologies;
- use of tools and IDE (i.e. Java, Eclipse and Sourceforge);
- features and constraints of the two selected open source projects (local administration vs. tracking system) .

Defining curricula was based on what can be used to design courses and curricula concerning Agile development [10].

3.1 Defining the Curricula

The preliminary analysis identified the training goals selecting the areas of knowledge and skills the teachers, coaches and tutors needs to focus the lessons and to plan what designing and development aspect had to be treated.

The output of this activity has been the definition of the curricula based on the needed skills and started from the owned skills in way to fill the competence gap. Developers meeting evidenced they needed to keep confidence with development methodologies like advanced development practices (i.e. design patterns), database management, data abstraction and web applications. They had not experiences on software development in distributed environment so they did not know practices like test driven development (TDD), continuous integration, unified modeling language (UML) design and related tools. Furthermore the developers need to study a ERP and a CMS. In Table 1 are listed the identified modules and their correspondent categories.

3.2 Results of the BLM

The developers participated actively to activities on Moodle platform and its knowledge base. It is interesting to note that they assumed a proactive pose

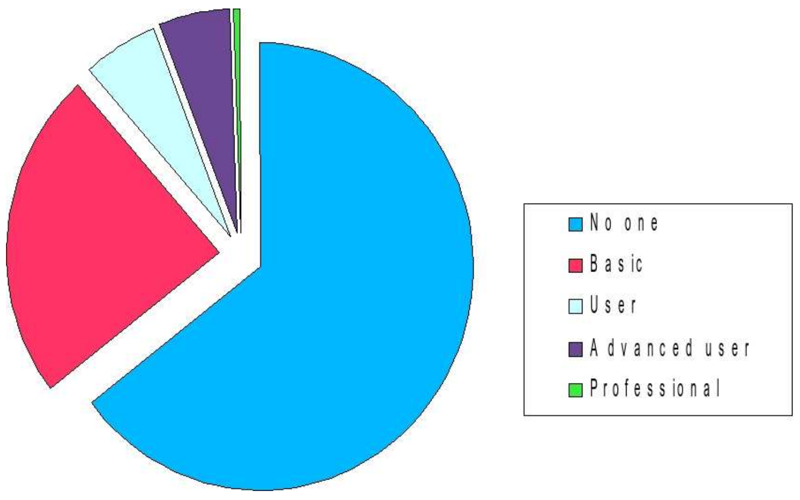


Fig. 2. The graph exposes the initial developers skills.

among the community. Many people had loaded appropriate material for other formers, they share informations, suggestion and doubts with on line coach and tutors. Problems was solved all together according to the collaborative learning. They appreciated the platform tools as much as web forums and chats.

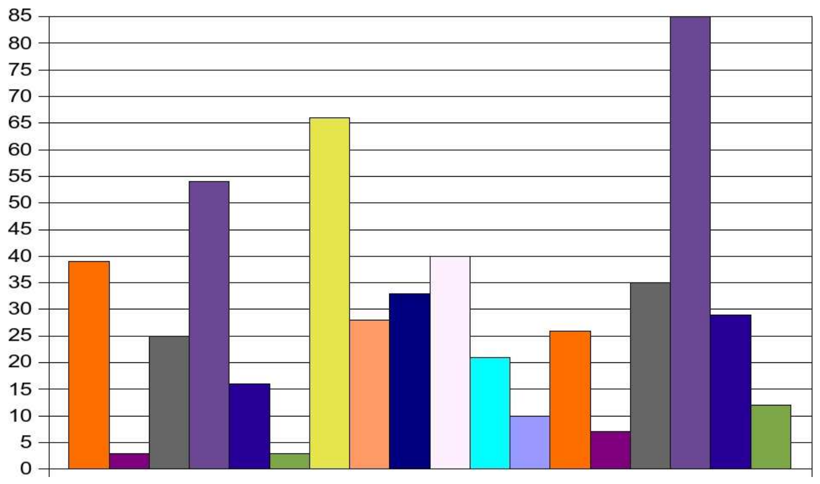


Fig. 3. Number of visits to Moodle platform for each developers.

Module Name	**Category**
What is Open Source? - Open Source licenses and philosophy	Seminar
Collaborative Environment and LMS - Moodle and its tools	Seminar
Eclipse Base - Basic use of Eclipse	Tools
Java Advanced - Design Patterns	Tools
Eclipse Advanced - Developing application using Eclipse and CVS	Tools
UML and tools - Diagrams	Methodologies
TDD e Refactoring - TDD, Junit, HttpUnit, Frequent release, Short iteration, Refactoring	Methodologies
Web Application Development - Tomcat, Velocity, JSP	Tools
DB and Data abstraction - Data persistence, DAO, Hibernate.	Tools
Distributed architectures and programming - Client/Server programming, Java network programming, XML standard	Tools
Infoglue CMS - System architecture and technologies analysis	Open source software
Compiere - System architecture and technologies analysis	Open source software

Table 1. The curricula modules and their correspondent category.

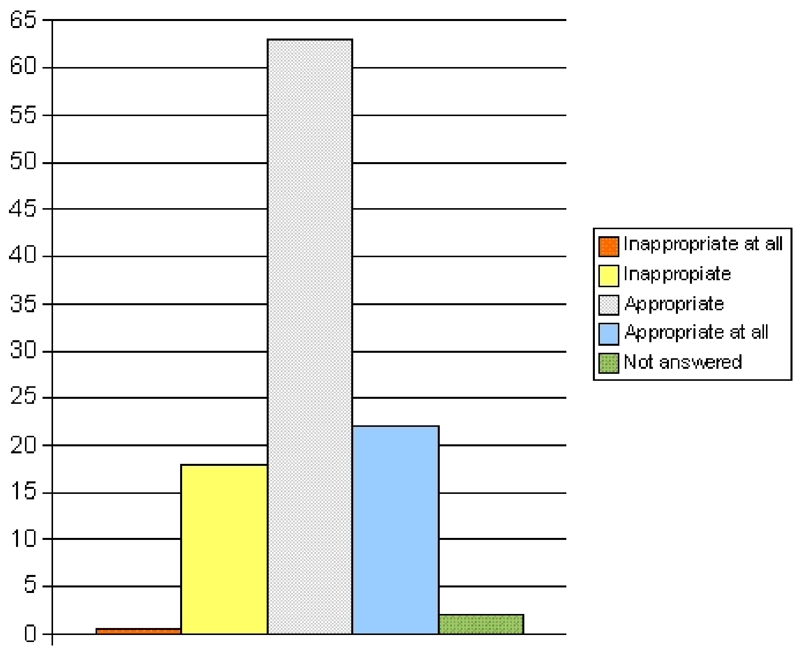


Fig. 4. The results of the final test.

At the end of the third executive phase, a test was submitted to developers to collect their sensations on the whole project. The 82% of them provided a positive judgment: the 62% evaluated the learning process as "appropriate" and the 20% considered "appropriate at all". Nobody found it "inappropriate". On such results, we can assert the contents tough were appropriate to our target.

4 Conclusions

Same works have exposed the use F/OSS like a learning and model environment, for example [11], [12], an [13].

The results of Polaris4OS enforces such considerations. It has been successfully completed and it can be considered a very interesting case study to evaluate overall impact of F/OSS adoption in SMEs. Both company managers and their developers appreciated the F/OSS model and the concrete opportunities it gives. Working with real-world open source projects, they learned they can enhance their business, products and skills by means of F/OSS methodologies and practices. However, migration to F/OSS requires a focused effort to be understood, accepted and implemented.

The learners participated actively to curriculum definition negotiating topics with teachers in order to both satisfy project needs and reach their own expectations. It was a sort of negotiation about the pros and cons of each proposal and we want highlight that the last two curricula modules, focused on Compiere and Infoglue, were required by CTOs and developers.

Other works have treated F/OSS

References

1. McNutt L., Brennan M., , Learning Styles and elearning, what is the Connection?. FIE'05. Proceedings 35th Annual conference, USA, 2005
2. Bunse C., Grutzner I., Ochs M., Peper C., Steinbach-Nordmann S., Applying a Blended Learning Strategy for Software Engineering Education. Proceedings of the 18^{th} Conference on Software Engineering Education and Training (CSEE&T), Ottawa - Canada, 2005
3. Beaton.C, Evolution of ethics blended learning. FIE '05. Proceedings 35th Annual Conference, USA, 2005
4. Moodle. `http://moodle.org` Shin-Wei Chou, Chien-Hung Liu,Learning effectivness in Web-Based Technology-Mediated Virtual Learning Environment2005
5. –
6. Infoglue. `http://www.infoglue.org`
7. Compiere. `http://www.compiere.org/`
8. Rogovin A., Learning by doing: home and school activities for all children, Abingdon Press, 1998
9. Eclipse. `http://www.eclipse.org`
10. R. Camplani, A. Cau, G. Concas, K. Mannaro, M. Marchesi, M. Melis, S. Pinna, N. Serra, A. Setzu, I.Turnu, S. Uras, Using Extreme Programming in a Distributed Team . Workshop: Sharing Experiences on Agile Methodologies in Open Source Software, Genova - Italy, 2005
11. Crowston K., Annabi H., Howison J., Masango C., Effective work practices for FLOSS development: A model and propositions. Proceedings of Thirty-Eighth Hawai'i International Conference on System Science (HICSS38), Kona - USA, 2005
12. Lehmann F.,FLOSS Developers as a Social Formation, `http://www.firstmonday.org/issues/issue9_11/lehmann/`.
13. Rishab Aiyer Ghosh, Free/Libre/Open Source Software as a learning environment. Proceeding of International Symposium on Open Source Software, Abano Terme - Italy, 2005

Computers for Kids: More Than a Simple Recycling Operation

Kıvılcım Hindistan[1], Boran Puhaloğlu[2], Emre Sevinç[2], Ergin Sevinç[3], Alper Tuna[1], and Vehbi Sinan Tunalıoğlu[2]

[1] Fazlamesai.Net `sundance@fazlamesai.net`, `parsifal@fazlamesai.net`
[2] İstanbul Bilgi University `boranp@bilgi.edu.tr`, `emres@bilgi.edu.tr`, `vst@bilgi.edu.tr`
[3] FarklıBireylere Eğitim Derneği `ergin@faber-der.org`

Summary. In this paper we present a case study from Turkey, a social responsibility project named "Çocuklara Bilgisayar Projesi" (Computers for Kids Project) which incorporates the dissemination of free and open source software as well as helping children and youngsters from low socio-economical sectors to have a digital experience using GNU/Linux running computers [1].

1 The Main Idea of the Project

The core idea of the project is simple: Many individuals and enterprises buy new computers in order to satisfy the higher criteria demanded by ever growing sophisticated software products. This means they are left with many secondhand computers that can be useful but since there is no systematic and "free" recycling facility, people don't know what to do with these computers.

The result is either unproductive use of old computers or simply disposing of them. Both cases are a missed opportunity to help young people that need computers. If there exits such a facility, people can simply donate their old computers and forget about them, knowing that the hardware will reach the correct target and be used in a productive fashion.

Thus, the project aims to be useful in three important ways: socially, environmentally and economically. Socially, by helping young people to reach computers, environmentally by preventing useful hardware becoming trash and economically by enhancing the digital experience and education quality of more people without much financial support, solely utilising the human resources quality of project participants and volunteers.

2 Implementation Details

The core idea as summarised above is simple but the implementation is not. As far as we know, there's no pioneering project that had similar aims and comparable results. When we looked at other social responsibility projects we saw that either the scope of the project was minimal such as The Free Linux CD Project [2] or it had extensive international and financial support such as the 100$ Laptop Project [3].

That is one of the reasons that motivated the project initiators: if we can show this to be feasible and practical with a minimum cost then the whole process can be adapted and implemented in different places of the world.

Such a project needs much more than technical know-how. It includes very careful project planning, locating a pilot region to start the experiment on a small scale, very close contact with local authorities and good support from at least one of the local NGOs (Non-Governmental Organisations) as well as a coordinating body between these organisations which may speak very different languages sometimes.

2.1 Planning, Coordination and Choosing a Pilot Area

There are three official partners in the project: Faber-Der, Fazlamesai.Net and BYSGM.

- Faber-Der is an NGO and the main initiator of the project [4]. The main purpose of the organisation is to provide education, psychological and social help to individuals who cope with difficulties in trying to adapt to society. Faber-Der is the main coordinator between technical partners and other bodies such as BYSGM described below.
- Fazlamesai.Net is the technical partner of the project [5]. This organisation provides a platform for technical discussions, where people related to information technology, open source and free software provide news, exchange ideas and initiate projects. Fazlamesai.Net had an active IT community for the last 5.5 years. Its members work in different organisations such as universities, consulting, software houses, etc. and have extensive experience in contributing to free and open source projects. This organisation is responsible for configuring the secondhand computers to run GNU/Linux in particular and technical consulting in general.
- BYSGM is a kind of meta-NGO that cooperates with many NGOs in Turkey as well as Begoğlu Municipality in İstanbul. This organisation has the capability of providing physical facilities as well as using the Smart City system of the municipality. The Smart City system is a database project that is used mainly by Istanbul's Beyoğlu Municipality to store socio-economical data about citizens living in Beyoğlu area (one of the biggest districts in İstanbul). The data is valuable because it shows the

economical status of families, how many children they have, the educational performance of children, etc. BYSGM is responsible for delivering the configured and tested computers to youngsters who really need them.

The main planning of project started in December of 2005. All three official partners had their internal plannings and with the coordination of Faber-Der they were able to communicate as smoothly as possible.

The most important points were choosing a pilot area and finding a place as the recycling centre. Since BYSGM had extensive experience in such subjects and close contacts with Beyoğlu Municipality, enabling them to use the Smart City system, Beyoğlu district was chosen as the pilot area. Considering a country with an area of about 780.000 km^2 and a population of approximately 70.000.000 people choosing a pilot area was inevitable.

After agreeing on one of İstanbul's big districts, we needed to have a place, a recycling centre that would be suitable for storing the computers as well as working on them for installation, configuration, testing, etc.

This recycling centre had to satisfy some physical and operational criteria:

- **Size**: The recycling centre must be big enough. It will mainly serve two purposes. A repository to store the incoming PCs and a workplace so that at least 2 or 3 people can be working on PCs simultaneously and comfortably. The currently place is about 50 m^2, divided into two rooms a big one with enough tables and chairs and a small one that can be used as a repository.
- **Security**: No computer or peripheral can leave the recycling centre without explicit approval of local authorities. Once the incoming computers are categorised, configured and tested they are labelled as ready to deliver and responsible people are informed. The current centre is under control 7/24 by the municipality security officers.
- **Connectivity**: The centre must have a stable Internet connection so that technical people can test the finished computers' connectivity. Another use of such a connection is that it enables various installation and configuration scenarios. The current recycling centre has a 512 Kbps ADSL Internet connection provided by BYSGM.

Once these physical criteria had been satisfied the project an announcement was made in March, 2006 (on the project's website and its e-mail list) that the project started to accept computers.

The recycling centre's operation would not be fully functional without a call centre. In addition to the website announcements, e-mail list activity and personal e-mail communication, Faber-Der has setup a simple call-centre so that potential donators can call and speak to a project representative, ask their questions, describe the details of computers and talk about donation details.

2.2 Which Computers? What Kind of Hardware and Software?

One of the decisions to be made had to do with the types of hardware that the project requested and accepted. Neither did we want the recycling centre become a computer cemetery nor did we have the intention of drowning our potential non-technical individual donators in technically detailed specifications.

So we decided to publicly announce that we are accepting "complete and, no matter how slow, fully functional, working" PCs or monitors. This was an easy way to make people understand that they didn't have to care about the technical specifications of their hardware and we did not expect them to send us a single keyboard or an old video card (even though we tried to be as clear as possible we still had communication problems, e.g. people tried to donate physically broken computers, or single peripherals without computers such as old video cards, broken hard disks, etc.).

Another important point is that no matter what kind of computers we receive from donators, we are going to provide brand new keyboard, mouse and an Ethernets card for each PC. We are in the process of negotiating a hardware sponsorship to that will enable us for this. We believe this to be very important because keyboard and mouse are the most mechanical parts of a computer that are exposed to outside effects and get worn and dirty easily. An Ethernet card, on the other hand is important and defacto standard for Internet connection. These peripherals are relatively the cheaper parts of a computer setup so we do not have any trouble finding sponsorship for that.

Internally we categorised the computers into two, low-end and high-end. A high-end computer in the context of this project means at least a Pentium-III powered PC with a minimum of 128 MB RAM and a minimum storage capacity of 10 GB HDD. A low-end computer is one below the high-end threshold described above.

We made this categorisation because a state-of-the-art GNU/Linux distribution makes heavy use of CPU power as well as high amounts of RAM. Trying to install the same set of software on very different machines would render some of them unusable and very slow. The performance of the latest KDE/Gnome, OpenOffice.org, Firefox, Thunderbird, etc. maybe slow but acceptable on a P-III or a Celeron with moderate RAM however the same suite of software won't have any chance on a low-end PC.

Thus we plan to have two different GNU/Linux set of software for two types of hardware. Of course, we plan to provide a similar functionality using two alternative sets of software. Kids who are going to receive the computers must be able to open and manipulate rich formatted text documents, spreadsheets and presentations, browse Internet, read and send e-mail, play music and video files.

Another important point is to choose a suite of scientific, educational and recreational software as well as popular software development tools for kids. The most important criterion when choosing scientific, educational and recreational software is that they are in Turkish. Since these computers will go to young people from not very high socio-economical families we assume that they will not posses even a modest amount of English to use the software or read its manuals.

We are in the process of examining GNU/Linux distributions which focuses on education such as Edubuntu [6] and SkoleLinux [7] as well as the packages in math, science and games sections of Debian repository [8].

2.3 How to Tell Others About It and The Importance of Media

A social responsibility project cannot be realised if the "social" component is ignored. This means that other people must be made aware of the project and they must have solid information in order to believe its purpose.

Thus the communicative power of Internet and national mainstream media must be utilised.

During the initiation phase the project was in need of visual materials such as logos, banners, Flash banners, etc. This was handled by volunteering designers in Turkey who designed visual and graphical material to be used on the project website. Later these material were also used as advertisement on other websites, too.

Our experiences showed that limited online advertisement was not enough: Our first donator learned about the project not from one of the Internet sites but from a news that was published in one of the national newspapers.

So we decided to have professional help and contacted with one of the experienced journalists who also happened to be very actively engaged in free software and open source projects, promoting activities and making translations for popular free software.

According to the planning there will be a widespread campaign in June. News are going to be published in the mainstream media, important computer magazines targeting the individuals as well as enterprises.

3 Conclusion

As of this writing the project's popularity started to grow slowly. We have received a few computers from donators, all individual. We expect donations from institutions and enterprises since they generally have the capability of donating secondhand computers in large amounts and uniformly (having 10 computers with the very same and modest hardware is better than having 20 computers some very good, some very low-profile).

The reactions from society is very positive. Even though we still don't have a logistics sponsorship people either bring the computers directly to the recycling centre or call us so some project volunteers can transport the machine to the centre. After the big advertisement campaign in June we plan to search for a logistics sponsorship because people from other cities called us and told they want to donate their old computers.

The deadline of the project is September, 2006, so we have approximately 3.5 months to complete the project with current project partners.

The operation of the project until now had went without important problems and we plan to send computers in packs of 10 or 15. As soon as we have that minimum amount of computers ready they will be delivered to the families' houses.

We are going to evaluate the performance of the project near the end of this September and produce documents alongside with an evaluation report. We believe that the know-how to be gathered at the end is going to be very important to shed light upon other similar initiatives in different cities of Turkey and different parts of the world.

The success of this project is going to prove that helping kids and youngsters to become computer literate is feasible in short time and without rich funding and big sponsorships from government and private companies. It will also prove that free and open source software is mature, stable and reliable enough to take part as the main technical building block of such an important activity.

References

1. Çocuklara Bilgisayar Projesi `http://www.cocuklarabilgisayar.org` (last seen on 2006-05-21)
2. The Free Linux CD Project `http://www.freelinuxcd.org/` (last seen on 2006-05-21)
3. The 100$ Laptop Project `http://laptop.media.mit.edu/` (last seen on 2006-05-21)
4. Faber-Der `http://www.faber-der.org/` (last seen on 2006-05-21)
5. Fazlamesai.Net `http://fazlamesai.net` (last seen on 2006-05-21)
6. Edubuntu Project `http://www.edubuntu.org` (last seen on 2006-05-21)
7. SkoleLinux Project `http://www.skolelinux.org/portal/` (last seen on 2006-05-21)
8. Debian Software Packages `http://packages.debian.org/stable/`

FLOSS in Academic Computing Infrastructure: Experiences in Computer Science, Mathematics and Informatics Departments

Mehmet Gençer[1] and Chris Stephenson[1]

İstanbul Bilgi University, Department of Computer Science
Kurtuluş Deresi Cad. No:47 34440 Dolapdere, İstanbul, Turkey
{mgencer,cs}@cs.bilgi.edu.tr

Summary. This paper reviews experiences in and solutions for a computing infrastructure serving Computer Science, Mathematics and Business Informatics departments. Strategic choice of using open source was primarily motivated by lower costs and higher level of system integration the open source GNU/Linux systems provide. However use of OSS solutions has led to fortunate improvements in several other aspects which were not clearly predicted at the beginning. The overall improvements were: (1) a higher level of system integration, (2) lower maintenance costs, (3) lower hardware costs, (4) better experience in education, and (5) increased flexibility and capacity in team problem solving for system administration. Organizational politics aspects are also reviewed as our problem has been one of trying to move to FLOSS in an institution where decision making is not particularly transparent and entrenched interests are against change.

1 Introduction

The range of computing infrastructure needs of a Computer Science department tends to be wider than that of any other department and requires a departmental level of control over the computer systems. However, typically the provided computing infrastructure is developed and supported by an organization wide IT department whose target user prototype is considerably different from Computer Science students and staff. In this respect the management of computer systems in the University has traditionally been a field of struggle which is a mixture of practicality and politics.

Our problem has been one of trying to move to FLOSS in an institution where decision making is not particularly transparent and entrenched interests are against change.

This has involved a mixture of practical endeavor, politics and diplomacy. More than 5 years down the road we have some achievements but still much to do. In this short paper we try to draw some lessons from the experiences in our University.

2 Short history

Bilgi University Computer Science Department is a new department in a new University. The University was founded in 1996 and we took in our first students (we were then only a section of the Math department) in 1997.

In 1999, when the present Computer Science Department head asked how to upload some files for students on to the University's web site, he was told by the IT department,"Oh just send us the .doc files and they will be up on the site within a month or so".

Given there was no obvious immediate internal solution, the department simply started to use rented space owned by the department head, available on an external server. This at least established the principle of online course data for students.

Once this tradition had been established, over a couple of terms, it was time to try to move to become independent of the IT department in two respects: (1)to establish our own servers and (2)to gain control over our own computer labs. We acquired a third hand Pentium 233 from another department for our server, and, after a long period of negotiation, the necessary DNS entry and network connection. We also won acceptance that our practical classes would not just be allocated from the general pool of computer labs in the University, but that there would be two labs reserved to the CS department. This was essential if we were to start to change the software installed on those computers.

It was around this time that the department took the decision to set itself some strategic targets, and one of those was to take up and promote FLOSS software. It now seems incredible how unsure we were of ourselves in those days. We had very few students and no graduates yet.

FLOSS software was attractive for several reasons. Documentation for open source software is much more accessible compared to their proprietary counterparts. This situation would ease student requirements for self-study. Also they would be able to install these software in their home computers legally. In addition when instructors wish to use new software for their courses, these can be provided quickly -as no purchasing phase is necessary, not mentioning the immediate cost savings.

We set up our new server using Red Hat 7 and made all our lab machines dual boot. The server sat under the desk of the head of department. We also set ourselves two objectives: (a) uploading course material to the site should be very easy, and (b) students should see an immediate improvement in the service they got, in particular they should no longer have live in a diskette economy where they carried their files on diskettes and submitted their projects on diskettes.

The reason for (a) was that many of our full and part time teaching staff were surprisingly computer illiterate. If they were to use the system it had to be really easy; and for (b) was because we -correctly- anticipated resistance from students. We felt we needed a carrot to make the change to FLOSS more attractive.

After a number of failed attempts, the problem of easy upload was solved by mounting the web directories so academic users could simply save files to them. However this did not solve the problem of maintaining and updating links to new material on the system. This was solved with a three page PHP script, written over a few days by one person. This simply created a cascading menu system linking to any of a number of well used file types (pdf, html, txt) uploaded into the relevant directories. It was a zero maintenance, near to zero development cost primitive content management system. This was something that would only have been possible in a FLOSS environment. Despite the rather ugly interface, the system is still in use, virtually unaltered, today.

In the labs we made probably our biggest mistake. We made our labs Windows/Linux dual boot, and then tried to provide our improved computing environment to students in under both operating systems. This presented us with an development task we did not want and lacked the resources to do. We were only able to make progress when we dropped Windows and decided that we would only support Linux in our labs.

Once we had our server system up and running our next step was to acquire some more third hand computers and a switch, set up a gateway, a file server and mail services. We established mailman groups for all our courses and for the department. Finally, a move of campus two years later allowed us to make the case, showing what we had achieved with essentially free equipment, to finally spend some money getting a rack of new servers and our own independent Internet connection. The department now has 13 servers and around 200 desktops, all 100% FLOSS.

The only two items of software which incur any license cost in our system are a virus protection system for the few staff PCs that are still running Windows, and a Mat-lab system (which actually did not work despite the claimed compatibility and provider's all efforts).

3 Follow me Computing

Follow-Me-Computing was probably our best idea, but it was not easy to achieve. The idea was that we would virtualise student access to our computer resources. Anywhere the student went, a log in would be enough to get the student's own desktop set up, and personal files on the screen in front of them, whether they were in the University or at home. Our first mistake was to try to make a dual OS implementation. Providing the facilities we wanted under the restrictions of a proprietary operating system proved impossible. Only when we abandoned this course could we make progress.

With a basic Linux only system a high level of integration was achieved for users which we have failed to achieve with commercial operating systems. In what we call “follow-me-computing” environment, a user may work in any client computer by logging in with their password. The computer is integrated with the central home file-system via NFS(Network File System) so that the user will be greeted with their desktop and language settings, and can access their personal files ubiquitously so that they do not have to carry their files around on other media or be upset with language settings on their desktop. In addition SSH(Secure SHell) enables users to access these files with relative ease when using a computer outside the campus. This second type of access is not limited to Linux computers.

3.1 Current practice and technology

Linux systems allowed us to boot client computers over the network using PXE technology, instead of booting from local hard disks. There are various advantages of such a system: (1) no hard disk drives are necessary, (2) all client software is updated once, thus reducing maintenance overload substantially in addition to savings on hardware costs.

4 Advantages to user and to management

The advantages of our current system, compared with the environment provided to users of the proprietary system outside the CS department, to both teaching and student users are fairly obvious:

- Ease of access
- Permanent, backed up storage
- Large mail accounts
- Easy to set up mail groups
- Instant loading of web pages

This is to say nothing of the obvious educational advantages of a FLOSS environment.

There are other advantages that are less obvious. While the proprietary environment outside of CS has a considerable staff involved full time in looking after the system, cleaning viruses, reinstalling software, the CS systems require no specialized staff at all. The maintenance load is extremely low, even though it may be irksome when it is carried by someone who has another, full time, job.

Another important advantage is flexibility. When we found that rapidly rising student numbers meant our email system of student project submission was imposing an unacceptable load on our staff. Our FLOSS environment meant that developing a rough and ready course management system that provided project acceptance and assessment, with many important automated features, could be developed tested and put into production within three weeks of an inter term break.

4.1 Disadvantages of providing service without resources

There is a down side. If you make a virtue of cheapness in an environment where resources are limited, you will be expected to do things for nothing. Although we accepted hosting requests from other departments and programs who are interested in our extensive features, it turned out unmanageable and eventually we resorted to keeping a low profile on this front.

5 The spread of our influence and difficulties in extending the example

Now we are being asked to propose alternatives for other systems - even to prepare a plan for switching the whole university to open source - that is when our real problems are beginning.

Such a transition requires a concerted effort of university-wide IT department with all their organizational experience and the CS team with their FLOSS experience. However there is obvious -and understandable- fears in the IT team as they will be asked to move away from their core-competences (the proprietary systems). On the other hand lack of larger scale experience prevents the CS team from putting a sensible transition plan on the table to overcome these fears. Thus without a strategic decision to come from management, it does not seem possible to create a critical moment for such transition.

5.1 The danger of being cheap - library example

With the word of mouth going around about open source we were asked for opinion about software systems to be purchased for the University library. Despite our efforts to communicate that deployment of FLOSS systems will require none or little license costs but instead more application labor cost the perception turned out different. We were basically being asked to replace a 150.000 dollar system in 3 months, preferably with no penny put in for additional staffing. Considering the already existent shortage of experienced Linux system developers and administrators in the region this is a real trap! Thus we learned that communicating certain aspects were politically important although they may not seem that way to begin with.

6 Where next?

In today's world campus borders are quite obscure. Although classroom experience is irreplaceable, there is mounting pressure for doing most things through Internet. Thus the next step from the system and service development perspective would be providing students and staff the software environment and access methods for having computing experience similar to current in-campus arrangement from home, dorm or elsewhere.

Although we are still far from releasing a complete solution many technical aspects for this type of service is already resolved. On the server side, a combination of Kerberos and new generation NFS(NFSv4) provides secure access from off-campus Linux systems. On the client side we have developed enough experience and software to provide CD/DVD-bootable custom Linux systems[1]. Thus ideally a student or staff member can take a CD/DVD as their Linux system with all the software they need, and provided that they have a descent speed Internet connection they can work from any computer. Alternatively they could carry their files on a USB-disk type of removable storage if they do not want to rely on availability of Internet connection.

7 Conclusions

Increased problem solving capacity of our management team was most important result of switching to open source. This included improvement in administration of the system and development of solutions. As open source software packages and their documentation is highly accessible to those involved in management and development of systems, individuals or sub-teams were able to react promptly when problems are reported or requests for new features are received. Unlike proprietary systems which require previous exposure to software documentation, team members were able to fill positions whenever required. After initial period of adaptation to Linux systems, team response time and system error recovery time was considerably improved, although we lack clear logs suitable to present a net measure.

Technically, using Linux systems and FLOSS software enabled us to lower labor and time required for maintenance of our systems, lower hardware costs, and attain a higher level of system integration for better user experience.

However as far as Linux adoption in our wider organization is concerned, we found that after a long period of resistance, management switched to thinking that instant cost savings and painless transitions were possible. Meanwhile the IT department was just itching for us to fail. We now think that a longer transition must be targeted starting with establishing (1)a ground-up experience for the IT team, and (2)a realistic and wider perspective for the management. Thus we may conclude that one should not run into presenting technical aspects and cost-savings perspective without a comprehensive picture of cost of ownership and transition road map.

[1] See `https://babbage.cs.bilgi.edu.tr/svnrepos/acarix/trunk/`

A Questionnaire for F/OSS Training Needs Analysis

Onur İhsan Arsun and Selahattin Kuru

Informatics Research and Development Center, Isik University
Istanbul, Turkey
{arsun,kuru}@isikun.edu.tr

Summary. A training needs analysis questionnaire has been developed serve as an input for the F/OSS curriculum to be developed by the EC supported TOSSAD project. The questionnaire aims to quantitatively analyze the training needs of the European software industry in F/OSS domain. The questionnaire is designed a content-level approach. In this paper, we present the methodology and rationale behind the questionnaire.

1 Introduction

Within the EC supported TOSSAD[1] project, a training needs analysis questionnaire has been designed to evaluate the training needs in Free/Open Source Software to guide the development of the F/OSS curriculum. Clarifying the needs of the audiences for training is very important for the analysis phase for instructional systems design and is recognized as an integral part of any well-designed training program by academicians and training-theorists[2]. A training needs analysis is the method of determining if a training need exists and it does, what training is required to fill the gap. Especially if a new system is to be designed (as in this particular case), audiences must be analyzed to assure all their needs are met and their preferred modes of training are considered.

We have selected to use a questionnaire to achieve this goal, as questionnaires are primary instruments for quantitative analysis of this kind. We expect to harvest the following results after executing the questionnaire:

- Highlighting the subject matter needed to be covered in the F/OSS curriculum. More specifically, we aim to identify F/OSS topics where significant performance gaps or expertise shortages show, and the most important topics as regarded by the respondents.
- Identifying an accurate set of target groups for the developed F/OSS curriculum.

- Identifying the most appropriate training methods (classroom, e-learning, etc.) for each target group.

The questionnaire will be executed in 14 European countries and over 200 respondents. The target group are the software development companies of various sizes. The gathered data will compiled and analyzed in a special report, and will serve as a main input of quantitative data for the development of the F/OSS curriculum.

2 Methodology of the Questionnaire

As we have stated, needs analysis is an important phase of training systems design. There many methods for executing this analysis. Questionnaires are one of the most popular quantitative tools that result in in-depth data for analysis. In the design phase of the questionnaire, a study for determining the state-of -the-art for TNA questionnaires have been done. As a result of theses, the developed questionnaire bases itself on the following aspects[3]:

- Individual: The objective of individual analysis is to find out how well an employee is performing the the tasks that makes up his or her job.
- Organizational: The organizational investigation determines where training can and should be used within the organization. It focuses on the whole organization.
- Task-Oriented: The task-oriented analysis involves systematic collection of data related to a specific task. The purpose is to determine what an employee should be taught to perform a task at a desired level.

This aspect, called as content areas for the most basic structure of the document. However, these content areas that are used for many year prove to be insufficient in most cases. To broaden this approach, a novel framework called the Content-Levels Framework [4][5] have been introduced. This approach, based on the general systems theory that incorporates organizational, unit-specific, and individual levels of identifying training needs with a better focus on the relationships between the content and level type data. Our questionnaire employs this content-level framework in a fashion summarized in Table 1.

The questions that make up the questionnaire are created using this approach. The questions are designed to be simple, avoiding possible bias with a special consideration for the respondents' frame of reference.

The questionnaire uses the following question types:

- Contingency Questions: This allows simple branching throughout the questionnaire. Too many branches are avoided to keep the questionnaire as simple as possible.
- Matrix Questions: Questions that have identical answers are grouped for simplicity.

- Scaled Questions: A scale from 1 to 4 has been used for scaled questions.
- Closed-Ended Questions: Most of the questions are chosen to be closed ended for a more efficient quantitative analysis.

In designing the questionnaire, a number of F/OSS related questionnaires[6][7][8] and TNA related surveys [9] have been studied.

Level	**Content**		
	Individual	Task	Organizational
Individual	*What knowledge and skills do specific individuals need to learn for effective performance?*	*What are the knowledge and skill requirements for the accomplishment of specific tasks by an individual?*	*How do the goals of an individual effect or constrain performance?*
Unit	*What skill mix is needed for successful job performance within a given unit?*	*What activities, and technologies should be trained for effective task performance in a unit?*	*How do work group goals and culture effect or constrain performance?*
Organizational	*How does the training tie with strategic planing?*	*What are the technologies of the organization?*	*How does the organization prefer the training?*

3 The Questionnaire

TNA questionnaire for F/OSS has been divided into four sections.

Section 1: Organizational profile and IT strategy:This section gathers basic information from the respondent like organization type, size and its affiliation with the software industry. Also, data about F/OSS consideration in organization's IT strategy is collected.

Section 2: F/OSS Perception:This section gathers the perception level of F/OSS concepts by the organization. The F/OSS concepts include licensing, cost, and technical concepts such as reliability, security, documentation, etc.

Section 3: F/OSS experience: This section gathers the level of skill of organization in F/OSS and software development. Various types of software categories (operating systems, networking, data management, etc) is asked for rating from a scale of 1 to 4.

Section 4: Training needs in F/OSS:This section has two group of questions. The first group gathers the occurance frequency, performance gap, and needed skill force for each specific task. The second group of questions gathers which job title in the organization is required to have training of which of the specific task in the first group.

4 Conclusion and Further Work

The questionnaire will be executed in 14 European countries and over 200 respondents. The data gathered will be analyzed and will be presented in a future report. The results of this analysis will be a very significant contribution to the F/OSS community in Europe. The questionnaire's methodology and scale will provide important conclusion for training needs in F/OSS domain in Europe.

References

1. TOSSAD Description of Work, 2005
2. Moore, M. L., Dutton, P., "Training Needs Analysis: Review and Critique", The Academy of Management Review, Vol. 3., No. 3, pp. 532-545, Jul, 1978
3. McGehee, W., Thayer, P.W., Training in Business and Industry, John-Wiley and Sons, NY, 1961
4. Ostroff, C., Ford, J. K., "Assesing Training Needs: Critical Levels of Analysis", Training and Development in Organizations, Jossey-Bass, San Francisco, 1989.
5. Nelson, R., Whitener, E., Philcox, H., "The Assessment of End-User Training Needs", CACM 38(7), pp. 27-39, 1995
6. Stanford University Project on Economics of Open Source Software, `http://siepr.stanford.edu/programs/OpenSoftware_David/NSF-CISE-OpSourFinal.htm`, accessed May 2006.
7. Free/Libre and Open Source Software: Survey and Study, Final Report, `http://www.infonomics.nl/FLOSS/report/`, 2002
8. Lakhani K. R., Wolf, R. G., "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Projects", Perspectives on Open Source Software, MIT Sloan Working Paper, No. 4425-03, September 2005
9. Zaharias, P., et. al., "ICT Training Needs in SE Europe", Train-SEE project Internal Report, April 2002

List of Contributors