Overcoming barriers for successfully building an open source community

A case study of Open e-PRIOR,
an e-Procurement software project at the European Commission

By
Atle Johansen

Master Thesis in
Information Technology and Information Systems

Faculty of Economics and Social Sciences
University of Agder

Kristiansand, June 2012

Preface

This thesis presents research conducted during spring of 2012, as a final delivery for a master of information systems programme at the University of Agder in Kristiansand Norway.

The aim of the research was to explore barriers that need to be addressed for the Open e-PRIOR project to successfully build a sustainable community around the project and propose measures to be taken to overcome these barriers.

I would like to thank Didier Thunus at the European Commission (EC), who has been my personal coach for this thesis. Also thanks to Kelly Liljemo at the EC who has been of great assistance in both discussions and facilitation of executing this research. And of course thanks to the other people at the European Commission and the external experts who have provided data used in this thesis.

Thanks also to Mikael Snaprud and Tom Roar Eikebrokk, who have been my supervisors from the University of Agder and provided me with valuable feedback throughout the progression of my work.

Kristiansand, June 2012

Atle Johansen

Abstract

In 2009, the European Commission (EC) released an e-Procurement platform named e-PRIOR (*electronic PRocurement Invoicing and Ordering*). To support the objectives defined in 2010 e-Government action plan, an open source-version of e-PRIOR, called Open e-PRIOR, was also made available. Open e-PRIOR is a key enabler for secure e-Delivery in the e-Procurement domain, and the project aims to foster adoption of e-Procurement at European level. However, Open e-PRIOR is currently managed by the EC, not taking advantage of its open source nature. Besides, the EC also realized that the Open e-PRIOR project is of value both across sectors and across borders. Therefore, to improve the communication between users and developers, take advantage of the full potential of the software, and better allow sharing of experiences and collaboration, Open e-PRIOR is trying to build a healthy community around the project.

Through this study, barriers of successfully building an open source community are explored through a case study of the Open e-PRIOR project. This was done by analysing the Open e-PRIOR project through document analysis and interviews with the project stakeholders. To find possible measures to overcome the barriers, information on best practices was collected and a panel of experienced open source experts was interviewed. The identified barriers needing addressing were then discussed in detail against possible measures to be taken to overcome them.

The findings indicate among other things that the Open e-PRIOR project needs to make their processes transparent to enable the community to participate in the development. The project also needs a better way of collaborating on software development with the external community while lowering the technical barriers of entry.

Through applying the proposed measures to their F/OSS strategy, the Open e-PRIOR project and the EC will be better equipped to succeed in building a sustainable community to collaborate on developing the software further, finding bugs and new features, discuss architecture and security and improving documentation. Ultimately leading to a better solution for the benefit of all the stakeholders and meeting the goals of the EC of share and re-use of F/OSS.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

The concept of free/open source software (F/OSS) has been around for some time and is finding its way into more and more areas of application. A growing number of companies choose to release their software as open source, enabling users to engage in the participation of the development of the software through an open source community. More and more countries around the world are also having success with F/OSS in the public sector as an alternative to proprietary software. Examples can be found on all continents and in several areas in the public sector like educational, health and public administration to name some. Many projects are being initiated in the public sector, enabling sharing and re-use of F/OSS between public organizations and across borders.

These projects engage in a distributed development model, where software development is done through a developer community consisting of the users of the software. This way of organizing software development poses challenges of coordination and collaboration to succeed.

This study explores the challenges of F/OSS governance and community building in the context of the Open e-PRIOR project at the EC, and proposes suggestions for measures overcome the barriers involved in succeeding with building a community for F/OSS projects.

1.1. Document Purpose

This document is the final result of research conducted as part of a master thesis programme at the University of Agder. It has been carried out in collaboration with the European Commission, Directorate-General Informatics (DIGIT) unit B4.

The document audience is meant to be:

- The Open e-PRIOR management and development teams
- External examiners and supervisors at the University of Agder
- Other F/OSS project managers in the public sector

1.2. Background and Motivation

The Open e-PRIOR project would benefit from a clear strategy to further exploit the benefits of F/OSS according to a previous study of the project (Dar, Forsbakk, Johansen, & Liljemo, 2011). This study departs from this result and seeks to explore barriers that need to be addressed for the Open e-PRIOR project to successfully build a

sustainable community around the project and propose a set of measures which then can be systematically implemented by the Open e-PRIOR project and hopefully be of help to other open source projects.

1.3. Problem Statement

In line with the EC's policy of the sharing and re-use of F/OSS in Europe, the goal of Open e-PRIOR project is to build a sustainable community around the project, to enable adopting public organizations to participate in the further development of the software.

However the Open e-PRIOR project has not yet succeeded in doing this. The objective of this study is to remove barriers of achieving this goal.

As a result this research will propose to answer the following research questions:

- What possible barriers need to be addressed for building an open source community?
- What measures should be taken to overcome these barriers?

To answer the research questions, this study takes the perspective of change. Looking into barriers and enablers of F/OSS in the public sector and seeks to find solutions to these through a proposed change of processes, activities and technical solutions.

1.4. Scope

Some restrictions are made in order for the project to be feasible. The following lists the items that are regarded as out of scope.

- The assessment of the barriers and possible measures are focusing on the building of a community and related aspects, and activities such as conducting a cost/benefit analysis are left out of scope.
- In order for the Open e-PRIOR project to apply the measures proposed, clear implementation plans are needed. Such plans along with actual implementation are left to the EC to define and carry out, and are not in scope of this study.
- Similar to implementation, any operation or review is left to the EC and not in scope of this study.
- Any activities that span across the whole life-cycle of a project are also left out of scope. This includes aspects such as change management, training of staff, project management, etc.

1.5. Report Outline

The remainder of this report is organized like this:

Table 1.1: Thesis Outline

Chapter	Description
Chapter Chapter 2, Review of Literature	This section presents review of relevant literature on the subjects for this study.
Chapter Chapter 3, Research Method	Presentation of the methods applied in order to reach the project goal.
Chapter Chapter 4, Analysis and findings	Document analysis, interviews with Open e-PRIOR project stakeholders and interviews with F/OSS experts. Summary of the findings
Chapter Chapter 5, Discussion and Recommendations	Discussion of the identified barriers that need addressing and possible measures to overcome them.
Chapter Chapter 6, Conclusion and Limitations	Summarizes the discussions and recommendations.

Chapter 2

Review of Literature

The purpose of the literature review is to identify perspectives that can shed light on possible barriers of creating an open source community.

2.1. Free / Open Source Software (F/OSS)

Free and open-source software (F/OSS) is defined as software that is both free software and open source. In this context, free refers the software being licensed in a way that grants the user the freedom to use, copy, study and modify the software, rather than to the price of the software (Perens, 2007). Due to the nature of its liberal licensing, F/OSS' potential benefits has been increasingly recognized by both individuals and organizations. The development processes of F/OSS are fundamentally different from traditional software development models. The full source code is made available to the public and it is developed by a community of programmers collaborating over the internet. Participants are not directly compensated for their work and participation is voluntary (Hars & Ou, 2001).

2.2. F/OSS adoption in the public sector

In general terms, arguments exist that democratic governments have a public accountability to all of its citizens. The basic philosophy of F/OSS is congruent with this approach, and therefore F/OSS could benefit all.

In many countries, including China, India and Brazil, the use of F/OSS is widely established and many countries have passed, or are considering, laws encouraging F/OSS use (Applewhite, 2003). Germany in particular is seen as a leader, including the public sector, and also large businesses. Also in Australia and New Zealand successful F/OSS initiatives exist (Dougiamas & Taylor, 2003). The EU is increasingly adopting more F/OSS as awareness of its potential benefits is growing (Giacomo, Goedertier, Liljemo, Frade, & Hee, 2012)

Some suggests that although increasing availability of F/OSS applications on the market, the public sector is reluctant to investigate their potential use even though (Peeling & Satchell, 2001) indicate that flexibility that is offered by F/OSS could help to reduce some of the problems faced when replacing or integrating with old legacy systems considerably.

According to some studies (Giacomo, et al., 2012; OFE, 2011), public administrations tend to not trust a F/OSS solution to the same degree as a commercial one. If a F/OSS project is not backed by a commercial vendor, then public administrations tend to be reluctant to adopt the project due to security concerns. This sense of security by

obscurity has been shown to be an illusion, as one of the major benefits of F/OSS is the auditability made possible through the availability of the source code.

Some also point to the fact that F/OSS is not being as widely adopted in the public sectors as administrations would like because of 'comfort' factors. This is manifested in the fact that IT managers are sticking to software that they already know and disregard possibilities of innovation or change. McDonald et al. (2003) believes that these views are so engrained in the culture of public sectors that these institutions will continue acquiring enterprise-wide IS from traditional large vendors for the foreseeable future. Further explanation of the lack of open source adoption might be found in the field of institutional theory, which attends to the deeper aspects of social structure including how schemas, rules, norms and routines become established as authoritative guidelines for social behavior. DiMaggio & Powell (1983) states that mimetic isomorphism may be reflected on environmentally constructed uncertainties. When the goals are ambiguous, or when the environment creates symbolic uncertainty, organizations may model themselves on other organizations.

2.3. Governance of F/OSS projects

Early literature on open source software development focused mainly on individual based participation in F/OSS communities based on a community-founded governance model. However this model has later been adopted by organizational entities like public and private corporations, transnational organizations like the EC as well as nation states (O'Mahony, 2007). From this, a new governance model has emerged, where organizations release internally developed code to a public forum in hope of growing a community to improve and maintain the future code base (West & O'Mahony, 2005).

According to O'Mahony (2007), projects from such organizations are likely to face different challenges than community-founded projects, especially if they are trying to build a community around code that has already been developed. Not developing code and community in parallel can affect external developers' ability and motivation to contribute by making the learning curve too steep.

These post-hoc communities also face considerable problems of motivation and coordination. Since the external community members have not been part of building the system from the ground up, they might have trouble developing a sense of ownership for the technology.

As the external community and the organization may have different visions, goals and priorities, such projects also face a struggle for control over the future direction of the project. If the organization fails to relinquish some degree of control of the direction of the project, the result might resemble proprietary development conducted in a glass house. Outsiders observe but only participate at the margins of the organization's development (West & O'Mahony, 2005).

Further literature on this way of organizing software development can be found in Victor and Boynton (1998), who have coined the term 'co-configurative work' for these new, technology-enabled forms of 21st-century, post-bureaucratic, networked conglomerates, wherein organizations, users and other external actors share resources, and collaborate. Kristen Nygård (2012) defines related methods of co-operative work as

"participatory design", a term used in a variety of fields, including software design. Participatory design is a way of creating more responsive and appropriate environments for their inhabitants' and users' cultural, emotional, spiritual and practical needs.

Engeström (2004) integrated the concept of co-configuration work into activity theory. Co-configuration is a participatory model that is not confined to collaboration between professionals, and integrates users as active subjects. Users are active in the shaping and reshaping of products and eventually become experts themselves. F/OSS projects are prototypical examples of co-configurative work, including professionals, experts and users. Collaboration among people with such varying expertise necessitates a dynamic, dialogic relationship between multiple actors; it is a relationship characterized by collaborative and discursive construction of tasks (Engeström, 2004). Such groups are radically different from conventional teams or communities of practice (Lave and Wenger 1991; Nardi et al. 2000) in that membership at the periphery are fluid; current members can leave the community and new members can also join at any time.

Literature also goes on to point out the importance of scoping authority, access rights and participation, into different levels; i.e. users and developers. Criteria for membership should exist while the participants need to have the opportunity to assume greater responsibilities (Von Krogh, Spaeth, & Lakhani, 2003).

In another study, O'Mahony and West (2008) found that when organizations design a community, they are likely to offer only transparency, rather than to offer accessibility to external community members. They underline the presence of a control vs. growth tension. Organizations seeking to leverage the resources of communities to contribute to their bottom line, will try to maintain control over the community's strategic direction. However, organizations soon discovered that restricting access to community processes, limited their community's ability to attract new members and grow.

2.4. Collaboration

In open source community projects not all developers and users necessarily work on the project in the same location, but scattered over a wide area spanning often national borders and multiple time zones. Communication channels then require some kind of electronic means of communication. The most common forms of communication among open source developers and users are email and electronic mailing lists. For real time communication, an instant messaging method such as IRC (Internet Relay Chat) or similar is often used. Web forums are another common way for users to get help with problems they encounter when using an open source product. Also wikis are becoming more common as a communication medium for users and developers (Wikipedia, 2012).

In his studies of information quality in open source communities, Neus (2001) observed three typical issues of information quality : 1. Ignored altogether: There is little structure and everyone can post freely, making it difficult to obtain quality and re-use the information. 2. Over-controlled: All communication has to be approved by a leader or moderator before it is made available to the community. 3. Buried inside an unwieldy tool: The lively discussion that characterizes open source communities is squelched by a tool that was designed for information storage and retrieval, not for discourse and collaboration. This indicates the importance of finding a good balance between the ease

of collaborating on producing information, and the quality assurance of this information.

O'Mahony and West (2005) emphasized that enabling collaborative software development among organization's programmers and external community requires three types technical infrastructural tools that be common to the two groups. Source code repository for collaborating on the code, an issue tracking database for monitoring "to-do" lists and desired enhancements , and finally discussion groups often in the form of email lists i.e.

2.5. Anti-patterns

Anti-patterns offer a pragmatic way of describing problematic areas and also illustrating a way to solve them. It is unclear who first coined the term anti-patterns. In 1995 Andrew Koenig published a short article in the Journal of Object-Oriented Programming using the term, and in 1996 Michael Akroyd presented a paper at the Object World West Conference that documented harmful software constructs. Credit for promoting the term, however, must go to the authors of the anti-patterns book by Brown et al. (1998). They expanded the scope of anti-patterns to include software project management.

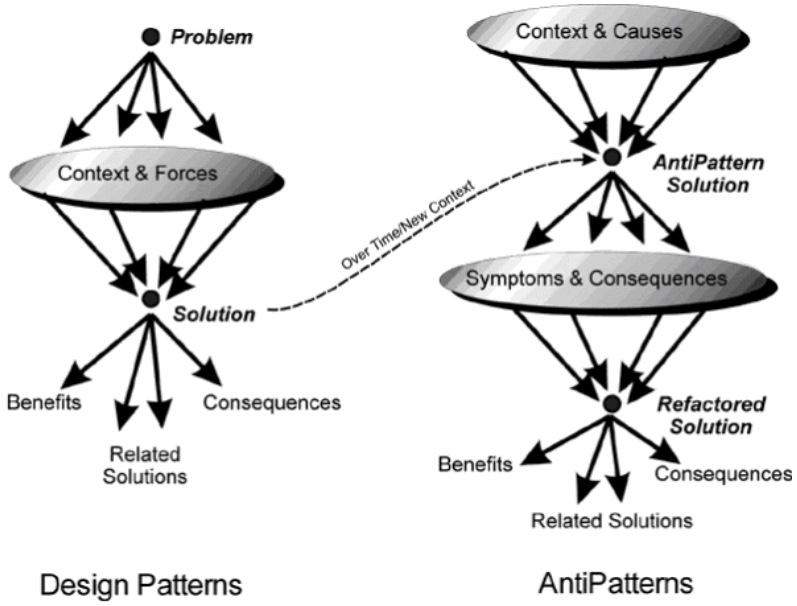

Figure 2.1: Design and Anti-pattern concept (Brown, et al., 1998)

An anti-pattern is a form of pattern that has two solutions. First a solution that describes commonly occurring outcome where negative consequences are present. This might be the result of lack of experience with a particular problem, or using an otherwise good pattern in the wrong context, resulting in negative consequences. To remedy this, the anti-pattern is documented, describing it in general, the primary forces in effect, the symptoms to help recognize it and the possible consequences resulting from it. It is then refactored into a version which describes how to change the anti-

pattern into a healthy solution (Brown, et al., 1998). Anti-patterns in F/OSS projects have some additional commonly occurring patterns mostly at a community level. These describe social and managerial issues regarding communication, interaction and coordination among developers and participants (Settas & Cerone, 2011).

In a talk at the MeeGo Conference (Neary, 2010), David Neary talked about Community Anti-patterns where he listed anti-patterns relevant for F/OSS community projects and possible cures/treatments for them. These anti-patterns are also listed at the Community Management Wiki and include anti-patterns like; Command and control, big show, bike shedding, black hole, water cooler, cookie licking and many more (CommunityManagementWiki, 2012).

Chapter 3

Research Method

A qualitative method was the natural choice for the project as this study is an exploratory investigation where the goal is to explore the context around the Open e-PRIOR project and its stakeholders, identifying possible barriers and measures for successfully building a F/OSS community.

To reach the goal of proposing measures to build a successful open source community for Open e-PRIOR, the study is broken down into several pieces. The study collects data from multiple sources which is analysed and discussed. Then derived from discussing the findings, a concluding set of measures for use in a F/OSS strategy is proposed.

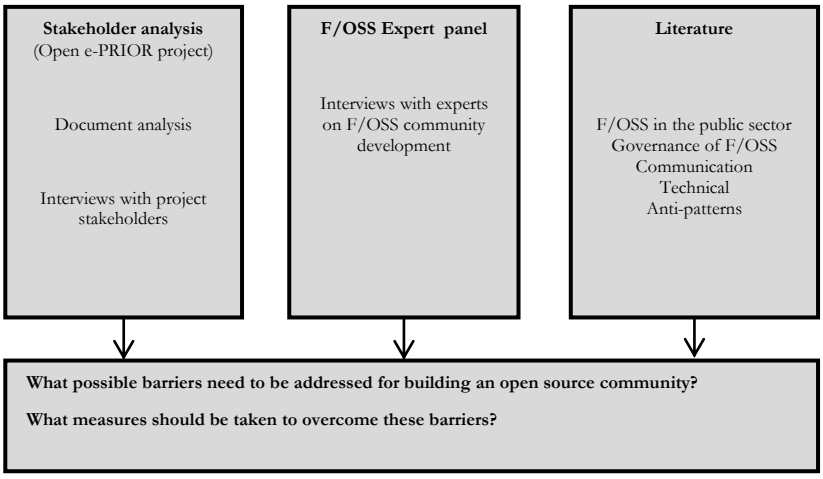

Figure 3.1: Building blocks of this study

3.1. Case description

3.1.1 Document analysis

As part of this study, several documents were provided by the EC, describing the complete project development plan (Thunus, 2012), detailed software architecture (D'Orazio, Fichera, Rodrigues, & Daniels, 2010) and other technical documentation for the Open e-PRIOR project.

These documents gave a detailed insight into the management of the e-Procurement section of DIGIT.B4 and how the Open e-PRIOR project is related to other on-going projects. Documents were also helpful in gaining a good understanding of the technical aspects of the project, its software architecture and how it connects to other systems like the PEPPOL network and customer back-office systems.

Through the document analysis, an understanding of the complexity of the whole solution was obtained and it was possible to have an impression of what skills are needed to pick up the project and get started on using it and developing it further.

Previous evaluations of different aspects of the Open e-PRIOR project (Dar, et al., 2011; Teinum, 2012) was also consulted. Using this previous analysis in the document analysis and collaboration with members of the Open e-PRIOR team at DIGIT.B4, a good picture of the state of the project eco-system was established. Then potential improvement areas were identified for further investigation.

3.1.2 Interviews with stakeholders of Open e-PRIOR

This phase of the project consisted of data gathering through interviews with different stakeholders of the Open e-PRIOR project. The interviews with this group were kind of inward-facing, looking at the Open e-PRIOR project from within, and the data gathered through the interviews aimed to supply more information to the insight and current state and help identify evidence of barriers and anti-patterns that cause counterproductive behaviour.

The group consisted of members of the ISA unit at the EC, which is providing the JoinUp platform, a forge for open communities within the EU, and of course members of the Open e-PRIOR project team.

In total 7 informants were invited to participate in this group. 4 ended up participating as the others did not have the opportunity because they were not available in the scheduled window. 1 of the informants participated through email and answered the questions in writing.

The interviews were carried out in a semi-structured fashion. The informants received an e-mail with an invitation to participate. Short description about the research project was also in the email. Interview dates were then scheduled via mail. In front of the interview the informants received an email (Annex I) with detailed information about the research project and the interview.

The informants were located in Belgium, and the interviews were conducted through Skype. This was done either by direct Skype call, or calling a landline phone. The interviews were, after getting the consent of the informant, recorded using software, and then transcribed for analysis.

An interview guide (Annex II) was used to help guide the interview through several different topics; An introduction of the informant, a series of questions on governance, followed by a section on communication, then a series of questions of technical nature.

In total 4.5 hours of speech was recorded during the interviews. The recordings were then deleted after transcription as stated in the information sent to informants.

3.1.3 Interviews with F/OSS experts

Also a panel of external F/OSS experts was interviewed. In this group, there were several prominent F/OSS experts, authors, community managers and other informants with a strong background in F/OSS and community building. All the informants were or had been involved in one or several successful F/OSS community projects. The data gathered through these interviews aimed to supply more information around measures to the potential improvement areas to overcome identified barriers.

In total 13 informants were invited to participate in interviews. 7 agreed to participate. 1 of these informants replied to the questions through email. 3 declined because of busy schedules while the remaining 3 never replied to invitations.

As most informants were spread all over locations in different cities in Norway, UK, France and the USA, these interviews were also conducted through Skype, either by direct Skype call, or calling a landline in the informants' country from Skype. These interviews were also recorded and then transcribed for analysis after getting the consent of the informant. One of the interviews was conducted in person in the office of that informant.

An interview guide (Annex II) was used for informants to help guide the interview through the different topics; an introduction of the informant, a series of questions on governance, followed by a section on communication, then a series of questions of technical nature.

In total 8 hours of speech was recorded during the interviews. The recordings were then deleted after transcription as stated in the information sent to informants.

3.2. Analysis of the data

The data gathered from document analysis and interviews was then analysed. The data consisted of collected documents and accompanying notes, recordings of the interviews with Open e-PRIOR stakeholders and expert panel, and notes taken during the interviews. The recordings were then transcribed into written language. Findings were then extracted from the document analysis and interviews. Two of the interviews were held in Norwegian and findings from these were freely translated into English. The findings were then structured and categorized according to the main findings with statements from the data supporting the finds.

3.2.1 Discussion and recommendations

The findings from the previous section were then discussed in the context of Open e-PRIOR and prior research, discussing the identified barriers that need attention and possible measures for overcoming them. Finally proposing a set of measures to help Open e-PRIOR successfully build a F/OSS community for the project.

Chapter 4

Analysis and findings

The purpose of this chapter is to describe the current state of the Open e-PRIOR project with regards to potential barriers that need addressing based on document analysis, online resources and interviews with relevant Open e-PRIOR stakeholders. Secondly, identify potential measures and best practices on F/OSS development and community building from a panel of experienced F/OSS experts in order to propose measures to be taken by the Open e-PRIOR project.

4.1. Document analysis

4.1.1 Adoption

The Open e-PRIOR software is intended for use by public administrations in the EU member states. Currently the install base for Open e-PRIOR is just a handful of installations (i.e. Greece, Spain, Norway and Ireland), even though the software has been downloaded by more.

However, this is not materializing in a corresponding growing of the current community. There is no tracking of who downloads the software, so the opportunity to follow up downloads is not present.

4.1.2 Distribution and release

The Open e-PRIOR project is made available to the public as Open Source so that public administrations in EU member states can freely use the software and become part of the community being built around the project. Distribution of the software is done through binary deliverables available for download through the projects site at JoinUp (2012). The source code of the current release is also available through this site, but the internal e-PRIOR repository that EC developers are working on is not made available to the community. Relevant changes in the e-PRIOR software used internally at the EC are pushed to the Open e-PRIOR source tree before each release. This prevents the community from monitoring the progress of the intermediate source code between releases. In practice, there is two development threads, one for the internal e-PRIOR and one for Open e-PRIOR, with the internal being the main development thread.

4.1.3 Community

JoinUp is a natural meeting place for the community, although according to the JoinUp site's member list, the community has less than 20 members as of today (JoinUp, 2012).

The site provides communication through mailing lists and forums, but the activity level on the site is rather low.

Open e-PRIOR has a role defined in an Open Source Gardener, which covers several open-source projects in the organization and works as a community facilitator. The responsibility of the facilitator is to tend to the project site at JoinUp and organizing the repository available to the public.

The JoinUp platform also provides a set of services for community development collaboration. The idea is that JoinUp is the hub for community activity in the project, providing access to source code, binary distributions, documentation and communication between developers. In collaboration terms, these features are not in active use by the community today, and serve as a one-way communication channel of distributing software releases and documentation from the Open e-PRIOR project to the public.

4.1.4 Governance

The roadmap for the project is documented in an internal document (Thunus, 2012) covering a detailed long term plan stretching into Q4 of 2013. Further in Q2 of 2012, the completion of an e-Procurement post-award feature is planned. This will provide electronic notifications of delivery and goods receipt. Also the implementation of the post-award process of e-Payment is planned to be delivered in Q2 of 2012. Finally, the pre-award process will be supported, starting in 2012 with e-Submission, continuing in 2013 with e-Award, and so on.

These planning and decision making processes of the EC is a slow and time consuming process and involve several stakeholders in different levels of the EC organization.

There is a subset of the development plan present on the JoinUp site reflecting the near-future plans for the project. But the rest of the plans are only available through internal documentation. The planning of new features is done internally within the EC by the Open e-PRIOR engineers and is this process is not visible to the external community.

Development model

The Open e-PRIOR project uses a tailored version the Rational Unified Process (RUP) for iterative development cycles. This process is defined through internal guidelines for RUP development in the EC.

Since the initial development of Open e-PRIOR was done solely by the EC, there is an established routine of working internally at the EC, having face-to-face meetings and discussions. The result of these meetings is then documented in internal documents.

The process is a closed process used internally in the EC organization and the development process exists more or less separately from the activity on the Open e-PRIOR JoinUp site. Some output from the internal process is published to the JoinUp site.

4.1.5 Technical

Technological solution

Technologically, Open e-PRIOR is an integration point with external web services (SOAP API) that are accessible for suppliers and an internal API for connecting to back-office systems optionally through an Enterprise Service Bus (ESB) or directly through the Spring integration framework. In addition, the platform has a built-in PEPPOL access point that enables it to connect to the PEPPOL network (D'Orazio, et al., 2010).

Software architecture

Open e-PRIOR is modularized into several different components, according to best practices in systems development. This part explains some of the most important components of the Open e-PRIOR software.

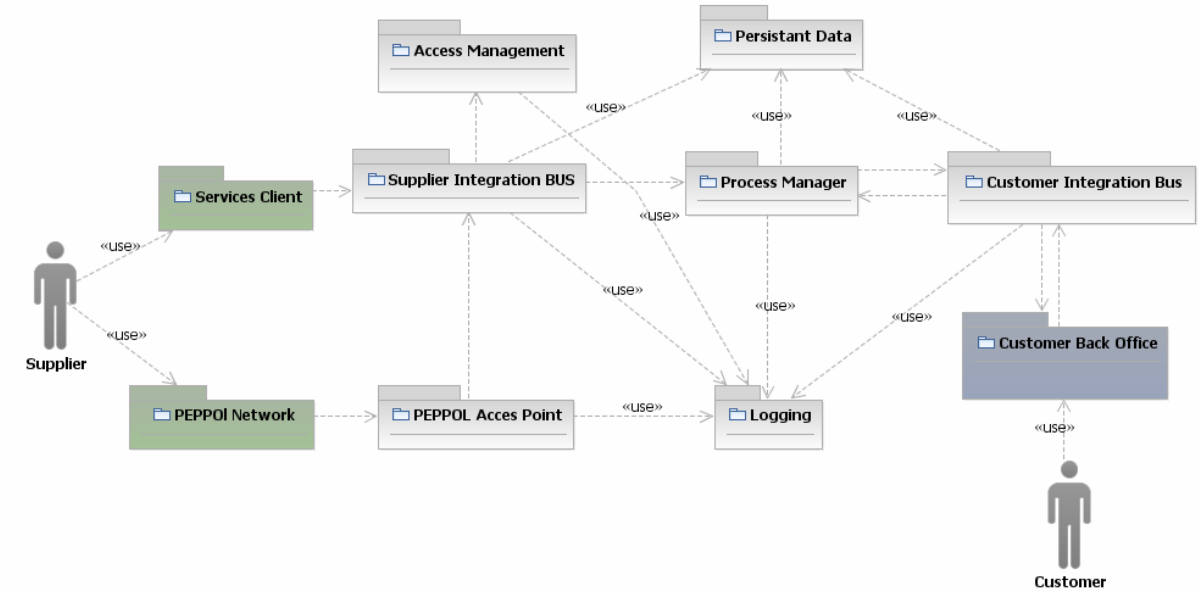

Figure 4.1: System Users and Main Packages (D'Orazio, et al., 2010)

Supplier Integration Bus or Spring Integration layer: Provides management of services in the SOA architecture. The component is the system entry point to services and calls the Process Manager that orchestrates services interactions. It includes functionalities like access management and logging. This component is reached from Internet and is also the entry point for messages coming from the PEPPOL Network.

Access Management: Covers authentication and authorization of all calls to the system services. The access management prevents unauthorized users from accessing certain services.

Logging: Provides logging services for each component in the system. This component logs information about who does what when and with which role. It also covers logging of specific business information during workflows' activities.

Process Manager: Does orchestration of services. The module provides infrastructure to manage implementations of workflows, correlation between message exchanges, asynchronous messaging, etc.

Customer Integration Bus: This module is a mirror of the Supplier Integration Bus, only for the customer side. It provides similar entry points to the system for the Customers and its back office systems.

Persistant Data: This module provides persistence of data to keep track of exchanged messages, information relative to Suppliers, configuration of back-office interfaces, internationalized codes and legal information.

Technical entry-level

Previous studies have indicated difficulties in getting the Open e-PRIOR software up and running quickly.

In January 2012 an evaluation of the software was conducted by a student at the University of Agder (Teinum, 2012). Though inexperienced in the field of e-Procurement systems, the student is a software developer by trade and his goal was to contribute to the project. During this evaluation he, however, experienced that getting Open e-PRIOR to work was a complex task, and expresses in his report that;

> "The initial goal of this project was to build an administration console. Because of problems encountered during the installation process, the scope has been narrowed down to only include the setup of a development environment, the documentation, and the testing capabilities. Thus, this report is about challenges faced during the phase of getting into the project, and proposed solutions to the encountered problems. "

With the narrowed scope, and consequent longer timeframe, the student was able to successfully have the software up and running in his test environment.

Another evaluation of deploying the Open e-PRIOR software in a test environment was completed by two software development companies in Norway in spring 2011. Both companies had, similar to the aforementioned student, difficulties getting the software up and running quickly. Within the given timeframe the software developers were only able to partly get the software to work (Dar, et al., 2011).

4.1.6 Documentation

Based on document analysis of the available documentation, it's clear that the software is intended for fairly technical personnel and requires some detailed knowledge about the deployment environment that Open e-PRIOR is intended for to deploy the software solution. This involves the JBoss application server, Service oriented architecture and RDBMS database servers.

Through analysing the available project documentation, the project site at JoinUp has all the relevant project documentation organized in rather comprehensive office

documents which makes it difficult for the user to find information quickly. This was also pointed out in both of the previous evaluations of Open e-PRIOR.

4.1.7 Collaboration

For organizing and planning of day-to-day development of Open e-PRIOR, the EC uses Atlassian JIRA (Atlassian, 2012a), which is a popular issue tracking and project management system. In addition to the internal JIRA system, the project has an issue tracker for the external communication of the project at JoinUp which is meant for use by the Open Source community for reporting bugs and improvement suggestions.

As source control, the Open e-PRIOR project uses Subversion, and the development on the code is done internally at the EC. The code is currently made available through the JoinUp platform, but as mentioned before is not easily accessible.

The project also uses an internal Confluence site. This is an enterprise wiki for content management which is used to collaborate on content within the project.

In addition to these tools, other tools are also used internally like RequisitePro and MS Project for project management, SoapUI for web service testing, TOAD for Oracle database management. Most of these are proprietary software and not much used in open source software development.

4.2. Interviews with Open e-PRIOR stakeholders

Table 4.1: List of interviewed Open e-PRIOR stakeholders

Stakeholder A
Stakeholder A works at the EC representing DIGIT covering multiple roles, including architect and project manager for projects related to e-procurement. The stakeholder has also worked as a developer and architect for the Open e-PRIOR project.
Stakeholder B
Stakeholder B is a consultant for a large consulting company working as a contractor for the ISA program for the EC. This is the program funding the Open e-PRIOR project. The stakeholder works on F/OSS related projects for ISA and is involved in open source research for the EC.
Stakeholder C
Stakeholder C works as a functionary and project manager for a number of actions for the EC and the ISA program, including the development of the community platform used by the Open e-PRIOR project; JoinUp.
Stakeholder D
Stakeholder D is working as an enterprise architect consultant at the EC.

4.2.1 Community

Most of the planning of new features and requirements is being controlled by the EC, leaving the community with limited influence over what new features to be implemented and the EC deciding all new features that go into the software. This is illustrated by a couple of statements from Stakeholder A.

> *"[…] if they want to commit something, it's important they do it according to some best practices which are Europe wise. And for users they have to understand that they cannot ask for features that are too much customized for their solution."*

> *"[…] the developers will contribute to commit the features that they have implemented according to the project plan, so we have a project plan in the beginning where there are some features that the external contributors can implement. They will implement that, commit on the SVN, and then we will test it internally. If everything is ok, we will release the new feature. So it will be release feature based, but the implementation of the features will be done outside, but under a common control."*

The stakeholder also described how the internal project manager and the development team was pretty much in control over managing the feature requirements for what gets implemented.

> *"If we talk about activities, what has to be done when we set up and we have to communicate with developers are: Clearly stating what is needed. So, on activities to collect all the requirements, to collect all the features that have to be implemented by our project manager […]"*

4.2.2 Governance

Continuing, the stakeholder illustrates how the planning and meeting activity was done internally at the EC.

> *"Yes, we did it, we did it internally. We didn't do it online, but internally in internal meetings. So we decided what had to be done, what are the features that have to be implemented. Several people was involved in this, but not online. I mean, because it was all internal in the initial phases."*

Also the stakeholder explained that it had been like this from the beginning when there were only EC developers on the project, so since everything was mostly done internally at the EC.

> *"[…] everything was almost internal here at the European Commission. Me and the people were working here, we didn't need forums and online tools to synchronize the development and things like that. So the open source community was used mostly for people who wanted to download and install the Open e-PRIOR product. So in that case we didn't need an online transparent process to be followed by other developers, because we were all here"*

This made the development process hidden from the external community. As the informant explains, the community site as JoinUp was only used for release distribution of downloads and installation documentation etc.

On the topic of support, there were some posts in the forums with information about answers to questions, but as Stakeholder A explained, support was not primarily run through public communication channel at JoinUp, but through email and a support inbox for use by the EC team. So there was no good re-use of support requests in form of an archive for the community.

> *"We should move to giving all the answers in the JoinUp community which we did for some users. So we did it, but sometimes it's easier to go through email and we start discussing through email. This is something that we should avoid because the answers we give to someone, if we publish it on the website, can be useful for other people who wants to download the application"*

4.2.3 Technical

Stakeholder A mentions that developing on the Open e-PRIOR software can be a complex technical endeavor, meaning that potential contributors must have a good understanding of many factors of e-commerce in addition to technical software development skills.

> *"[…] if people want to develop things for this is that they are aware of the interoperability factors, meaning that they will not be developing something that will only be usable for themselves, or only for their needs. So they have to listen to the complex environment that is all the member states of the European Union, so they have to be aware of all the standards that are in place, the best practice thing, and of course they must have clear developing skills in Java, and all the technologies that are used in the project."*

Focus is already on keeping a good design on the software architecture of the Open e-PRIOR software and some improvements have been done already, but there is still room for improving the modularization.

> *"Before, everything was together. It was kind of a not easy for people to understand where what is […] we separated the domains, and we are also trying to separate the layers, meaning that the integration part is being done in a separate package […]All the database classes are now grouped together. So it's clearer, but we still have to improve it. We still have to continue working on this…"*

4.2.4 Documentation

Documentation for installation and deployment of Open e-PRIOR software is organized in documents today. Keeping this documentation up to date can be a challenge, as Stakeholder A illustrates:

> *"[…] we develop something new, we have to update that document and this document. Maybe this document is a previous version, and the software is another version. So people say, okay, I read this in the document, but actually this is not that, so we have to say, yes ok, we forgot to update the documentation […]"*

4.2.5 Collaboration

The JoinUp platform has some collaboration tools included, mainly for communication purposes, like mailing lists and forums etc. The very simple features of the bug tracker do not include the necessary coordination features needed for community development.

Stakeholder B confirms this and admits that the JoinUp platform might not be at the forefront in collaboration tools.

> *"We won't say that JoinUp now joins the best of breed in these kinds of tools, so again I think, this is based on Drupal, and modules that have been thrown together. So the issue tracker is not the best issue tracker, the JIRA issue tracker is much better. What we see on JoinUp is that the issue tracker on Google code is much more user friendly, so that's unfortunate, but that's now the situation."*

4.3. Interviews with F/OSS expert panel

Table 4.2: List of interviewed F/OSS experts

Expert A
Expert A is on the board of directors for several open source software foundations and groups, like the Perl foundation and Python foundation to mention a few. The expert is also on the FLOSS foundation group, which is a meeting ground for leaders of open source software foundations with 150-200 members from about 100 different software foundations. The expert also has broad experience from open source project management and has worked as a contributing developer for various successful open source projects.
Expert B
Expert B is a community manager for a successful open source content management system with a large community of over 40 000 members. The role of this expert is to lead the community and leverage the community as capacity for innovation and as a marketing vector. The expert has experience with open source from school through a computer science and telecommunication education. The expert had worked with open source projects from the Linux kernels to webservers and web languages. The expert also communicates in several open source communities and is an active contributor to some of them
Expert C
Expert C works with consulting governments and corporations on open source policy and processes. This expert has founded several open source projects and has experience in project leading of these projects. The expert has also been a contributing developer to several successful open source projects, like Debian, and also been central in open source licensing.
Expert D
Expert D is an open source consultant that works for a company that consults the public and private sector in adopting open source software. The expert has worked in the ICT industry over a decade and with open source about half of that time, both in sales and project leadership.
Expert E
Expert E is head of technical group in a project involved in managing and developing a service portal for open source software for the public sector. The expert has experience with project management and development of open source content management systems.

Expert F
Expert F is a surface manager for an organization that provides advice on use, development and licensing of open source for the higher education sector. The expert has been involved in lots of open source projects for a number of years. This expert has experience as a contributing developer on some Java based open source projects, and is involved in research of open source communities and advising open source projects that develop software on how to build their communities.

4.3.1 Adoption

Several experts in the group weigh the importance of boundary spanning when raising awareness of an open source project, a well-known technique where you find other communities that have similar or overlapping areas of interest.

Expert F explains:

> *"[…] there are usually mailing lists for communities, other communities. So you need to be aware of what other systems there are in this sphere that you're operating in, and what communities are there out there already. And then make them aware of what you do."*

Expert C pointed out that *"Well I think that all the ways that companies marked projects are applicable to open source projects". With this the expert meant that companies have a better ability to market themselves to their target groups, for example simple and readable websites that have scoped information that is relevant to the visitors. Separating the information for users from developers for example.* The same expert goes on to say that open source projects often don't have focus on anyone but the people that are already a member of the community and is often characterized by being by developers for developers only.

> *"Well, I think there is a very big problem in open source in that maybe the open source projects are inward-facing. They do their development for each other, they feel that their users are each other, and they don't really care that much about an outside community."*

4.3.2 Distribution and release

The experts' answers indicates that how to organize source code and repository-trees vary a lot on different factors from project to project. Source management strategies in open source have a lot in common with proprietary software projects and best practices apply to them both in general.

> *"I think this is also dependent of the size of the software and the size of the community and the frequency of releases. But in general, the best practice in software development is also valid here."*

Keeping the code visible is important for the general public to be able to inspect it at will. Expert A illustrates this with in a quick summary. *"I'll do the quick summary. Keep the code visible, keep the community practices visible, behaviors visible […]".*

Expert F also points this out as having quick access to the source code increases the motivation of potential new members to contribute.

"[…] additional benefits of that is that you always have a current state of your code which can be examined by people who are new to your project. So if I am new to a certain project and would like to try it out, being able to just download it or just check out the source code and being able to build it just by pressing one button. That would increase the likelihood that I will be interested in a project."

Expert C points out that even though you have two branches of source code, having only one thread of development will reduce the amount of work needed to merge between the two as they diverge.

"[…]once you have two different threads of development, you always have this problem that they diverge and the problem of getting them back together is a very, very difficult, time consuming and expensive. So once in a while I think just taking from the community version, fixing bugs, putting the bug fixes back into the community version making sure they are in the certified version as well, and you know, releasing that and six months later going back to the community for a new version."

The same expert also explains why Git, which is a very popular distributed revision control and source code management system, is good for decentralizing community development, creating more flexibility for contributors. Commits to the external branches can then be pulled back in to the central repository by a ruling body for the project that decides what to take on in the next version.

"I would base any new project on Git, because of that, and you know, let people do their work and not concern myself that there are little forks out there. I think that having a central repository that gets pulled into though, is absolutely critical. And that having that operated by high status developers so that all the other developers are watching that one and in general doing their development in a way that it's intended to be merged with that one. That's important."

4.3.3 Community

Experts also give answers that play on the importance of rewarding the community to keep motivation up and engage the community members, promoting participation in the project.

Expert B revisits the topic of transparency as a motivational tool to engage the community and promote contribution by giving access to the team of engineers in the organization governing the open source community.

"[…] the more visible your community is as a community vector, the higher the engagement of the people in the community. It's motivating to see experts from […] highly recognized to be working in open mode, you can learn from them. You are kind of aspired and inspired by these guys. It's really a motivational driver"

Expert F weights the importance of being friendly to community members, especially newcomers to encourage them to stick around and get more involved in the community, and ultimately become more valuable contributors.

"And I think that in the way that you engage with people that are new to a project you can either encourage them or discourage them from becoming more involved. […] I

think it's important to even if someone asks a very simple question that have already been answered earlier or that's already on the website. By being friendly to them and pointing them to the right resource rather than, you know, slacking them off or something, I think that's very important, so you need to be friendly."

Expert C emphasized that community developers should be able to do whatever they want and take the project in any direction they want, even if it is contradictory to the direction of the corporation that runs the project. He argues that by doing this, it will motivate community members and they come up with new ideas that the corporation would not expect, important to the community members.

"The developers must be able to take the project in the direction they wish, especially when that is not the direction desired by the corporation. And the reason is that you get very serendipitous results that way. You get things that the company did not expect important, and indeed are. And what you get from this really is that governance of developers is the enemy of innovation"

Expert F focused on the importance of being welcoming to new members because of the barrier some will have to take that first step and start their first contribution.

"[…] for people that are for the first time engaging with your project, sometimes that's a bit scary for people. Not everyone is used to working in a public environment, so for some people there is a threshold that they have to overcome before they send the first email to a mailing list, so you might be even a bit more friendly that you would be in other circumstances"

Several other experts also points out the need to be including and welcoming to community members and not letting them become 2nd class citizens. Expert A points out an example where the organization behind the project is preventing community members from working on the central repository, and where there is no other branches or open repositories available for the community members.

"[…] makes the volunteers and the community developers feel like 2nd class citizens and that really kills motivation. If you just have one central repository and everyone has to work on that central repository, but the community volunteers don't have any access to commit to it, then it doesn't make them feel like they're really part of the project."

The same expert also describes another situation where the engineering team of the organization is using a private bug-tracker that excludes the community members from taking part in the development process fully.

"Well the biggest problem is transparency. You know for every bug that is in the private bug tracker, you're not getting public contributions to that bug so. The internal team is totally responsible for it. And again it's like that 2nd class citizen thing"

Another Expert, D, raises concerns for when an organization takes too much control over the development of a project and cares little about further development and suggestions coming from the community. He references a good example from a project.

"The supplier has been too strong in the development. So suggestions and development has been neglected by the supplier so that the community is also neglected to some degree.

The supplier feels too much responsibility for the software solution. And then it's not very pleasant to contribute to a project where you feel your suggestions are neglected."

There were answers from several experts that emphasize the need for a community project to have a clear mission statement that clearly states the goal of the project. This way no one is in doubt about the direction of the project, and this will increase motivation among the members of the community. But as expert B pointed out as very important is that if there is a strong organization or company running the project, the mission statement must be unified and commonly interesting for both the community and the organization.

"In order for a community to be motivated, to be united, there needs to be a strong, we call that mission statement. […] And with the mission statement, you need to have the community members adhere to it, and also the company needs to adhere to it. So basically it's a very classical social pattern that you can find in history many times, also in kind of war situations. People are united behind one struggle, one cause, one fight. And this mission statement here is the case that everyone is chasing, aiming at; the community members, and also the company"

4.3.4 Governance

A lot of the experts mentioned that it's very important that the processes in an open source project are transparent to the community for the reason of being able to engage new members in distributed development in a community. They also emphasize that the whole process needs to be transparent, meaning all planning and decision-making in addition to the development activities and distribution processes. Expert F says:

"[…] that's very important. It's one of the main aspects of open communities that we preach actually. If you make your project transparent, that would mean that people who are not part of the project can see everything that happens and how decisions are made. And since an important goal of projects usually is to increase uptake and to increase level of engagement from users and development, being transparent is one of the key aspects of engaging new users. As long as people are not able to see what's going on, and how decisions are made for example, or how processes are run, it's much more difficult for them to become involved."

Expert A indicated that the lack of transparency would demotivate potential members from engaging in the community: *"so I would say for running a project that's really, really important. It's extremely demotivating to come into a project where you have no idea why things are the way they are"*. Expert B also mentioned the importance of process transparency and states that the transparency would additionally also be important to generate trust and the adoption of the open source software.

"[…] it is one of the cornerstones of open source development […] also development process, but also decision processes and governance. So I think it is key to generate trust, creating a strong feeling of belonging in the community, and in terms of adoption of the product and the project and engagement."

Expert C explained that he had never seen an open source project succeed, that tries to keep their development separate from its community and explained why:

"[…] there are companies that have tried to keep their development separate from their community, and they always fail. And the reason is because they go out and do months and months of development, and meanwhile the community is going in a different direction and they can never merge the two pieces together."

4.3.5 Technical

Common for informants in this group was the emphasis on having as low entry-barriers as possible for both users and developers. Most important is the entry-barriers of users. Expert A put it into words quite well:

"[…] it's important for both. It's especially important for users. Like if your users can't figure out how to get started, you'll bounce quite a lot of them. They'll come, they'll try out your project a little bit, but then they'll just give up and go away. And they won't always tell you what frustrated them, and that's what's hard to improve is if they aren't leaving any trace of why they didn't use your software. Sometimes they won't even bother complaining, it's just too frustrating. Developers have a higher tolerance for pain. If you make it really easy for users, you can require a certain level of knowledge for developers to really get involved. It is helpful if you can have the easy, how to compile instructions. You know, the easy how to report a bug instructions. Those are really good sort of easy on-ramps but it's ok if they do have to spend a few weeks reading some more technical documentation before they can really make a good patch to your software"

Expert B went on to focus more on the development side of things, and the importance of lowering the technical barriers for developers seeking to extend and change the software. In the project that the expert is currently working, they are doing exactly this by rewriting the software, separating the code in modules and creating clear layers in the software architecture:

"[…] it's a key requirement. And it needs to be reflected in the way the project is built. That's what we're doing today […]. We are rewriting the API so it's easier for anyone to learn more quickly. I think it's VERY important."

Expert C also pointed out that modularizing and layering the architecture of the software is important. So a developer that is starting out as a contributor doesn't have to learn the whole system, but can focus only on the part that he wants to make changes to:*"Ok, so the low learning curves. So what is the lowest learning curve necessary to start using this project? And then what is the learning curve necessary to be a developer on some corner of the project?"*

Another scenario that an Expert D mentioned was that with some open source projects you're often left to yourself for installation and deployment. And automatic setup and configuration is important to counteract this entry-barrier.

"[...]if an open source project is just dumped online, then you're 100% left to yourself to install it, and then it's critical that it's easy to do an install. Ideally you should be spared doing compilations and lots of technical configurations of configuration files, so ideally this should be a streamlined wizard. This should be the goal for anyone who runs an open source project."

4.3.6 Documentation

Expert D illustrates the benefit of having documentation in a collaborative form, like a wiki, compared to documents that follow the software. *"[…] it can of course follow the software, but it will quickly become outdated, the wiki is always updated with documentation on installation, use and everything."*

Expert A points out that even though a wiki can be useful, there needs to be active content management. *"Wikis often run to garbage after a while, you really have to have periodic cleanup processes, to clear out all the wiki pages, otherwise they end up being largely junk.".*

4.3.7 Collaboration

There was a broad agreement among experts that the importance of the availability of well-functioning collaboration tools for the community is essential for an open source project. According to Expert B, the most important tools mentioned were bug-tracker with simple project management tools and forums, or mailing lists. And the expert also emphasizes that the tools and processes used need to be intuitive to the developer:

> *"[…] basically an issue tracker with light project management tools is what people need. In many communities people don't have loads of time to spend on the project; it's maybe an over-generalization after my experience with […], where most members contribute to the project after hours in the night and on weekends. So people need to be able to jump into the processes and understand them in like 10 minutes. […] So the widespread tools for that is a bug tracker or an issue tracker in general with simple product management tools […] And then you mostly see forums and mailing lists. Mailing lists I think are good for small team collaborations, like max 20, and then it gets a mess."*

This expert also points out the importance of also having good processes that are supported by these tools. It's claimed that the tools themselves have limited value without well-defined workflows that involves the people and tasks at hand.

> *"Most important stuff are workflows and processes. So the first work is to define the processes and find the workflows. Which group of persons is going to work with what other group of persons. What are the steps involved in approval, or review etc. and then how do you support the processes you have defined with proper tools."*

Expert A points to the fact that the bug-tracker is usually the first tool new users meet when they start interacting with the project.

> *"Yep, those are essential. I mean you can make your choices, not every project has a wiki. The bug tracker, having a public bug tracker that is easy to submit to is very important. New contributors often come in that way, you know, the first thing they do is just report a bug and then someone responds and shows them how to solve their problem and they end up seeing an improvement going into the software, that can be a good motivation to go on to like the next step where they get deeper and deeper into the project"*

This expert also mentions the importance of them being easy to use and indicates that familiarity is important when starting to use new tools by drawing on the example of sending an email.

"I would say anything that slows the user down or distracts the user is going to be a big hindrance. That's very generic but basically making it possible to get the most value out of the tool as quickly as possible. So whether that's finding the answer to your question very quickly, or submitting a bug very quickly, I think that's why email has become such a popular standard, because although the email clients aren't very good, submitting an email message they don't have to spend a week learning the system"

The interview with Expert C also mentions the need for facilitation when new users are involved and goes on to suggest that if there are a high population of novice users, other more familiar tools can be evaluated.

"[…] that especially for what I would call the naive user, the one that isn't really a programmer, that they have to be led into this system. There has to be some degree of facilitation on the inside to use their input effectively. They won't write a good bug report on their own, and that thus you might want to have channels that they are more familiar with, like chat or telephone"

This is backed up by Expert F who says that it's important with some facilitation when employing these tools to help the new members use them and help both them and the community take advantage of their benefits.

"For example if a user posts a question to the list and it turns out it's a bug, ask them to fill it out in the issue tracker themselves, rather than doing that for them. So engage them and try to, just as an example, involve them by pointing them to the tools that you have. And only set up tools that you use. If you're not using a tool, don't set it up."

Experts mentioned in a case of a vendor or supplier backed community the importance of using the same tools for the community as for the internal engineers that are working on an open source project. Expert F said that there are only a handful of reasons why information should be hidden from the public community, like reporting security issues and discussions about people. But in these cases it's better to have features in the tool-chain that enables hiding certain information from the public and only give privileged team members access, than to have a whole separate toolset for this purpose.

"Well there could be some issues with security sometimes, for examples we have private communication lists for if you want to report vulnerabilities in the code or something like that, or if you have discussions about people. You want to vote in a new committer for example or in the community, you will have that discussion in a private environment so people can speak freely. But that are probably the only reasons where you have private communications and I think all the rest should be as public as possible"

Expert B had also seen this problem before, and was in the process of changing a big project to use the same tools for engineers and community members.

"[…] the need for collaboration between […] engineering and the community is also increasing consequently. What we've set up at the moment, is that we're going to have the community to use the exact same tools as our engineering team. Meaning; the engineering team is using an issue tracker and Github. Then everyone has his own ID at Github, and that's the baseline. Then they have Campfire as an online chat, they have Review Board as a code review tool. This is for advanced, this is daily 8 hours a day work. For community needs, we just need to have an issue tracker and Github,

period. So then that's going to be JIRA by the way, and Github, and these 2 tools are going to be used by both […] engineering and the community."

Expert C had an interesting comment about competing tools, that if you want a group of people to start using a certain toolset, it needs to be better than the other toolset it's competing against.

"[…] when you want to attract those people to […] rather than JIRA and Subversion etc. you have to actually provide them with features that are more desirable than the tools that they are currently running"

Expert D points out a scenario where keeping things separate would make sense. If the community members are mostly non-technical users, having them use the same tools as developing engineers would possibly generate a lot of noise for the developers, so assigning a super user to filter the content from the community tool over to the private one.

"[…] you can think that if you would give the users access to the community site, it could generate a lot of unnecessary noise because the users don't have the competence. Like is it me that has an error or is it the software? Or that the users don't know how to use features. Then such private solutions would have some value, where you would have a super-user that report this back to the community. But yes, it can vary […] if you have sufficient technical skills; you understand what you can and cannot ask, and what you should try first."

4.4. Summary of findings

The following table summarizes the identified barriers that need addressing and supporting findings from the data collected from the document analysis and interviews with project stakeholder group.

Table 4.3: Summary of findings from stakeholder group

Categories	**Barriers**	**Findings** **(D: Document analysis / I: Interview with Stakeholder)**
Adoption	Few public administrations adopt the software.	D:Install base for Open e-PRIOR is just a handful of installations
Distribution and release	Source code visibility is low	D: The only link to source code of Open e-PRIOR found on the JoinUp site is in a thread in the forum D:JoinUp features are not in active use by the community today, and serve as a one-way communication channel of documentation and source code
Community	Community's influence on project is low	*I:" they have to understand that they cannot ask for features that are too much customized for their solution"* *I:"So, on activities to collect all the requirements, to collect all the features that has to be implemented by our project manager"*

	Recruitment to the developer community is low	D:Downloads not materializing in a corresponding growing of the current community D:Community has less than 20 members D:Activity level on the JoinUp site is rather low
Governance	Development process is hidden from community	D:Detailed project plan for the project is documented in an internal document D:Development process is a closed process used internally in the EC organization and the development process exists more or less separately from the activity on the Open e-PRIOR JoinUp site *I:"We didn't do it online, but internally in internal meetings"* *I:"we didn't need an online transparent process to be followed by other developers, because we were all here"* *I:"sometimes it's easier to go through email and we start discussing through email"*
	Decision making process of EC is slow	D: planning and decision making processes of the EC is a slow
Documentation	Access to documentation is unwieldy	D:Project documentation organized in rather comprehensive office documents *I: "[…] we develop something new, we have to update that document and this document. Maybe this document is a previous version, and the software is another version."*
Technical	Technical entry-level is high for beginners	D:Student struggled to get Open e-PRIOR working despite being software developer by trade D:These [companies] also had difficulties getting everything up and running *I: "All the database classes are now grouped together. So it's clearer, but we still have to improve it"* *I: "[..] they have to listen to the complex environment that is all the member states of the European Union, so they have to be aware of all the standards that are in place."*
Collaboration	Collaborative tools for community development is insufficient	*I:"We won't say that JoinUp now joins the best of breed in these kinds of tools, so again I think, this is based on Drupal, and modules that have been thrown together"*

The following table summarizes the potential measures to overcome the barriers and supporting findings from the interviews with the F/OSS expert panel.

Table 4.4: Summary of findings with F/OSS expert panel

Categories	Measures	Findings (I: Interviews with F/OSS expert)
Adoption	Learn from commercial marketing	*I:"[…] the ways that companies marked projects are applicable to open source projects"*
	Raise awareness with boundary spanning	*I:" […] you need to be aware of what other systems there are in this sphere that you're operating in, and what communities are there out there already. And then make them aware of what you do"*
Distribution and release	Make source code more accessible	*I:" […]Keep the code visible."* *I:" […] always have a current state of your code which can be examined by people who are new to your project[…]"*
	Organize source code according to best practices	*I:" […] once you have two different threads of development, you always have this problem that they diverge and the problem of getting them back together is a very, very difficult, time consuming and expensive"* *I:" […] base any new project on Git, because of that, and you know, let people do their work and not concern myself that there are little forks out there. I think that having a central repository that gets pulled into though, is absolutely critical"* *I:" But in general, the best practice in software development is also valid here"*
Community	Let community have influence on the project	*I:" […] developers must be able to take the project in the direction they wish […]"* *I:"So suggestions and development has been neglected by the supplier so that the community is also neglected to some degree. The supplier feels too much responsibility for the software solution"*
	A clear unified mission statement for organization and community	*I:" […] for a community to be motivated, to be united, there needs to be a strong, we call that mission statement […] And with the mission statement, you need to have the community members adhere to it, and also the company needs to adhere to it"*
	Rewarded and motivate community members to get engaged and join	*I:" […] the more visible your community is as a community vector, the higher the engagement of the people in the community.[..] It's really a motivational driver"*
	Don't treat members like 2nd class citizens	*I:" […]community developers feel like 2nd class citizens and that really kills motivation."* *I:" The internal team is totally responsible for it. And again it's like that 2nd class citizen thing"*
	Be welcoming to new members	*I:" […] the way that you engage with people that are new to a project you can either encourage them or discourage them from becoming more involved […] that's very important, so you need to*

		be friendly" *I: "for people that are for the first time engaging with your project, sometimes that's a bit scary for people […]might be even a bit more friendly that you would be in other circumstances"*
Governance	Open up process and make visible to community	*I:" […] that's very important. It's one of the main aspects of open communities that we preach actually […]"* *I:" I would say for running a project that's really, really important. It's extremely demotivating to come into a project where you have no idea why things are the way they are"* *I:" […] it is one of the cornerstones of open source development […] also development process, but also decision processes and governance."* *I:"[…] there are companies that have tried to keep their development separate from their community, and they always fail"*
Documentation	Make documentation easily accessible	*I: "[…] the wiki is always updated with documentation on installation, use and everything."* *I: "Wikis often run to garbage after a while, you really have to have periodic cleanup processes"*
Technical	Lower entry-barriers as much as possible for all roles	*I: "It's especially important for users. Like if your users can't figure out how to get started, you'll bounce quite a lot of them"* *I:" […] it's a key requirement. And it needs to be reflected in the way the project is built."* *I:" […] low learning curves. So what is the lowest learning curve necessary to start using this project. And then what is the learning curve necessary to be a developer on some corner the project."* *I:" [...] if an open source project is just dumped online, then you're 100% left to yourself to install it, and then it's critical that it's easy to do an install."*
Collaboration	Provide well-functioning collaboration tools	*I:" […] an issue tracker with light project management tools is what people need."* *I:" Most important stuff are workflows and processes […]then how do you support the processes you have defined with proper tools"* *I:" Yep, those are essential […] having a public bug tracker that is easy to submit to is very important. New contributors often come in that way."* *I:" […] anything that slows the user down or distracts the user is going to be a big hindrance.[…] making it possible to get the most value out of the tool as quickly as possible"* *I:" […] the naive user, the one that isn't really a programmer, that they have to be led into this system. There has to be some degree of facilitation"*

		I:" […] try to involve them by pointing them to the tools that you have. And only set up tools that you use. If you're not using a tool, don't set it up."
	Unify tool-chain for community and vendor's engineering team	*I:" […] the need for collaboration between […] engineering and the community is also increasing consequently. What we've set up at the moment, is that we're going to have the community to use the exact same tools as our engineering team."* *I:" […] when you want to attract those people to […] have to actually provide them with features that are more desirable than the tools that they are currently running."* *I:" […] it could generate a lot of unnecessary noise […] But yes, it can vary […] if you have sufficient technical skills"*

Chapter 5

Discussion and recommendations

In this section the challenges identified through the analysis is discussed against relevant literature on the subject and the possible measures for overcoming them is recommended.

5.1. Adoption

Even though there is an increased awareness of the benefits of F/OSS in the public sector and governments are considering laws to promote its use, there are still challenges. Despite the expected benefits, the reluctance described by Peeling and Satchell (2001) seems to prevail, exemplified by low rate of adoption of Open e-PRIOR. This is likely to change as more governments pass laws encouraging the use of F/OSS (Applewhite, 2003). The persevering reluctance of F/OSS adoption in the public sector agrees with the suggestions of McDonald et al. (2003), who believes that IT managers stick to software they already know. The behaviour and perceived uncertainties these managers have of adopting F/OSS is backed up by institutional theory. DiMaggio & Powell (1983) describes that organizations will, when faced with ambiguous goals and an uncertain environment, choose to model themselves on other organizations. This means that as long as aspects around using open source software are unclear, and similar organizations are not using F/OSS, public organizations will likely not choose to adopt F/OSS.

Overcoming these barriers is possible by addressing the uncertainties in the public sector, raising the awareness of open source alternatives to public organizations and their benefits. For example addressing the security concerns that some public administrations might have against F/OSS, indicated in the EnFeOSS study (Giacomo, et al., 2012). One informant from the expert group points out that F/OSS has proven itself to be more secure than its proprietary alternatives in recent years. This is much due to auditability benefit of F/OSS that is mentioned in literature (GBDirect, 2001). It is therefore important to make public administrations aware of the benefits of F/OSS. Also helping is government's policies towards increased use of F/OSS.

A technique for communicating to the outside world is the use of boundary-spanning when raising awareness of an open source project. The Open e-PRIOR project is already doing boundary-spanning techniques in that they are going to conferences and speaking to people at events and also doing boundary-spanning online. The literature also mentions that this is a very effective technique when used on the internet (Hars & Ou, 2001). Open e-PRIOR could approach organizations with common interests like Difi (Agency for Public Management and eGovernment), and the corresponding agencies across Europe to a workshop discuss a draft proposal on how to collaborate on the further development of the project The expert group also mentions other

boundary-spanning techniques that are widely used today. Getting publicity in the press by being interviewed by reporters is relatively easy if you have something interesting to communicate. Besides, getting an article in the paper is very valuable if you compare it to what an advertising campaign with the same coverage would cost. Also blogging and guest-blogging was mentioned several times by informants. From these it's also possible to derive a social-media strategy that leverages social media channels to get publicity around the project. In addition, case descriptions in online platforms like the ePractice.eu (ePractice, 2012) or reports like the European eGovernment surveys or the United Nations e-Government survey can also help spread the awareness.

5.2. Distribution and release

The Open e-PRIOR software is today made available through the JoinUp platform where you can download the latest and previous releases of the software. However, the intermediate source code is not available to the community on the JoinUp project site. Making source code available through the project site so it's easy to find and inspect by the public would be in line with general recommendations for open source development and motivate more people to contribute.

The problem of having two different development threads are illustrated by expert B, where you will have a problem with the two diverging as they are both developed in different directions. Although having two source code branches is not a problem, it's recommended that that there be only one main development thread. Expert B suggests that the community version should be the main development thread, and then periodically take from the community version to make a "stable" version that has gone through quality assurance rigorous testing procedures. The expert mentions a good example of a project where two versions of the software exist; one community version, that is the main development thread, and a "certified" version. Both versions are open source, but the certified version is thoroughly tested and suitable for production use.

Similarly, an option for the Open e-PRIOR project is to merge the two versions into one version, making the Open e-PRIOR version the main development thread, and have the EC development team develop on this version. Periodically taking selected contributed features from this version over to a "Certified" version that go through the quality assurance and testing procedures for releasing a new version.

5.3. Community

The Open e-PRIOR project faces several governance challenges of evolving the internal endeavors of the EC into engaging the users and other organizations in EU member states in a community around further development of the software. Engeström (2004) describes that this model of "co-configurative" work requires the integration of the user-organizations as active subjects. For this collaborative model to work, a dynamic dialogic relationship between the different actors is needed.

To be able to motivate and engage new and existing members to the community, the current Open e-PRIOR team should change their processes to work in the public. Then as new members join the community, they will be able to see the whole process of developing Open e-PRIOR and participate. Several of the F/OSS experts state the importance of the planning activities and decision making being done in public thus

generating trust and credibility to the project in the eyes of the outside world, while the full transparency of activities in the project will motivate and engage new members to join. Since these activities are now done internally at the EC, this leaves the external members of the community with little influence on the direction of the project. To enable the community to have more influence on a project, a commonly used method combined with a public planning process, is introducing ways to vote for features etc. Another study by O'Mahony and West (2008) describes that there is a control vs. growth tension when organizations try to keep control over the project's strategic direction. Keeping this control limits the ability for the community to grow.

Informants of the expert group mention the importance of communicating a clear mission statement on the website. A few informants go on to elaborate that it's important that this mission statement is unifying for both the community members and the vendor or organization running the project. It should be something that both will want to adhere to. This will help create clarity of what the community and project is there for, and help as a guide to make sure the project stays on course. This again will help motivate community members by building trust to the organization; this trust from the community is kept as long as the organization stays on course and doesn't drift from the mission statement.

In the Open e-PRIOR case, the community is being built around the existing software already developed by the EC. In their literature, O'Mahony and West (2007; 2005) warns that with projects like this, the challenges differ from community-founded projects in that since the community is not a part of the initial development, the motivation for joining the community and contributing is affected. This is also backed up by the expert panel who points out that rewarding the community is the way to motivate and engage them to join and contribute to a project. Making the community members feel a greater sense of ownership to the code is to consider creating something that can compare to a "summer of code" event for Open e-PRIOR. Another example mentioned that could be relevant for Open e-PRIOR is the reward of increasing skill base. New members will be working in collaboration with experts from the engineer team that will be able to help them by reviewing their code etc. ultimately increasing the skills of other members of the community. This is also supported by research done by Hars & Ou (2001).

5.4. Governance

Until now the Open e-PRIOR project has not been very transparent to outsiders, mostly because most of the team members doing any work on it are all located at the same place. The project also started out this way, and they never had any need to use public communication channels since it was easier to just have meetings face-to-face. Ambitions to expand the project to include external community members will, however, drastically change the game for how work needs to be done. Experts interviewed underlined the importance of open and transparent processes, stating that it is one of the cornerstones of open source development.

The lack of transparency, slow decision process and rigid command and control structures can leave normal community members feeling like 2nd-class citizens, and that demotivates members enough to in some cases leave the community. Because if they feel like they can't get forward in the community and gain more influence, they tend to

become bored and quit. For Open e-PRIOR this might be reflected in the fact that the Open e-PRIOR engineers use different tools than the community has access to, excluding the other members somewhat from parts of the project that they might be interested in having access to. Some informants emphasize the importance of not ignoring suggestions from the community when they engage themselves. Many informants mention examples of projects ignoring their community from the real world, like i.e. Sun and Open Office who did exactly this, with negative consequences to follow.When Sun microsystems bought a company called Star division, they had a good many people inside who were doing development. But they didn't really care that much whether the outside community was contributing or not. This sort of project really should have been as important as Linux, except that company did not run it as a community project at all. And for that reason they only had about a 100 different entities who had signed on as contributors to Open Office. In general if you're not going to operate as a community project, it might not be terribly effective to be an open source project at all. Finally, when Sun microsystems were purchased by Oracle, the community forked the project and split it off to Libre Office. And Libre Office has been developing at very high speed.

One of the things mentioned in the data often is the need to have meritocratic government in open source software projects, where a contributor would get rights in the project based on his achievements. O'Mahony and West (2005) and Von Krogh et al.(2003) also points out the importance of this. As members become more involved in a project and they contribute more, their rights and status increases in the project. In really large communities this would be very difficult to manage in a fair way if done manually. One of the informants suggests that a reputation engine be used in those cases, which will automatically generate metrics based on a member's activity in a project. Then it is easier to be fair when managing rights in a very large community. Open e-PRIOR on the other hand is not there yet, and a manual process would work just fine as a start, but it might be useful to establish a meritocracy never the less. Informants explained that it's normal for a vendor or organization backed open source project to have a central ruling body that takes the important decisions. A meritocracy could work as a motivating mechanism for any member to work their way into the project and end up in the ruling body. Accomplishing that would also give the community and project a whole new level of diversity, which according to informants, is very important to a well-functioning community long-term. Other communities like Ubuntu and Gnome for example, have organized foundations as their ruling bodies and they employ a voting system for deciding what and when new features should be implemented. The Open e-PRIOR project could explore this idea of forming a foundation for the project. In this approach, inviting external members for the board is important as the expert group explains, to get diversity in the decision group.

5.5. Documentation

As pointed out by external evaluations, the Open e-PRIOR site at JoinUp has documentation for the software available through office documents that are fairly comprehensive. This makes it more difficult to navigate for readers who are looking for a certain piece of information. Keeping documentation in documents like this and having the user download it matches the way Neus (2001) describes burying information in an unwieldy tool. Multiple informants emphasize the importance of having websites that are more intuitive for anyone and the information is digested and placed on a

natural place so it's easy to find. For the Open e-PRIOR project this could be done by extracting important information from documents and making them available as web content for easier navigation and use collaboratively by the community; for example replacing a broken link or updating the information as it is outdated. A very common tool for this is a wiki, which enables the community to collaborate on content more easily. Scoping the information to different types of readers is also important.

5.6. Technical

The software architecture of Open e-PRIOR is fairly complex and consists of many different components doing different things. The software has initially been designed by the EC and then released as open source, building a community around the project. O'Mahony (2007) warns that not developing the code and the community in parallel can affect the external developers' ability to contribute to the project by making the learning curve too steep for beginners. Getting started should be as easy as possible. Many from the expert group mention this. It is especially important for users downloading and installing the software because almost all new community members start off as users first. So it's important to attract them as users. Some may stick around to become contributors eventually. Developers can tolerate a bit more hassle than users to get started, but that doesn't mean that the developers in the community shouldn't try and lower the entry-barrier as much as possible. Experts explain that the lower the barriers are, the bigger the chance that a new member will be able to contribute back to the project. One way to make it easy for developers to get started is to have good architecture and modularize the code base so that a new developer doesn't have to learn the whole system to be able to make his first changes. Another is the installation and deployment procedure for the software. For end-users this should have the lowest learning curve, preferably automated through wizards or install scripts. To lower the threshold for users to get started you could provide a sandbox installation for users to try out online, and also make sure that there is always an updated properly compiled package to be downloaded, similar to installing Apache or Postgresql on a Debian system. This will make it less likely that a user gets frustrated and moves on before even getting started. For developers the same argument applies to some extent. If a developer is unable to get the software up and running on his machine, then he might just quit and often never even tell anyone about it. The evaluations done earlier (Teinum, 2012) illustrate this will, as the evaluator didn't manage to get the software up and running in a reasonable time and therefore failed to contribute any code to the project, even though this was the intent. It can be wise to publish a selection of low hanging fruits as entry points for beginners. Those could be testing tasks, documentation etc.

5.7. Collaboration

It's well established that providing good collaboration tools is crucial for anyone trying to set up an online community, and when development of software is involved it becomes even more crucial. Some informants in the expert group pointed out that the tools were not as important as the processes they support. For most online communities, regardless of whether there is software development involved, a form of asynchronous communication channel is essential. Also the research by O'Mahony and West (2005) underlines the importance of tools for online discussions. This can either be a forum or a mailing list for example. The choice really depends on a lot of things

like, the size of the community, the purpose etc. Open e-PRIOR has already a mailing list and a forum, but they are not being used extensively by the community, nor by the EC developers. The technical aspects of a mailing list are not that complicated. A community member simply needs a way to subscribe and unsubscribe to it. What's more interesting is that the content being sent out on the mailing list should be digested in a way that there is a minimum of junk going out on the list. If there is too much junk in the mails from the list, members will often either unsubscribe, or start ignoring mails from the list. One informant went on to call these kinds of mails for "Cyber CRUD". He went on to describe several mails from different mailing lists he had in his inbox, and his personal opinion was that users tend to start ignoring pretty quickly mails that look very machine-generated, whereas a better design of the content would help getting the message through to the reader at a glance, rather than having to dig into the mail to find the interesting parts.

Another very common tool for this kind of communication is forums. They are less intrusive than mailing lists since the user has to go visit them to find and collaborate through messages. This paradigm is better suited for some collaboration forms, like support questions to the community for example. This is often the main collaboration tool of an online community, which is why it's essential that the technological solution is stable and easy to use for community members. Otherwise they will switch back to mail or other communication forms that disrupts one of the main benefits of these tools. Messages that are posted are archived so that new members can search for what they want to know, and if it has been asked and answered before, the community doesn't have to answer the same questions over and over again. This way, forums will gain higher and higher value to the community the more actively they are used. Even search engines will index the entire publically available contents of a forum easily so the content is searchable from all major search engines.

When moving over to the software development activities, O'Mahony and West (2005) and the expert group agree that the most important tools for a community to collaborate is a bug tracker with light product management features and forums or mailing lists. For Open e-PRIOR there are two competing tools that do a lot of the same. JoinUp's bug tracker, which is the one publically available today, and the internal JIRA bug tracker. The JIRA bug tracker is the main tool used by the EC for software development. Even though they are both bug trackers, the feature sets of these two tools are quite different. The bug tracker at JoinUp is inferior in functionality to what the JIRA product offers. The JoinUp bug tracker is missing project management tools required for good community development, like planning and coordination between developers. Some of the informants from the EC group with intimate knowledge to JoinUp went far to indicate exactly this, and said that the JoinUp platform was not really the best of breed in this regard. They also pointed out that there are other free tools out there that would do the job much better.

JIRA is actually one of these free tools, as Atlassian offers all of their tools for free to open source projects under some simple terms that the Open e-PRIOR project easily qualifies to (Atlassian, 2012b). One immediate benefit of this scenario would be that the Open e-PRIOR engineering team already knows how to use these tools and would be comfortable straight away. Another is that JIRA is one of the most popular issue trackers in the world, and the chances that new members would also already know this tool are pretty high. A third is that Atlassian also offers all the other tools in their suite for free, and all these tools are made to be integrated with each other. Everything from

bug tracker, wiki, build server, code review, automated testing tools, to mention a few. There are also other open source tools available that could cover the same functionality, i.e. Git, Trac, MediaWiki and Gerrit, that would be familiar to open source developers. But regardless, what's important is that the members of the external community and professional engineering team all use and collaborate through the same tool-chain.

Another important tool mentioned by O'Mahony and West (2005) is a source control tool for collaborating on the code. For simple projects it might be enough with a single repository for the whole project, especially when the community and code base is very small. When the community or the code base grows, the need to properly organize the source code will arise. This will also depend on the amount of activity going on in the project and is a coordination challenge. For the ruling body it might be tempting to just dictate and take decisions without any regard to the needs of the community. But for the community members, it's also a question of freedom and influence, and being able to do what they want in their own tree. So the need to have more than one branch or tree will arise eventually. In these situations it is important to find a solution that works for both the engineering team and the external community developers. The expert panel proposes that one way of doing this is to distribute the source control management through Git for example. Git will allow you to keep your central repository that the ruling body has complete control over. But any developer can fork the central repository and start his own branch without any extra management. If the developer on the satellite branch ends up with something that he wants to contribute back to the central repository, Git uses a mechanism of pull requests. This way the ruling body of the central repository can pick their litter in contributions to go into the next release by accepting a pull request after evaluating whether the contribution is good enough to go into the next release. Then normal quality assurance can be done on the code in the central repository and the release process runs as normal. For Open e-PRIOR, this might also be an interesting approach to enable the community to contribute easily without a lot of extra administrative overhead. Git is widely used by many prominent open source projects today, like the Linux kernel, Android, Drupal, Fedora, Qt and VLC to name a few (GitProjects, 2012).

5.8. Anti-patterns and refactored solutions

In the case of Open e-PRIOR, the findings indicate the presence of some of the most common anti-patterns for communities and they all have well documented proposals for refactored solutions. Interestingly enough, the refactored solutions to these anti-patterns coincide very well with the input from the external F/OSS expert group. This should only strengthen the case for the proposed solutions. As the literature on anti-patterns describes, once an anti-pattern has been identified, you can remedy the problem by refactoring the pattern toward improved benefits and minimized consequences (Brown, et al., 1998).

5.8.1 Command and control anti-pattern

As the EC is the founder of the Open e-PRIOR project, it is natural for the EC to want to want to have strong control of what features are included in the software, for reasons such as to protect reputation with users, ensure quality and testing. This "command and control" approach is likely to counteract the successful growing of a contributor community. This is supported by the control vs. growth tension described by

O'Mahony and West (2008). Examples of this can be roadmaps that are not being published or lacking the room for external influence.

The volunteers entering into the project are likely to be undermined by the "command and control" management pattern and features will seem to appear by announcement without public discussion, much as described in the case of Open e-PRIOR's planning activities. The consequence of this anti-pattern can be a reinforcing loop preventing the successful growth of a healthy community due to the lack of trust in the community itself.

The literature describes that the command and control structure anti-pattern is best treated by exchanging influence for control (CommunityManagementWiki, 2012). This means that the community members who are not a part of the control organization would have to get incorporated into the decision process and have the possibility of influencing the direction of the project. The founders of a project will in any case warrant enormous respect and authority in a project.

Literature also supports this solution, where F/OSS projects are described as co-configurative work. Collaboration among people of varying expertise including professionals, experts and users requires a dynamic, dialogic relationship between the different stakeholders of the project (Engeström, 2004).

Several informants in the external experts group mention that the community needs to be rewarded to keep the community motivated and engaged; by given influence in the project and the direction it's going. This solution also fits into the description of meritocracy in a community project.

Another proposed treatment to this anti-pattern mentioned by literature is to open up the processes completely and work in the public (CommunityManagementWiki, 2012), which fits well with the need for transparency that can be seen very strongly indicated by all the informants in the data gathered through interviews.

5.8.2 Water cooler anti-pattern

Since the majority of the communication in Open e-PRIOR is done internally in the EC, there is clear evidence of the "water cooler" anti-pattern. This anti-pattern is described by the symptom of too much of the work being done behind closed doors. Although there is a public newsletter and forums available through the JoinUp site, it might be difficult for community members to understand the motives and priorities of the project.

Literature also have a clear treatment described for this anti-pattern, which coincidently is the same refactored solution as the second one in the command and control anti-pattern; process transparency and working in public mode.

5.8.3 The big show anti-pattern

Up until now most of the development work on Open e-PRIOR has been done internally in the EC. The Open e-PRIOR project releases periodically new versions of the software based on a roadmap of features. However the intermediate source-code

and development progress between releases remains hidden for the public community. This is an indication of the Big Show anti-pattern, a phenomenon where organizations do their work behind closed doors, maybe for months at a time before announcing it to the public. The ability to motivate outsiders to become interested and for a community to grow is negatively impacted by this behavior. Project members assigned to the project work on these features, interacting less with the community. At the end, the result is release of a huge chuck of code that's had no peer review from the external community. This might also result in people outside the company feeling like 2nd class citizens.

The big show anti-pattern solution needs a bit more balancing to remedy the situation. But a code-drop type release announcement like in Open e-PRIOR today is too late for the community to get involved and start participating. So announcing new features to the community early and then involving members at an earlier stage is a solution (CommunityManagementWiki, 2012). Like some informants emphasise that community members should be involved in the planning phase, and that planning decision making should be done in public in the community in addition to the development.

5.9. Summary of recommendations

The following table shows the identified barriers that need to be addressed and corresponding potential measures for overcoming them.

Table 5.1: Summary of barriers and measures

Categories	Barriers	Measures	Suggestions for Open e-PRIOR
Adoption	Few public administrations adopt the software.	Learn from commercial marketing Raise awareness with boundary spanning	Invite Difi, and the corresponding agencies across Europe to a workshop discuss a draft proposal on how to collaborate on the further development of the project.
Distribution and release	Source code visibility is low	Make the source code more accessible Organize source code according to best practices	Provide binaries to download and install. Supplement with online sandbox installation for users to get a first flavor of how it works.
Community	Community's influence on project is low	Reward and motivate community members to get engaged and join Don't treat members like 2nd class citizens Let community have influence on the project	Consider to introduce ways to vote for features etc.

	Recruitment to the developer community is low	A clear unified mission statement for organization and community Be welcoming to new members	Consider how to create something that can compare to a "summer of code" event for Open e-PRIOR.
Governance	Development process is hidden from community Decision making process of EC is slow	Open up process and make visible to community	Make sure internal EC staff pursue more decision-making openly online and publish questions and answers etc.
Documentation	Access to documentation is unwieldy	Make documentation easily accessible	Replace PDF documents with Wiki pages, and rather consider to create PDF documents related to stable versions for the record.
Technical	Technical entry-level is high for beginners	Lower entry-barriers as much as possible for all roles	Publish a selection of low hanging fruits as entry points for beginners. Those could be testing tasks, documentation etc.
Collaboration	Collaborative tools for community development is insufficient	Provide well-functioning collaboration tools Unify tool-chain for community and vendor's engineering team	Consider to switch to more commonly used open source tools such as Wikimedia etc.

Chapter 6

Conclusion and limitations

6.1. Conclusion

Because of its basic philosophy F/OSS has a potential to succeed in the public sector. The openness and sharing mentality of F/OSS is congruent with the accountability governments have to its citizens. The Open e-PRIOR project is an example of such a project. A goal of the project is pursuing an open source strategy that will build a community around the project and enable the sharing of knowledge, collaborative development and re-use of this software across the whole of Europe.

Through this study of the Open e-PRIOR project, several barriers that need attention for the project to build a community and reap further benefits of F/OSS have been identified along with possible measures to overcome these barriers. The barriers and measures are presented in **Feil! Fant ikke referansekilden.**

Public administrations reluctance to adopt the software is hindering users to become potential community members. In addition, because of the way the project development is set up today, it lacks a practical way of letting outside members of the community collaborate on development of Open e-PRIOR. Even though the project is present at the JoinUp forge, it is not used by the EC development team, hiding the development process from the public and leaving the community with little ability to influence the project. The source code developed by the team is only made available on each release and the accompanying documentation is released in unwieldy PDF documents. In addition, the technical entry-barrier to the project is an issue for new users and developers. Another factor hindering community contributions is the lack of transparency in the decision-making process. Planning and development is mostly done internally at the EC, and future plans are communicated in a one-way fashion and not in detail to the community. In summary, much of main challenges boil down to transparency of the processes and enabling the community to be able to become an engaged part of the project in an easy way.

This answers the first research question.

Succeeding in building a sustainable F/OSS community around the Open e-PRIOR project, this study proposes that the EC address uncertainties that public administrations might have to F/OSS and raise awareness of the benefits that open source might contribute to these public organizations. Secondly changing the project processes from internally hidden, to be more transparent and visible to the community, also enabling the community to take part in the decision making process i.e. by introducing a voting system for community members. The project needs a better way of collaborating on software development with the external community either by improving the current collaboration platform, or find alternative F/OSS development

tools to build a new community developer site. The collaboration platform need to provide the community with access to the current development source code, a better bug tracker with simple project management features, and asynchronous communication tools, all enabling for the public visibility of the on-going development process. To further motivate new members to contribute to the development, the entry-barrier should be lowered as much as possible, making it easy to get started on a low hanging fruits for beginners to pick.

These barriers are also described through a practical lens of three identified community anti-patterns which help illustrate the presence of challenges. Refactored solutions the community anti-patterns are described, by applying the measures identified.

This answers the final research question.

6.2. Limitations

Though care has been given to ensure that the proposed strategic measures is of generic nature within the context of public administrations, it could be argued that following the use of one single case-study, the findings are partly limited to Open e-PRIOR. Hence, in order to further generalise the findings, further research on other F/OSS projects should be carried out. Also, since the data collected from the F/OSS expert panel is based on their own experience it is unavoidable that in this study, certain degree of subjectivity can be found.

Further, the EC and any other entity considering the use of the proposed strategic measures as input should bear in mind that in order to ensure successful building of an open source community, the measures should be accompanied by a number of other aspects. Such aspects were defined as out of scope of this project (Chapter 1.4) and include activities such as defining of implementation plan, review, change management, training of staff, project management, etc.

Chapter 7

Bibliography

Applewhite, A. (2003). Should governments go open source? *IEEE Software, 20*(4), 88-91.

Atlassian. (2012a). JIRA - Track bugs, tasks and projects for software development, 2012, from http://www.atlassian.com/software/jira/overview

Atlassian. (2012b). Open Source | Atlassian, 2012, from http://www.atlassian.com/opensource/overview

Boynton, A. C. (1998). *Invented here: Maximizing your organization's internal growth and profitability*: Harvard Business Press.

Brown, W., Malveau, R., & Mowbray, T. (1998). *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*: Wiley.

CommunityManagementWiki. (2012). Community Management Wiki - Anti Patterns, from http://communitymgt.wikia.com/wiki/Category:Anti-patterns

D'Orazio, S., Fichera, M., Rodrigues, J. F., & Daniels, M. (2010). *Open e-PRIOR Software Architecture Document*. European Commission.

Dar, O., Forsbakk, D., Johansen, A., & Liljemo, K. (2011). *Open e-PRIOR study : A Master student project at the University of Agder, Carried out in a collaboration with the European Commission*.

DiMaggio, P. J., & Powell, W. W. (1983). The iron cage revisited: Institutional isomorphism and collective rationality in organizational fields. *American sociological review*, 147-160.

Dougiamas, M., & Taylor, P. (2003). *Moodle: Using learning communities to create an open source course management system*.

Engeström, Y. (2004). New forms of learning in coconfiguration work. *Journal of Workplace Learning, 16*(1), 11-21. doi: 10.1108/13665620410521477

ePractice. (2012). ePractice.eu, from http://www.epractice.eu/

GBDirect. (2001). Benefits of Using Open Source Software. Retrieved from http://open-source.gbdirect.co.uk/migration/benefit.html#auditability

Giacomo, D. D., Goedertier, S., Liljemo, K., Frade, J. R., & Hee, N. V. (2012). *ENFEOSS Vision Document - The vision and business case for an enhanced software catalog for e-government*. European Commssion.

GitProjects. (2012). Projects that use Git for their source code management., 2012, from https://git.wiki.kernel.org/index.php/GitProjects

Hars, A., & Ou, S. (2001). *Working for Free? – Motivations of Participating in Open Source Projects*. Paper presented at the Hawaii International Conference on System Sciences, Hawaii.

JoinUp. (2012). JoinUp : Open e-PRIOR Retrieved 26.10.2011, 2011, from https://webgate.acceptance.ec.europa.eu/joinup/software/open_e-prior/description

McDonald, C. J., Schadow, G., Barnes, M., Dexter, P., Overhage, J. M., Mamlin, B., & McCoy, J. M. (2003). Open Source software in medical informatics--why, how and what. *International journal of medical informatics, 69*(2-3), 175-184.

Neary, D. (Producer). (2010). Community Anti-Patterns - MeeGo Conference 2010.
Neus, A. (2001). Managing information quality in virtual communities of practice. *IQ*, 119-131.
Nygaard, K. (2012). Kristen Nygaard, from http://en.wikipedia.org/wiki/Kristen_Nygaard
O'Mahony, S. (2007). The governance of open source initiatives: what does it mean to be community managed? *Journal of Management and Governance, 11*(2), 139-150.
OFE. (2011). Open IT Procurement in the UK Public Sector. Retrieved from http://www.openforumeurope.org/openprocurement/openprocurement/open-procurement-library/Open%20IT%20procurement%20final%20version%2001_11_2010.pdf
Peeling, N., & Satchell, J. (2001). Analysis of the impact of open source software. *QinetiQ Ltd. QINETIQ/KI/SEB/CR010223. Available at http://www.govtalk.gov.uk/interoperability/egif_document.asp*.
Perens, B. (2007). The Open Source Definition. Retrieved from
Settas, D., & Cerone, A. (2011). Using antipatterns to improve the quality of FLOSS development. Retrieved from
Teinum, A. (2012). *Open e-PRIOR - An evaluation*. University of Agder.
Thunus, D. (2012). *DIGIT.B4 eProcurement Section - Software Development Plan*. European Commission.
Von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy, 32*(7), 1217-1241.
West, J., & O'Mahony, S. (2005, 03-06 Jan. 2005). *Contrasting Community Building in Sponsored and Community Founded Open Source Projects.* Paper presented at the System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on.
West, J., & O'mahony, S. (2008). The role of participation architecture in growing sponsored open source communities. *Industry and Innovation, 15*(2), 145-168.
Wikipedia. (2012). Open source software development, from http://en.wikipedia.org/wiki/Open_source_software_development

Appendix

Annex I. Interview invitation email

Dear Sir/Madam,

My name is Atle Johansen. I am a Master Student at the University of Agder in Norway. I am doing my master thesis, conducting a case study of an open-source e-procurement software project at the European Commission.

My master thesis is focusing on “Succeeding with an open source strategy”, where I seek to obtain a detailed understanding of the processes and approaches of barriers in community building around an open-source project in the public sector.

It would be extremely useful for me to learn about your expert experience and knowledge of open-source software.

In order to investigate your insight in open-source software I would be very grateful if you could have time to allow me to conduct a telephone or Skype interview. The time and date will depend on your agenda, and the interview will basically cover the following topics:

1. Open-source governance and development process
2. Open-source communities
3. Communication barriers
4. Open-source adoption
5. Technical barriers
6. Open-source project infrastructure

Your participation will significantly contribute to the success of this research and your help would be highly appreciated.
If you have the opportunity, I propose conducting the interview at **[DATETIME]**
If this is not a good date for you, feel free to propose another date that is more convenient for you.

Should you have any queries, please feel free to contact me through
telephone: +47 911 23 093, Skype: johatl or email: atlej04@student.uia.no.

Thank you for your kind attention and I am looking forward to receiving your reply soon.
Have a nice day!

Yours sincerely,

Atle Johansen
Institute for information systems,
University of Agder

Annex II. Interview guide

[Name of Informant]
[Address]
[Telephone number]
[Email]

[Date]

Succeeding with an Open Source strategy: Interview Guide

Introduction

This interview is conducted as part of my Master's Thesis "Succeeding with an Open Source strategy: a case study of Open e-PRIOR, an e-Procurement software project at the European Commission". The master thesis is run at the University of Agder in Kristiansand, Norway, at the Department of economy and social sciences - Information systems, and in cooperation with the Directorate-General for Informatics of the European Commission, Unit B4.

As indicated in earlier communication, the purpose of this interview is to gain further insight into how to succeed with an open source strategy.

The interview consists of mostly open questions leaving the interviewee room to answer as freely as possible according to his/her context, and the duration of interview will be between 40-60 minutes.

Contact information for interviewer is:

Atle Johansen
Email: atlej04@student.uia.no
Tel: +47 911 23 093
Skype: johatl

The interviews will be recorded, transcribed for analysis, and then deleted after project is completed. I would like to reassure you that your questions will be handled strictly confidentially and no individual information about your answers will be disclosed without your previous agreement. All answers will of course be handled confidentially.

The interview will be carried out through a telephone/Skype interview on [Date Time]

The contents of the interview are further explained in the next section, followed by the questionnaire.

Interview Details

The questionnaire is divided in four sections:

1. **Introduction**

   This section aims to collect information about you and your previous experience. We would like to once again reassure you that any information provided will be kept strictly confidential unless otherwise explicitly agreed with you.

2. **Analysis Area 1: Governance**

   We would like to collect your input on governance aspects. We have divided the section between development process and community.

3. **Analysis Area 2: Communication**

   Here we would like your input with regards to communication from an Open Source project to the outside world to attract users and contributors.

4. **Analysis Area 3: Technical**

   In this last section we would like to collect your input on technical barriers and important aspects of project infrastructure.

Questionnaire

The table below contains the questions that will be asked during the interview along with a brief explanation to describe possible options of each question.

Question	Interviewer Notes
Introduction	
1. What is your name?	
2. What is the name of the organization that you represent? What is your role in the organization?	
3. Where are you located?	- Country / institution
4. What is your relation to the Open e-PRIOR project? (Have you heard of the Open e-PRIOR project)	- Developer / Facilitator - Peripheral - External
5. What is your experience with Open Source software	- As user - As community member / contributor
Analysis Area 1: Governance	
Governance/Dev. process	
6. What are your thoughts on the importance of process transparency in open source software projects?	- Also Open e-PRIOR for relevant people
7. What activities do you think are most important be visible to the community in an open source project?	- Ask about planning - Ask about release - Ask Open e-PRIOR about possible "hidden" processes
8. What release / distribution process do you think is most beneficial for an open source software project?	- Continuous integration - Nightly builds - Periodic, Feature-based releases
9. At what point do you think a member should gain commit access to source code when joining a community?	- Ask about joining script - Also Open e-PRIOR for relevant people
Governance/Community	

Question	Interviewer Notes
10. What do you think are important factors for members who would like to join a software community project?	- Community process - Joining script
11. What are your thoughts on trust relationship within an open source community?	- Skill level of members - Communication "tone" - Comfort for members
12. What factors would you say are important when organizing a software community? 13. (different kind of community types) (roles and responsibilities)	- Co-developer community (extending) - Deployer community - User community - Homogeneous mass - Different groups of users
14. What do you think are important incentives for an organization to release software as open source?	-
15. How do you think this is different in the public sector?	-
Analysis Area 2: Communication	
Communication/Barriers	
16. What do you think are good ways to raise awareness of an open source software project and reaching target groups?	- Also Open e-PRIOR for relevant people
17. What available information do you feel are most important on an open source project website?	- Ask about JoinUp when relevant Use source-forge when not - Technical level of information - Audience for community site information
18. How do you think support is best organized in an open source software project?	- Forums/phone/ITIL
19. What are your thoughts being a member of a community if you have confidentiality challenges i.e. through an NDA?	- Rules of EC / NDA
Communication/Adoption	

Question	Interviewer Notes
20. What do you feel are important incentives for adoption of open source software?	- Describe
21. How important is (perceived) software maturity for open source software adopters do you think?	- Perceived quality - Perceived usefulness - Technology acceptance
Analysis Area 3: Technical	
Technical/Barriers	
22. How important do you feel it is for an open source software project, that getting started is as easy as possible?	- Requirements - Installation / Deployment - Customizing/ Building
23. What importance does software architecture have for open source software projects?	- Modularization of software - Open vs e-PRIOR
24. What are your thoughts on access to the source code and source control in open source software projects?	- Git / SVN - Restrictive access - Dual repository
25. Many software projects have online demos of their software; do you think these could have a significant effect on open source software adoption?	- Feasibility of online demo of Open e-PRIOR
Technical/Project infrastructure	
26. What importance do you think the availability of collaboration tools are for a community?	- Wiki - Bug tracker - Forum/mailing list
27. What do you think is important features to promote usage of such tools in an open source project?	- User friendly - Collaborative features
28. If several overlapping systems are present; what consequences do you see could arise?	- Integration of tools - Use of only one?