Inner Source – to Excel on Developer Initiatives

Master's thesis in Master Programme Software Engineering

QI WANG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY AND UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Master's thesis 2016:10

Inner Source – to Excel on Developer Initiatives

QI WANG

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden 2016

Inner Source – to Excel on Developer Initiatives

Supervisor:
Regina Hebig, Department of Computer Science and Engineering
Dan Berg, PDU Packet Core of Ericsson

Examiner:
Eric Knauss, Department of Computer Science and Engineering

Master's Thesis 2016:10
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Inner Source, Open source for the enterprise
(https://cloudramblings.me/2013/11/08/inner-source-open-source-for-the-enterprise/)

Gothenburg, Sweden 2016

Abstract

This thesis improves and studies the Inner Source Development environment in Ericsson. Inner Source Development focuses on cooperation among developer, manager and customer, and collaboration among developers. This thesis starts with 7 semi-structured interviews with developers and managers from different teams in Ericsson. The aim of semi-structured interviews is to know the requirements of stakeholders and their opinions on existing tools used in Ericsson and on the market. The thesis presents a comparison among 5 different tools: Eforge, OpenALM, Gerrit, Gitlab and Github Enterprise from 4 aspects: searching, sharing, collaboration and other quality issues. With the interview result and the comparison, the original plan to improve the Inner Source Development environment was to import Github Enterprise or Gitlab EE. However, with some obstructions of funds and pilot plan, this plan was rejected and turned into improving the existing tool: Gerrit. Gerrit is improved on 2 aspects: searching function and complete information of project. Three new function were developed and deliveried to Gerrit Open Source community. The largest challenges in this thesis are understanding of large amounts of code, different ways controlling of modified system, communication problems because of jet lag and lack of time to test. The study provides a table of comparison among 5 tools.

Keywords: Inner Source Development, Requirement Specification, Open Source Development

Acknowledgements

This thesis work was cooperated with Ericsson and Chalmers University of Technology and University of Gothenburg. The author would like to thank the supervisors: Dr. Regina Hebig, Mentor Dan Berg and the examiner: Dr. Eric Knauss for his valuable feedback and guidance. The author would like to express her thanks to the technical guiders: SW Configuration Manager Stellan Bondesson in Ericsson in Gothenburg, Gerrit Developer Hugo Ares in Ericsson in Montreal, the explaination and support from Github: Hakan Axtelius, Bas Peters and Sten and Sven Stolk, and environment configuration support: Magnus Mörtberg and his colleague (Vlad). The author also would like to thank all the interview partners and participants involved in discussions.

Qi Wang, Gothenburg, October 2016

Contents

List of Figures

List of Tables

1
Introduction

Inner Source development is defined as applying Open Source development within a closed organization [2]. Inner Source concepts are similar to open source, except Inner Source is bounded by an organization or a company. Inner Source platforms are tools embodying Inner Source concepts to help users be involved in projects, share projects, submit contributions, search, and join other projects.
Open Source Software development is now very common and has achieved great success in both industry and academia. Several outstanding hackers (including Bruce Perens and Eric Raymonf), initiated the Open Source concept in 1998 [1]. Open Source software refers to software projects where the source code was publicly published and available to be reviewed, contributed, and even used by other developers. Huge amounts of open source projects were published on different platforms, and the amount is still growing. In 2016, Nithya boldly stated that all companies were using open source or would use open source in the future [3]. Stakeholders took many benefits from Open Source development. However, organizations and companies are continuously looking for more solutions to manage projects.

Open Source development is internet-wide for users, i.e., anyone who has access to the Internet is free to open source projects. This has various risks for organizations and companies who build their products in Open Source. There is no secrets in Open Source world, for some profit organizations, Open Source may risk leaking their products, intellectual property, or customer information. To avoid these problems, a specific strategy for organizations based on Open Source development has been proposed, called Inner Source. Inner Source Development applies Open Source strategy of culture, tools and practices into an internal software development[50].

1.1 Problem

Ericsson has a structured way to handle production code and test environment or portal development. However, they also have many side projects aiming to facilitate their development, such as small tools and scripts, troubleshooting and analyze support. Those small projects are developed by developers. In the company, people are not satisfied with their current platforms to share projects. There are many departments working on different floors. Furthermore, some departments have cooperation in different countries. It is hard to hold regular meetings or discussions face-to-face. Even two people working in the same department and developing the same scripts, may not recognize each other. People work individually as islands during the development of the small tools and scripts. To organize them and to increase efficiency and promote collaboration, a good tool is necessary. This tool would support developers in the development of all kinds of "side" projects supporting main stream development.

1.2 Purpose

The motivations for Inner Source development include increasing project quality, code reuse openness, and transparency within the organization [4]. A suitable Inner Source development environment within a company, will help build higher quality projects and reduce expense.

This thesis was developed in cooperation with Ericsson, with the aim to improve their Inner Source development environment. The project must first understand the state of the art of Inner Source platforms in the context of Ericsson's requirements. Then develop a prototype platform that combines advantageous features of existing platforms and addresses the identified requirements not currently supported in existing platforms. This could be achieved by setting up an existing platform to meet these requirements or improve a tool currently in use.

1.3 Research Questions

Two research questions [5] guide the strategy of this thesis:

RQ1: How does an Inner Source development support team work?

RQ1.1 - What are the stakeholder's requirements for Inner Source development?

This research question is the foundation of this thesis. We must first understand what is Inner Source, why is it useful or desirable, and understand stakeholders' basic model and functionality for Inner Source development applied to team work, since almost all developers work as teams in Ericsson. This research question includes sub-question: RQ1.1.

RQ2: How could the Inner Source development environment be improved?

This represents the forward research aspect of the thesis, to improve the Inner Source development environment according to the answers to RQ1.1 stakeholder requirements. According to the requirements of stakeholders, comparing the existing tools and then improving the environment in a reasonable way. This could be achieved by importing a new platform (new to the company) or improving an already used platform.

1.4 Research Approach

To achieve the research purposes, detailed in Sections 1.2 and 1.3, three directions were considered:

1. improve the most popular existing tool within the company,
2. import the most suitable tool available, or
3. develop a new prototype based on the identified requirements.

The choice of the most appropriate approach is the first part of the research. This paper is separated into two iterations, the first iteration (section 3) researches stakeholders' requirements and identifies the best way to meet them. Based on these outcomes, iteration 2 (Section 4) implements the identified best attempt.
Iteration 1: concentrates on the first research question, RQ1, including the subquestion, RQ1-1. Stakeholder requirements are solicited and best direction to achieve the project aims decided. The primary approaches of this iteration are interview study and comparison of different tools. In this step, platforms were picked according to the interview study of stakeholders. After that, summarize the final direction decision according to the requirements and the features of compared platforms. Finally, evaluate the outcomes with relevant stakeholders, such as funding and server support staff.
Iteration 2: answer second research question. According to the direction decided in iteration 1, some improvements of the platform in use in the company would been implemented during iteration 2. To achieve the goal of this iteration, discussions with stakeholders were held to obtain targeted responses to particular problems, issues, or solutions. Then formulate use cases according to the discussion results and implement the improvement. Finally, evaluate the improvement with stakeholder feedback and identify if further improvement is required.

1.5 Thesis summary

The main outcomes of this thesis include:

- understand Inner Source development benefits,
- mine stakeholder requirements, and
- identify and improve Inner Source development environment in Ericsson.

Section 2 provides the background of Inner Source development, the current problems faced by Ericsson, related work, and relevant studies. Section 3 mines stakeholder requirements, focusing on research questions RQ1 and RQ1.1, using semi-structured interviews and comparison among dierent potential Inner Source tools. Building upon the outcomes of Section 3, Section 4 implements the chosen direction to provide improved Inner Source environment at Ericsson, and validates the new environment. Section 5 discusses the process, outcomes and specifically the benefits as seen by Ericsson, and Section 6 provides our conclusions and intended future works.

2
Background

This section describes what the Inner Source development is with its description, stakeholder and benefits. Stakeholder structure is introduced as an ‘onion’ model. Description and benefits are discussed according to some paper learning. In section 2.2, relative works are two literatures of the challenges when adopting Inner Source internal the organization, and the factors of Inner Source.

2.1 Inner Source Development

A software project developed by an external-organization community called Open Source. While this community internal an organization called Inner Source [6]. Inner Source is described as a controlled ‘Open Source’, because it only access for the users internal the organization, which could protect the code and related information to avoid the risk of business secrets leaking. In Riehle, Capraro, Kips and HornMany’s research [10], it presents that many companies have tried Inner Source of cooperating software development. The companies include: DTE Energy [11], Ericsson [12], Hewlett-Packard [13] [14], IBM [15] [16], Kitware [17], Lucent [18] [19], Nokia [20] [21] [22], Philips [23] [24] [25] [26], and SAP [27]. Inner Source is a good strategy to support the development inter a company.

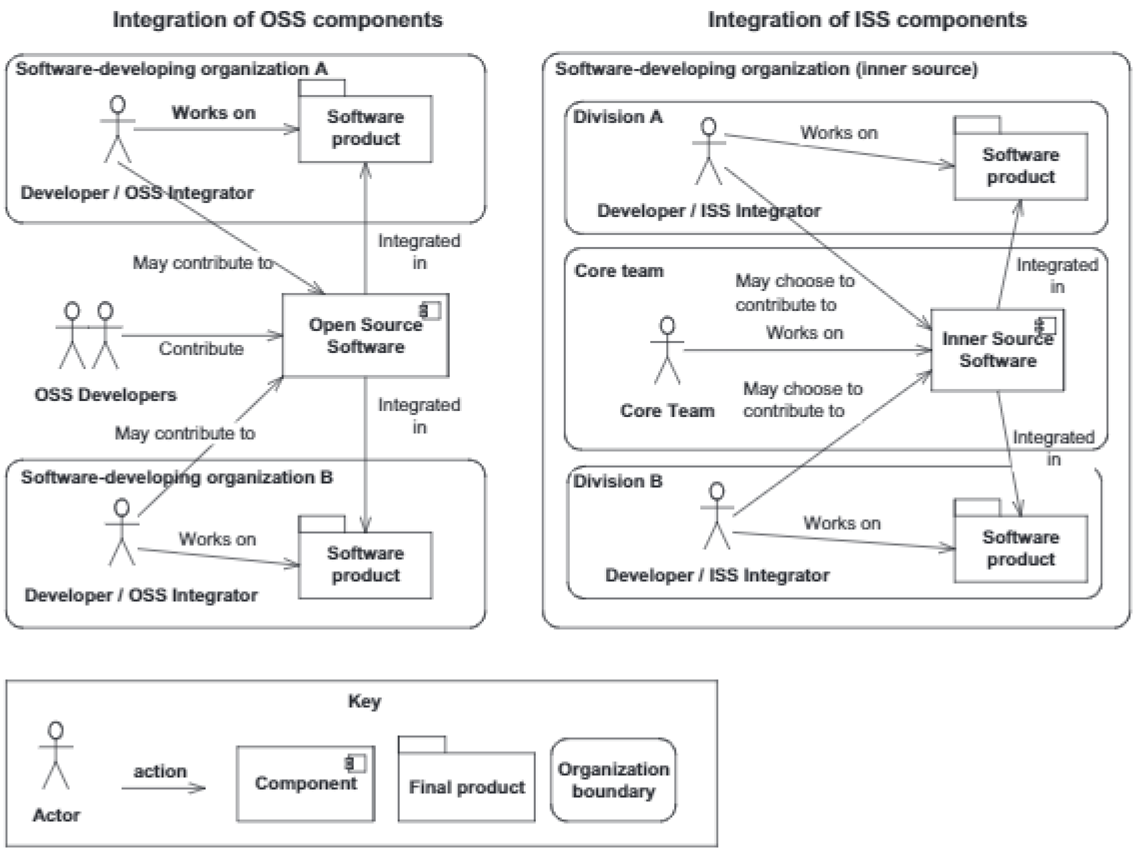


Figure 2.1: Comparison of integration scenarios (Taken from [6])

According to Figure 2.1, the left side shows the integration of Open Source Software components and the right side is the integration of Inner Source Software components. The contributors of Open Source Software are accessing the product worldwide, and it is external of the organizations. Oppositely, Inner Source Software is bounded by an organization. The Inner Source Software is a kind of 'in-house side' Open Source Software.

2.1.1 Stakeholders

An Inner Source community has several stakeholder sorts. For example, individual developers, teams, customers, etc. Thus, all the members of the entire organization can contribute their experiences and resources remotely [7]. Different from normal project development concept, stakeholders of Inner Source Development are available participate in the projects and review the code whenever and wherever they want in organization-wide. The following onion model shows the basic structure of Inner Source project.

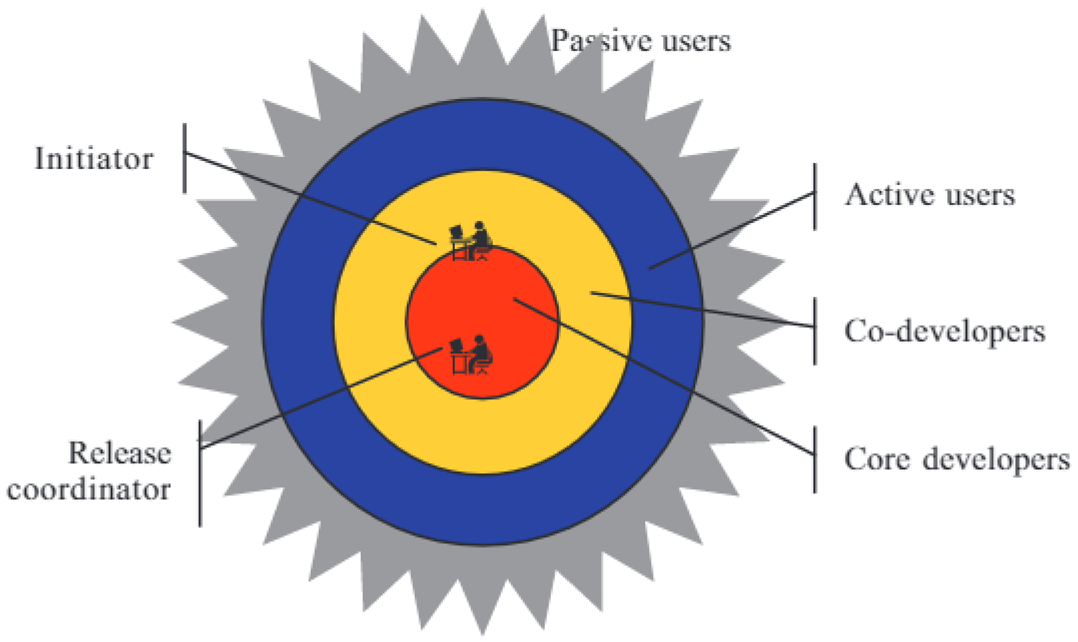


Figure 2.2: The onion model representing the community structure of a typical ISS project (taken from [28])

There are many ways to classify the members in an Inner Source community. Moreover, different organizations have their own names for different member types. In this paper, members are named basically according to this onion model, but some other additional sorts are also be listed.

Initiator
The initiator can be one person or a core team that initializes the project. The initiator is the main direction leader of this project. They have the authority to select the new features to the project.
Release coordinator
The release coordinators are responsibility for managing the version of the project. They release the project with different version and control the features and improvements to be published. They should have a clearly blueprint in their mind. In some scenarios, initiator and release coordinator can be the same person or group. They are named as Project Owner in some cases.
Core developers
Developers are the core part support the whole Inner Source community. Core developers are the developers who contribute the code regularly and supervise design. They deliver new features of a repository with the highest authority to submit the contributions. The core team should be small with a high level of interaction. Otherwise, it is hard to maintain [28]. The core developers must have a deep understanding of the whole project and follow the working cycle and process.
Co-developers

Around core developers are the co-developers. They can become individual contributors, who occasionally modify code or fix bugs which detect by themselves or reported by others. They don't need to understand all the code of the whole project, but the parts they work with. The co-developers may be larger than the core, because the level of interaction is much lower [28].

Active users

Surrounding co-developers is the active users, they are an important part in the Inner Source community. They are the users who pay a lot attention to this project. They use the latest versions of the projects, report bugs and arise some function requests. Tester is one part of active users. They may not contribute the project, but with their help, the project grows better and better. They are the people who innovate the project. Readers can be active users as well, they use the project and try to understand how the project works. They can be the manager of a department or a learner who wants to learn to program.

Passive users

The grey part which seems like the skin of the 'onion' model is passive users. The border of this circle in the model is indistinct, because of the population is hard to account [28]. They are the users in this community, which shows the value of the project.

Stakeholder model of Inner Source Development is very similar with Open Source Development. However, Open Source Development may have more types of stakeholders, such as peripheral developers. Peripheral developers are those who occasionally contribute new functionality or features in the existing system. "Their contribution is irregular, and the period of involvement is short and sporadic" [29].

Comparing to the model of traditional Product-Based Development, there might be no difference between core developers and co-developers. Because all developers are organized as a regular team and product is closed to others who are not in the team. Thus, those contributors are all focusing on the product. They are both the core contributors and active contributors.

2.1.2 Benefits

An Inner Source project can gain a lot benefits with its "transparency, egalitarianism, collaboration, meritocracy and self-organization" [31]. Oram [30] and Dinkelacker et al. [13] introduced some advantages and benefits from adopting an Inner Source strategy.

Reduce cost

In Inner Source Development, projects and scripts are stored in an internal forge, people who have authority to the forge are all available to review the projects and use the APIs. They don't need to w aste time for redeveloping the same things, but focus on the innovation of the new feature releasing which increasing the reusing of code and saving time. Furthermore, Inner Source Development implements technology standardization early in developing process which avoids the cost to review the code and re-standard it.

Complete Documentation
An initial introduction documentation would be set up at the beginning and each commit refers to a short description. The documentations provide the information about the project, and record the behavior of communication, contribution and discussion according to the project. Those documentations recall the memories of members, and helps beginners understand the project and code in a short time, which interests them to contribute to it [30].
Higher Quality
Multiple stakeholders keep eyes on the projects and scripts they are involved in. When reviewing contributions, they can become testers to detect the problems with the contributions and fix themselves or report to contributors to fix bugs. According to Linus's Law: "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone". To be concise as: "Given enough eyeballs, all bugs are shallow" [7]. Large numbers of people motivate the contributors to make a good project. Only one team or one developer may have no ability to fulfill all the requirements, but it is possible with cooperation from all contributors to deliver high quality projects.
Increase Speed
The keys of Inner Source Development are cooperation and sharing. Inner Source Development could get benefits from the large development community with increased speed and faster time-to-market. "Since developers are familiar with a standard set of common tools and infrastructure used within an Inner Source setting, as well as with the available software on the internal forge, developers can be more easily transferred to other projects or products" [13].
Improve Innovation
Inner Source creates a large community for all stakeholders. Developers, users, testers and other stakeholders are all allowed to participate in the project directly. Feedback from organization-wide participant can collect more ideas of different types of stakeholders, which as inspired by Linus's Law as: "Given enough mindsets, all ideas are obvious." [7].
Protect Project
Inner Source Development is organization-wide development. The inner source projects are protected against organization wide, which means they are not accessible outside the boundary.

Inner Source Development applies Open Source Development in a boundary. It has all the benefits which open source has. However, with the respect to Open Source Development, Inner Source protects the projects as 'public' projects in organization-wide boundary and 'private' projects in public which protect the projects. Moreover, in the same organization or company, people have the same knowledge and the same understanding of an issue. It is easy for members to communicate and innovate.

2.2 Relative Work

The findings from some related literatures of Inner Source development will be summed up briefly in this section.

2.2.1 Challenges of adopting Inner Source and Open Source

Srol et al. [38] did a research to analyze the challenges of Inner Source by collecting the data onto in-depth face-to-face interviews for SoftCom with eleven experienced participants. They addressed common challenges of Open Source and Inner Source which helped organizations to improve their development practices by adopting Inner Source and Open Source. They identified 21 challenges of Open Source in total with 6 aspects: Product selection, Documentation, Community, support and maintenance, Integration and Architecture, Migration and usage and Legal and Business. After that they compared the challenges among two types of Inner Source and Open Source. The table 2.1 lists all the 21 challenges in their research. And Table 2.2 is a comparison of relevant challenges among Open Source, infrastructure-based Inner Source and project-based Inner Source.

Table 2.1: Challenges in integrating OSS in product development (Taken from [38]).

Category	ID	Challenge
Product selection	C1	Identifying quality products among the large supply is difficult due to uncertainty about quality (e.g. usability, stability, reliability)
	C2	Lack of time to evaluate components
	C3	Decide what “fork” of the project should be chosen
Documentation	C4	Lack of, or low quality documentation
	C5	Several descriptions of the same component
Community, support and maintenance	C6	Dependency on the community for further support and upgrades; possible need to hire additional talent for maintenance; difficult to control the quality of the support; lack of help-desk and technical support
	C7	Custom changes need to be maintained, which is time-consuming and may cause problems with future versions/community may take a different, incompatible approach.
	C8	Convincing OSS community to accept changes (modifications may be too specific); contributions can be difficult or costly. Difficult to control the architecture if not a core member.
	C9	Uncertainty about product future and consequences for company product.
	C10	Community members would like to have a bigger say in features and integrating final product with company.
	C11	Contributing and investing in OSS project costs resources.
Integration and Architecture	C12	Modifications needed to implement missing functionality or fit into architecture.
	C13	Backward compatibility concerns.
	C14	Incompatibility between components or existing systems.
	C15	Horizontal integration.
	C16	Vertical integration / Mismatch of platform/programming language.
Migration and usage	C17	Complexity of configuration.
	C18	User training/learning costs.
Legal and Business	C19	Complex licensing situation.
	C20	Concerns about, or no clear strategy on Intellectual Property and Rights issues.
	C21	Lack of clear business models that are appealing to industry.

Documentation
This is a very common challenge of Inner Source or even in normal development practice. Lacking of documentation will confuse the beginners and users of how to use the project and how to read it, which will also exacerbate the challenge to attracting the new member's interest. The content of documentation can be the knowledge of the architecture, how data flow works, how to set up the environment of running the project and some tips or standard during developing, etc. In addition, because most of the members are self-organized, it happens that they repeatedly write documentation for the same components. It could happen not only for documentation, but also for the code. However, an actively dashboard could help a lot with these challenges.
Community, Support and Maintenance
An Inner Source community depends so much on the core team. There is no specialized helpdesk and any support for other members, the architecture is important for both core team to review the code later and for new members familiar with the project in a short time. The core team should balance its resources spent on providing implementation of new features on the one hand, and performing architectural refactoring on the other hand [38]. For customer side, they may have some changes in the project according to their own strategies, it may occur mismatch and conflicts to the original release. However, in Inner Source community it is a challenge to get a person to do the maintenance.
Integration and Architecture
In the Inner Source practice, the branches strategy works very well to support development individually, but it produces some architecture conflicts or redevelopment during horizontal integration between new contributions. It takes time to review the code.
Migration and Usage
It is a challenge to have training for both developers and users in an Inner Source practice. It may need a lot of time to learn it by themselves, and because the project is developed by different people, so it's another challenge to find the right person to get proper knowledge for beginners.

Table 2.2: Overview of relevance of challenges to OSS, infrastructure-based Inner Source and project-based Inner Source (Taken from [38])

Category (number of challenges)	**Relevant to OSS**	**Relevant to infrastructure-based Inner Source**	**Relevant to project-based Inner Source**
Product selection (3)	Yes	Yes	No
Documentation (2)	Yes	Yes	Yes
Community, support and maintenance (6)	Yes	Yes	Yes
Integration and architecture (5)	Yes	Yes	Yes
Migration and usage (2)	Yes	Yes	Yes
Legal and business (3)	Yes	No	No
Total number of relevant challenges	21	18	15

Comparing to Open Source, Inner Source is internal of an organization and the project is closed, thus, it has no trouble of legal and business. For project-based Inner Source, members don't need to worry about the product selection. Since the members have only one planned project to develop, so they have no choice to choose other project. In infrastructure-based Inner Source, departments or individuals make freely available software on an internal repository, these challenges could occur [38]. Challenges discussed followed are picked relevant to Inner Source.

According to these challenges, product selection would be a kind of precondition in an Inner Source community. If users cannot find the project they are interested in because of the weakness platform, the community is meaningless. Thus, the project searching feature should be paid close attention on during the implementation. Documentation and information of projects are important as well. Without good documentation and information, other users will never know what is this project about, the project will be closed and unsharable. This is also a good point can be improved during implementation.

2.2.2 Factor to adopt Inner Source

Stol and Fitzgerald introduced nine factors of Inner Source and discussed what software product would be suitable for inner source [39]. They divided nine factors into three categories: Product Suitability, Practices and tools, and People and Management.

Product Suitability
The factors of product suitability are: seed product, stakeholders and modularity.
Seed product is an existing initial implementation of a software product or component [39]. It attracts stakeholders to have a look at this project and evolve into it, in order to make the seed product grows into a successful inner-source product. The seed product should prove the effective of the aimed product of the organization. If the seed product cannot prove its value to attract many stakeholders, it would be meaningless to carry out this product.
The amount of **stakeholders** shows the value of the seed product. A successful Inner Source project requires different types of stakeholders to gain benefits of it. According to the principle of "wisdom of the crowd", if the Inner Source community only has a stakeholder, the project cannot gain the real benefit from it [39]. Different stakeholders could have different requirement and focus, which makes the product become suitable for more scenarios.
Modularity is a basic request for Inner Source project. Owing to its self-organization property, each development only focuses on the own part of the project. Thus, the project should be modularized into different modules for developers to work different parts synchronously.

Practices and tools
Three factors of practices and tools are: development practices, quality assurance and tools.
Development practices of Inner Source project are different from traditional plan-driven project. Inner Source meets a much more flexible way to develop, collaborate and release, because new ideas and new requests from users may come anytime and needed to be reacted quickly. To keep stakeholders, new release shall be delivered very often to keep their confidence in this project.
The **quality assurance** in Inner Source sets it differ from traditional development. In Inner Source, a practice called "peer review" of the code is effectively to get feedback and some bug reports from users and testers, according to Linus's Law [32]. There is another maxim: "release early, release often" [39], which means small releases are more acceptable for users to use and test, and the early releases encourage the feedback early and fix the problems early.
Development tools are important to the whole development process of an Inner Source community. A seeming trivial will lead to break the capability of the set of tools of the organization [39]. In some organizations, especially large-size organizations, different departments have high degree of autonomy, they build their tools as themselves. That means different departments use their own tools, which produces heterogeneous and incompatible to hinder the sharing and collaboration between contributions or even influences the project running on some terminals [40].

People and Management
The factors of people and management are coordination and leadership, transparency and management support and motivation.
Inner Source has a flexible **coordination and leadership**. Anyone who is interested in the project can become a stakeholder of the project, and any developer

who has interest to contribute to the code can be a coordinator. The coordinators are self-organized to assign their own schedule and pick their own tasks no matter the development of new feature, bug fixing, documentation writing, testing or bug reporting.
Inner Source inherits the nature of Open Source; **transparency** is a crucial property of the community. The Inner Source development is a kind of community-based development. Developers collaborate and communicate through the community. The community is open to all the people, who has the permission to this community. So transparency is very common to let others have a knowledge of the project, but maybe not all the teams want to be transparency. Although it is contrary to the original intention of Inner Source, it still happens in normal work. To be transparent, stakeholders can get knowledge from wiki documentations, discuss by creating issues and control their code by different brunches.
Management support and motivation is significant to start an Inner Source project, but it is not easy to achieve. An Inner Source project normally initiated by an initiator or an initiate team (it can be the core group of the project), who has huge confidence to be successful in the project and get perfect repercussions from stakeholders in the future. However, it not always becomes what people expect. It is hard for the manager to judge if the project is meaningful or not.

2.2.3 Common features of Inner Source Development

In this section, some relative work about common features of Inner Source will be discussed. The primary features of Inner Source Development are project hosting [47], project repository clones [47], project planning [2], completely documentation, ownership and merging code.
Project hosting is a basic feature of Inner Source Development. Users are allowed to view code, contribute code, review contributions and maintain the project.
Project repository clones can be a part of project hosting. It means the users who have the authorities are all allowed to clone the repository into their own terminals.
Project planning is very popular in Inner Source Development. Although members of Inner Source Development are self-organization and work individually, they should pick their tasks in advance. There are many tools to do project planning in Inner Source Development. For example, issues from Github and Kanban, etc. The aim is to let members have a clear goal, and avoid doing repetitive work for the same task. Moreover, it guides the direction of the whole project.
Completely documentation sometimes is called project wiki in some organizations. It could provide the introduction, guide to set up the environment of the project, suggestion of reviewing the code and code standard, etc. It helps the beginner to understand the code, and the senior to review the code.
Ownership and merging code are very typical of Inner Source. It uses the branch model to control the ownerships. Each member can have his/her own branch to deliver code. After code reviewing, they can push the code into “master” branch to release. The value of the feature is avoiding conflict among different contributors and some mistakes to destroy the whole project.

3
Iteration 1: Requirement Research

This section is aimed to answer RQ1, including RQ1-1 through semi-structured interviews and some literature background. Comparison of different platforms helps to answer RQ2. The semi-structured interview is aimed to get knowledge of stakeholders' requirements. The purpose of this iteration is to find a suitable tool to import from market or improve an existing tool in organization according to stakeholders' requirements and the result of comparison. Also make the direction decision to support iteration 2.

3.1 Methodology

In this section, the research design and research steps are described. The brief steps are:

- understand the background of Inner Source Development,
- understand the problem and requirements by semi-structured interviews,
- review related literature to have knowledge of the common features of Inner Source Development, and
- compare platforms according to the interviews and platforms in use, and choose a suitable one to work with.

To solve the problem of Ericsson, some semi-structured interviews and group discussions with stakeholders are planned to get their opinions about current Inner Source Development and the Inner Source environment in Ericsson. Then decide to import a suitable in-market tool or improve the one of existing tools in use in the company.

3.1.1 Semi-structured interview

This research collected data by semi-structured interviews with some stakeholders. The semi-structured interview [43] is an interesting method of qualitative research which is collecting data by going side and talk to people. During interview, interviewees are guided to talk about their experiences and opinions about the topic that interviewer is interested in [41]. In semi-structured interview, interviewer and interviewee work as two equal partners which makes both of participants are relax and free to talk. This is a two-way communication; both interviewer and interviewee give and receive information.

In this research, candidates were picked in different departments which covered both employees and consultants, developers and managers. All the candidates knew more or less about Inner Source Development or had related problems. Seven stakeholders from different departments and positions participated in the semi-structured interviews. Four people rejected to attend the interview by some reasons. To reduce nonresponse bias [42], an advanced mail had been sent to interviewees to introduce the agenda of interview. In addition, a short introduction about definition and general benefits of Inner Source Development was presented at the beginning of each semi-structured interview. Several questions were prepared before interviews:

- How you share 'side' projects to others?
- What is your opinion of the existing collaboration platforms in Ericsson?
- What kind of features are you supposed to have in the platform?
- Which roles would you like to play in an inner source project?
- What is your benefit of Inner Source Development?

The average interview time was supposed be one hour. First ten minutes were used to introduce the project and Inner Source Development. Next forty-five minutes were used to discuss the prepared questions. The rest of time, interviewer and interviewee share experience of Inner Source Development and discussed the problems of interviewees. Interview contents (expected features, experience of inner source and their problems) were recorded by notes. The aim of the interview is to get users' experiences of existing platforms, users' expectation of the platform, how users' opinions about Inner Source and what is their problem during sharing projects.

3.1.2 Comparison among existing platforms

To work with the Inner Source tool, it is very necessary to have some knowledge about the existing tools. The aim of comparison among existing platforms is to find a best solution to set up an Inner Source Development environment. The processes of comparisons are:

1. point out the existing platforms are used in Ericsson or people interested in,
2. get interviewees' experiences of the platforms,
3. get familiar with the platforms by using it and reading documentations,
4. summarize the benefits and drawbacks of each platform, and
5. match up the requirements with each platform.

Comparisons were made with the help of all the interviewees with their experiences of existing platforms during interviews. In interviews, the first two prepared questions described in section 3.1.1 are related. Interviewees were free to share their opinions of platforms they used internal and external of Ericsson. Moreover, they were welcome to talk about the platforms they have used for Open Source Development. Some of the interviewees recommended some platforms they heard about or the platforms they were interested in which could be imported into the company.

Since the key points of Inner Source Development are sharing code and collaboration during development, and all the platforms are based on web-based code collaboration. Thus, in section 3.3.4 (the result of comparison among existing platforms), searching function will be paid the most attention. In addition, Table 3.2 compared different main features with four aspects: searching, sharing, collaboration and other quality issues in a table which helps with matching up the platforms with the requirements from stakeholders.

3.2 Data Analysis

The data collected from semi-structured interviews were recorded briefly during interviews. After that all the data were analyzed to answer each interview question. The most important mission was to mine stakeholders' requirements, and get their experiences on different platforms of Inner Source. Thus, according to the interview data, the requirements were summed up as requirement specification in section 3.3.3. Moreover, the mentioned platforms which were collected from the interview data were compared in section 3.3.4 with the tool descriptions and interviewees' experiences.

The data analysis in 4 phases:
Phase 1: Note down the key concerns during all the meetings, including semi-structured interviews, evaluation meetings, group discussion and validation meetings. This phase was important to record the initial ideas and familiarize yourself with it. An example of this phase below, which was taken from semi-structured interviews was to know the stakeholders' requirements:
"I will be retired in few months. I want to hand over the project to someone else in the company, so that the project can be maintained and kept running. This tool is very useful. There are over 1150 users currently. Not only inside the company, but outside as well. I submit it on an open source community, but still some features are for Ericsson only. I have tried to ask someone if they are interested in, or if they know anyone is interested in. However, it is hard and not that efficient. Yes, I found 5 persons who want to learn this tool, but they gave up with some reasons. So I must find someone else."

Phase 2: Generate the initial ideas and take the key themes. Because the initial ideas were very long with some unnecessary information. The theme taken from the example above:
Theme: To keep the tool running, a person was needed to keep maintain it.
Why: The tool owner will be retired.

Phase 3: Mine the requirement from themes. The aim of semi-structured interviews was to know the requirements from stakeholders. According to the example above, two functional requirement were raised up because the project owner need the sharing feature to share the project and other users need the searching feature to find the project.
The requirements mined from the example were:
FR1: Project search
DESC: The tool is able to show the result of related projects when users input key words.

FR6: Project sharing

DESC: The tool should let project owners to submit the projects.

Phase 4: Add dependencies. A requirement can have dependencies on other requirements. The dependence can be 'None', if the requirement is not dependent on any requirement. This phase was aimed to find the relationship between requirements. For example:
: Since project can only be searched if it is published. Otherwise, there is nothing can be searched, so FR1 is dependent on FR6. Then the dependence is added as:
FR1: Project search
DESC: The tool is able to show the result of related projects when users input key words.
DEP: FR6.

3.3 Result

This section places the result of semi-structured interviews and the comparison among Eforge, OpenALM, Gerrit, Gitlab and Github Enterprise. The requirement specification is formulated according to the requirements taken from semi-structured interviews. In this section, a subsection of literature background to introduce some practices which apply agile development with Inner Source Development. Those views provide some inspirations of features which are good to have in the Inner Source platform.

3.3.1 Semi-structured interviews

The semi-structured interview [43] is a very important beginning to preliminarily understand stakeholders' needs.

In the semi-structured interviews, three existing tools of Ericsson were mentioned: Eforge, OpenALM and Gerrit. For the first one, almost all the interviewees avoided using it, since it is hard to start and with unfriendly performance. For OpenALM, someone used it as a backup for projects, but not used it as a collaboration tool. For Gerrit, it is common to developers, but it is not a completely collaboration tool. Although some team use it to collaboration and control the version, it is aimed to be a code review platform. In addition, some of interviewees were using Github to organize projects, but it still produced problem, because it is open-sourced. They are not allowed to submit some product-related project to Github.
For most stakeholders, they want to have a forge ("an extensible Web-based platform that integrates bestof-breed software tools for collaborative software development" [8].) to manage the collaboration among developers and share projects which improve quality and reuse code. The forge shall be searchable, completely information, testability, maintainability and friendly performance (presented by interviewers).
For developers, the requirements are more specific. Owing to the firewalls against different servers, it is hard to communicate between product server and normal work environment. Thus, the forge is hoped to work as a cloud to communicate in between, that they can avoid copying and pasting side projects manually. Secondly,

the projects on a forge shall be enabling to rollback if some contributors destroy the projects. Additionally, they would be glad if the forge can support a plugin in development environment, such as a plugin in Eclipse. One of interviewees described that he preferred a place which he can store the tools he developed. He stored all the tools offline in his own computer. It is not convenient and need to be sorted out. He would like the forge store the tools in a good way and easy to search.

There were two special cases of interviewees. One of them is a consultant who had worked with Open Source for many years, he was in trouble with the authorities of some projects. As a consultant, he has limited authority to reach the server which stores product projects. In fact, the authority on projects he wants is not related to products, but stored in the same server. Therefore, he wants another inner-sourced place to hold their projects. Another case was from a developer who would retire in few months, he worked with a very good tool to view testing log. This tool had 1150 users in Ericsson and out of Ericsson. He would like to pass the tool to someone else in Ericsson that the tool can keep running after he leaves. The problem is it is hard to find such a person only by recommending from limited people. He needs a platform that he can show his tool and find some people interested in it and join him.

3.3.2 Literature Background

Since many interviewees mentioned they worked in an Agile Development way. They would like to apply Agile Development with Inner Source Development to make their work more efficient. Thus, some literature were picked in google scholar and Chalmers Biblioteket to learn the relationship between Inner Source Development and Agile Development.
Gandomani et al [45] did a systematic research on the relationship between agile methods and Open Source Software Development methodologies. They selected 23 researches to study to answer their 3 research questions. 17 of the studies proved that Agile Development and Open Source Development can share some similar practices. Agile Development is widely used in companies and software organizations. Agile Development and Inner Source Development have some similar concepts, benefits and practices [37]. They can support each other with their common principles and sharing practices.
On-site Customers
In Inner Source practice, the customers are available to join in the project through the Inner Source platform that customers can discuss directly with developers and download the latest version of the project which match properly with the practice of on-site customers of XP (Extreme Programming, one method of Agile Development). Normally, to achieve on-site customers is hard, but with Inner Source, customers can get quick responses with their requirements and questions from contributors.
Code Standard
The practice of coding standard of XP, which matches the Inner Source provision as well. Avoiding the mismatch during merging, the Inner Source stipulates that the

constraint on code standard should be set up before starting up the project.

Small Releases

Both of the Inner Source Development and Agile Development impress small and often releases practice [46]. The value of this strategy is that customers can keep themselves in the development iterations. Customers can show their opinions and feedback in early time, they do not need to wait until the complete project delivered. The misunderstanding and requirements changing can be fixed in early stage. This strategy also gives customers and developers confidence.

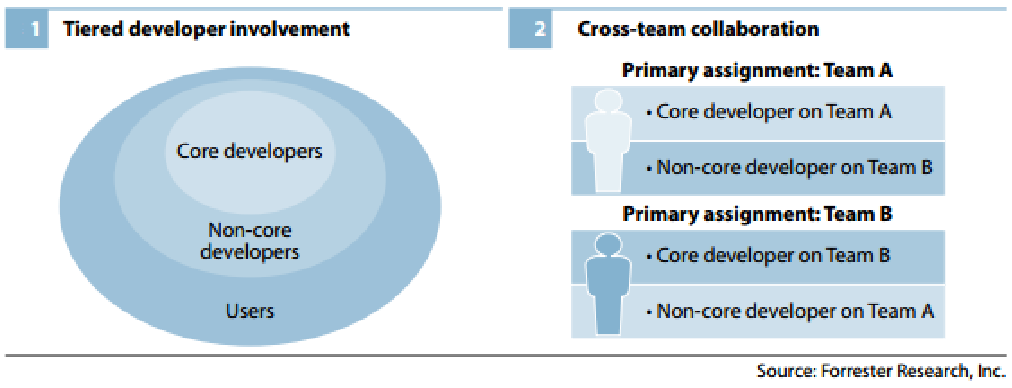

1 Tiered developer involvement
Core developers
Non-core developers
Users
2 Cross-team collaboration
Primary assignment: Team A
• Core developer on Team A
• Non-core developer on Team B
Primary assignment: Team B
• Core developer on Team B
• Non-core developer on Team A
Source: Forrester Research, Inc.

Figure 3.1: Experiment with Tiered Levels Of Developer Involvement (Taken from [46])

According to Barnett's research [46], she raised a strategy of cross-team communication and collaboration to adopt Agile Development and Inner Source Development. As the Figure 3.1(1) shows, a project produced by core developers, non-core developers and users. For cross-team collaboration aspect (Figure 3.1(2)), the core developer on team A also works as a non-core developer on Team B, vice versa. Mostly, the practice happens to one project with two different assignments, but it can still happen to two different projects with two related assignments. The value of this strategy is to push the developers do the assignment instead of do the things they should do [46].

3.3.3 Requirement

The requirements are specified according to the needs from stakeholders. There are 6 functional requirements and 1 non-functional requirements. The section starts with an overall list.

Table 3.1: Overall list of all the requirement

Requirement type)	**ID**	**Requirement**
Functional Requirement	FR1	Project search.
	FR2	Project information
	FR3	Roll back.
	FR4	Auto-test.
	FR5	Authority controlling.
	FR6	Project sharing.
Non-functional Requirement	QR1	Usability.

Functional Requirements
FR1: Project search
GIST: The platform is able to search related projects when users input key words.
STAKEHOLDER: Users who want to find a project on the platform.
SCALE: Relevance of search results to user expectations.
METER: Measurements obtained on view the first 50 results with 50 different key word searching.
MUST: 60 percent of results are relative.
PLAN: 80 percent of results are relative.
WISH: 90 percent of results are relative.
STRETCH: The tool shows 99 percent relevance of the search results to user expectations.
AUTHORITY: None.
DEP: FR2, FR6
Users need this function to find a project they are interested in with some simple words, that they can review projects, use projects or join in projects.

FR2: Project information
DESC: The platform is able to show the primary information of program language, project owner, description, active or not, etc of project.
STAKEHOLDER: Users who want to find a project on the platform.
SCALE: Stakeholders' understanding of a project with project.
METER: Measurements obtained on the understanding level of a well formulated project information on its main page with 50 stakeholders.
MUST: 80 percent of stakeholders can have a basic understand of project.
PLAN: 85 percent of stakeholders can have a basic understand of project.
WISH: 95 percent of stakeholders can have a basic understand of project.
STRETCH: 99 percent stakeholders can understand the project very well.
AUTHORITY: None.
DEP: FR6
Users need this function to review the information of the project and have a basic knowledge the project weather it is the one they are interested in.

FR3: Roll back
DESC: Project should be able to roll back into previous versions.
STAKEHOLDER: Developers and project owners who want to roll back projects.

SCALE: Completion of a project roll back.
METER: Measurements obtained on the completion on rolling back a project to any previous version with 50 project members.
MUST: 70 percent of stakeholders roll back the project to any previous version successfully.
PLAN: 80 percent of stakeholders roll back the project to any previous version successfully.
WISH: 90 percent of stakeholders roll back the project to any previous version successfully.
STRETCH: 95 percent of stakeholders roll back the project to any previous version successfully.
AUTHORITY: Project members who access to change the code.
DEP: FR5
Users need this function to roll back the project into previous version to protect the project if some users destroy the current release.

FR4: Auto-test
DESC: Each contribution shall be testing automatically after submitting.
STAKEHOLDER: Developers who want to have a automatically test after they submit.
SCALE: Success of testing a project automatically.
METER: Measurements obtained on the success times of testing 50 submitted contributions automatically.
MUST: 80 percent of contributions are auto-testing successfully.
PLAN: 85 percent of contributions are auto-testing successfully.
WISH: 90 percent of contributions are auto-testing successfully.
STRETCH: 95 percent of percent of contributiosn are auto-testing successfully.
AUTHORITY: Project members who access to change the code.
DEP: FR5
Users need this function to verified the contributions automatically to reduce the human testing work.

FR5: Authority controlling
DESC: The tool should have authority controlling.
STAKEHOLDER: All the users.
SCALE: Ability of users to get the right authority to use the features of the platform.
METER: Measurements obtained on the ability of users to get the authority of a project as their expectations.
MUST: 80 percent of stakeholders can get the authority of a project as their expectations.
PLAN: 90 percent of stakeholders can get the authority of a project as their expectations.
WISH: 95 percent of stakeholders can get the authority of a project as their expectations.
STRETCH: 99 percent of stakeholders can get the authority of a project as their

expectations.
AUTHORITY: None.
DEP: None
The platform need this function to detect different types of users (e.g. administrators and normal users), and different types of users should have different authorities.

FR6: Project knowladge sharing
DESC: The tool should let project owners to submit the projects with full of information they want to publish.
STAKEHOLDER: All the users.
SCALE: Information of a project can be published as users' expectations.
METER: Measurements obtained on the ability of users to publish all the information of a project as their expectations.
MUST: 80 percent of information can be published as stakeholders' expectations.
PLAN: 85 percent of information can be published as stakeholders' expectations.
WISH: 90 percent of information can be published as stakeholders' expectations.
STRETCH: 95 percent of information can be published as stakeholders' expectations.
AUTHORITY: None
DEP: None
Project owners need this function to publish their projects through the tool that other users can find project, understand it and join into it.

Non-functional Requirements
QR1: Usability
DESC: The tool should be easy to use.
STAKEHOLDER: All the users.
SCALE:Time required to finish a specified task without human training. METER: Measurements obtained on the time cost of finishing a common task with 50 stakeholders during interface test.
MUST: Up to 8 minutes are used to finish the task.
PLAN: Up to 6 minutes are used to finish the task
WISH: Up to 5 minutes are used to finish the task.
STRETCH: Up to 3 minutes are used to finish the task
AUTHORITY: None
DEP: None.
This requirement is needed to make sure the interface is friendly usable, understandable and learnable for users.

3.3.4 Comparison

To identify the suitable forge, comparisons be made among five alternates - Eforge, OpenALM, Gerrit, Gitlab and Github Enterprise. First three have been used in Ericsson. Gitlab has been used in some departments. And Github Enterprise is which developers are extremely interested in to import into Ericsson.

Eforge[1]
Eforge is an Inner Source platform which supporting agile way to work and collaboration on programming which provides source code revision control, such as Git and Subversion. It is a forge for product development in Ericsson. Eforge costs approximately 1000SEK per user per year. The cost is applied once the account is created, regardless of usage.
Benefits:

- It has a high authority system to protect the projects. Since Eforge is faced to the product development. To protect the product security, users are asked to send request to join projects and access to the source code.
- Eforge shows project statistics graphically. It helps users to judge how active the project is.
- Eforge provides forums that all the participants are available to create topics and discuss.
- There are six priorities to trackers - the highest, high, medium, low, lowest and none. Project members can set the priority to all trackers.

Drawback:

- Eforge pays more attention of regular team and product development. Unless the person has the authority on the project and knows how to find the project, otherwise it is hard for another person who is not in the team to find the project.
- Eforge supplies very limited information during searching and on project homepage.
- Eforge provides very low performance. It is hard for new talent to start.

OpenALM[2]
OpenALM is an Open Source Suite for Application Lifecycle Management in Ericsson which powered by Tuleap. It is a free tool for users. Project owner, project managers and team members are all able to join to the project to plan and monitor releases. Some departments use OpenALM to back up their projects.
Benefits:

- All the projects are classified into different categories in “Software Map”. Users are easy to find how many projects are available in different categories.
- There are four models provided in OpenALM - Tuleap model, Scrum model, Waterfall model and Kanban model. Users can choose different project model depend on their workflow.
- OpenALM provides the information of development status (planning, pre-alpha, alpha, beta and production/stable).

Drawback:

- Projects are asked to be validated by system administrator. It takes time to wait until the project passes the validation.
- OpenALM tries to handle too much features which make users confuse to use.
- Many features need to be activated by administrator manually.

[1]https://eforge.ericsson.se/sf/sfmain/do/home
[2]https://openalm.lmera.ericsson.se/

- The performance is not good enough which makes users feel tired to read.

Gerrit[3]
Gerrit is a web-based tool integrates with Git which helps team with their code reviewing. Comparing to rest tool in this section, Gerrit has very limitation available on team collaboration. Gerrit is developed by Google and it is free for users. Developers are allowed to review each change and score it. Large amount of developers in regular teams used Gerrit in Ericsson. This is why even if it is not a completely Inner Source tool, it is still listed in the comparison table.
Benefits:

- Search results are organized into a table with shows the status, owners, branch, project name, updated time, code reviewed and verified or not.
- Gerrit supports the automatically verification.
- Gerrit provides the history of the contribution.

Drawback:

- Hard to find a project by fuzzy query, only if the developer has knowledge of the project.
- There is limited documentation for the user who has not joined in the project.
- There is no good GUI performance.

Gitlab EE[4]
Gitlab is a web-based version controlling platform which provides similar services to Github. It is open-sourced. It supports Gitlab CE for community editions and Gitlab EE for enterprise edition. Gitlab EE has more features and commercial support.
Benefits:

- Maintainers are allowed to build their own Gitlab platform flexibly.
- It cost lower than Github Enterprise: 39 or 99+39 dollars/user/year.
- Developers are allowed to import their projects on other platform into Gitlab account.

Drawback:

- There is not so much information shows in search results.
- Customers have to maintain the platform by themselves.

Github Enterprise[5]
Github is a very famous open source development platform among developers. It works based on a Git repository hosting service. The enterprise version is available for organizations or companies to organize their own Github Inner Source development platform on cloud server or on in-house server.
Benefits:

- It provides a good GUI which helps developers program and contributes in an easy way.
- It supports much integration which helps developer works more conveniently.

[3] https://gerrit.ericsson.se/
[4] https://about.gitlab.com/
[5] https://github.com/business

- It supports full documentations of wiki, discussion history.
- It supports plugins in many development environment such as eclipse.

Drawbacks:

- There is no special license for readers who will not contribute (e.g. managers).
- It costs 225 dollars/user/year, which is very expensive.

The table 3.2 below shows the main features comparison among those five platforms from all the requirements listed in section 3.3.3, and a cost section added in the table additionally. A point between 0 to 5 in each field of the table shows the level of how platforms support the requirements. The larger the point is, the better the platforms support. If the point is 0, it means the platform can not support this requirement. The point is put according to the information of the platforms and interviews with stakeholders.

Table 3.2: Comparison among different platforms

Requirement		Eforge	Open ALM	Gerrit	Gitlab EE	Github Enterprise
Searching	Search by key words	2	2	1	5	5
	Fully information of searching result	1	3	1	4	5
	Advanced search by filters	0	3	1	5	5
Sharing	Projects can be forked	0	5	0	5	5
Information	Completely documentation	2	3	1	4	5
Roll back	Contributions are able to roll back	5	5	5	5	5
Auto-test	Verify contributions	4	4	4	4	5
Authority	Different authority for members	4	3	2	5	4
Usability	Learnability	1	3	3	4	5
Cost	Cost for license	3	5	5	2	1
Total points		22	36	23	41	45

According to the total points of all the five platforms, Github Enterprise get the highest points, which is considered to be the platform to work with. However, according to its highest cost, we make the second choice for Gitlab EE for backup. Those two options are made for the importation direction (the second direction in section 1.4). For the improvement direction (the first direction in section 1.4), improve Gerrit is a good option, since Gerrit has the largest amount of users in the company and it is an Open Source project with a lot of room to improve.

3.4 Discussion

Eforge
Based on Table 3.2 and interview data, Eforge contains many projects, but according to interviewees' discussions, they had not used it very often as an Inner Source tool to cooperate and share code. The voice was heard during semi-structured interviews:

- *"Normally we don't use Eforge to share code. It's hard to use it, and it charges cost when it gets a register user. You need to login to access the share asset, you can do nothing if you are a guest."*
- *"We use it to back up our products, because it can hold large project."*
- *"It seems the tool is maintained by another organization (Tuleap)."*

Eforge is used as a backup tool for their product, because of its low usability and cost. Thus, it is not very common to developers.

Open ALM
OpenALM is a free Inner Source tool in Ericsson. It contains many features to support Inner Source practices.

- *"OpenALM tries to integrate all the features on it, which makes it look a mess."*

Interviewees were not like OpenALM, although it shows the tool has many users. It has too many features which make it too complicated for users to adopt it.

Gerrit
Gerrit is quite popular with developers in Ericsson, people used it as their team cooperating tool in regular work.

- *"Gerrit can work well with the changes controlling, we use it very often."*
- *"Gerrit only takes care of changes(commits), but it doesn't care about projects. It hasn't a common project search as Github does."*
- *"I do not think Gerrit can work as an Open Source tool in the company. You cannot find a project which you want to contribute as an individual. You can reach the projects or changes only if you know it."*

Most interviewees used Gerrit more or less in their work. However, it's hard for users to find a project if they have no clear knowledge of it. For example, the project ID, owner name, etc. For a good Inner Source tool, users should be able to search a project they want with some key words. Gerrit is far to become an Inner Source platform, but many interviewees were happy to see if Gerrit become more Inner Source.

Gitlab EE
Gitlab is used in many com
panies and organizations, it is an open source tool, but for enterprise version, it is charged. Moreover, as an open source tool, it is customizable for organization to build their own tool and maintain it.

- *"I think Gitlab can work well as an Inner Source tool."*
- *"If we choose Gitlab, maybe we need one more team to build it in our company and maintain it."*

Github
In semi-structured interviews, Github was raised many times. Many interviewees had used Github to contribute projects or find useful projects to use. It has a high performance and friendly GUI, but it cost a lot comparing to other tools.

- *"Yes, Github is a very good tool. I contribute so much during my free time. And some job interview is required to show your Github account."*
- *"If we adopt Github, we need a really large fund to support."*

Consider about the comparison, semi-structured interviews the introduction meeting with Github, Github Enterprise is the first choice to be imported.

3.5 Evaluation

An evaluation meeting with some managers was held after the decision was made. On this evaluation meeting, the options were shown and the decision of which platform to work with would be given.
Because the biggest challenge to adopt Github Enterprise was the fund problem, and from the semi-structured interviews, almost all the developers were happy to adopt Github Enterprise. So this evaluation was held among managers to negotiate the fund problem. The evaluation meeting was held by author and her mentor (Dan Berg). The author introduced the background and the comparison, and Dan explained about the cost and the expected user amount. The expected users number was 600, it costs 225 dollars per user per year. Thus, for the company it could raise over one million cost per year in total. It seems a large fund, but with the hourly rate of 750kr per employee, it only cost 3 hours per user. It is worth the investment, if they can gain an efficiency work way and save more than 3 hours per user per year.
After that, attendees still hesitated to accept Github Enterprise. They would like to see users would gain more benefits of Github Enterprise and a more detail comparison between Github Enterprise and Gitlab. With their feedback, a pilot plan was planned to implement both prototypes of Github and Gitlab on the in-house server synchronously to have an evaluation among developers. However, the ITTE (who would create the sandbox for Github and take care of server) said that there is no problem creating a sandbox for Github as such for an evaluation, but they raised concerns about hosting and supporting two software development environments in parallel. It has a price to handle the support, and they had already run Gerrit which could be improved into an similar tool of our purpose. To reduce expenses, to make Gerrit more powerful is an affordable way. Moreover, some teams in Montreal were working with improving Gerrit which could help us. Therefore, according to their feedback, the iteration 2 of the research is to find the way to improve Gerrit and implement it. In next section, the use cases of how to improve Gerrit and how to implement use cases would be presented.

4 Iteration 2: Implementation

This section introduces the second iteration of this research. Iteration 2 describes the use cases to improve Gerrit and implementation of improving Gerrit. Section 4.1 discusses the methodologies are used in this iteration, including the group discussion, use cases, code analysis, evaluation and validation. Section 4.2 presents the results of group discussions. According to the results of group discussions, use cases are listed with their evaluations. Section 4.2.4 is about the implementation of two features. This section ends with the validation of the implementation.

4.1 Methodology

This section describes the research design and the methodologies are used in iteration 2. The brief steps are:

- Review the code, understand the structure of the tool and the features deeply.
- Discuss with stakeholders to understand their experiences of Gerrit and how their expectation of Gerrit is.
- Set up the use cases.
- Validate the use cased.
- Implement the improvement.

To improve Gerrit, firstly understand how dose the tool work, how is the structure of the code and what are the opinions from stakeholders are necessary.

4.1.1 Group discussion

Group discussion is a face-to-face communication, which is understand as a group of people sit together to share experiences and talk about a topic. There are two group discussions of this iteration, which were aimed to generate participants' ideas to improve Gerrit. To reduce misunderstandings, the first one was held after a presentation, the participants were picked widely from the Gerrit users who were interested in improving Gerrit. This discussion lasted half hour. Participants talked freely to express their views. Results of this discussion became the reference to the next group discussion.

The second group discussion is aimed to understand further about how Gerrit could be improved. Before discussion, two features were prepared to discuss and some sub-topics were prepared to guide the discussion as well.

The features could be improved are (refers section 4.2.2):

- Search operators help list.

- Search function.

The sub-topics are:

- What are the shortcomings of Gerrit according to your experience?
- What can be improved with Gerrit?
- What is your opinion about those two features given?

This discussion lasted 45 minutes.

4.1.2 Use Cases

Use case helps to describe the final purpose of a project feature design, which also guides developers to find the solution and achieve the goal. The use case let both designers and developers understand the requirements clearly.
Use cases contains why it is needed, the user stories and how it can be implemented. User stories are also called scenarios which describe requirements of user. Sometimes the requirements are formulated with some ambiguous words which could lead a misunderstand between users and developers. User stories describe the scenarios very detailed to avoiding misunderstanding before the feature starts to be developed. The aim at use case of this research is to make sure both designer and developer understand the requirement correctly. Furthermore, the comprehension of use cases would make the validation more efficient with saving time to explain the features.

4.1.3 Code analysis

The purpose of iteration 2 of this research is to allow Gerrit become more Inner Source to support the users works more efficient. The main emphasis was placed on finding improvement on its usability and functionality. There are two released versions could be involved: version 2.11 and 2.12.2. In Ericsson, the 2.11.x was in used, but the latest version of Gerrit released was 2.12.2. With talked with an expect engineer, the 2.12.2 was planned to be the version which is involved in this research. In addition, Ericsson has its own server and some particular repositories based on the public version.
To analysis the code, reviewed the documentations published on the homepage was the first step to learn its features. Then with some guides from a developer who works with Gerrit, the preliminary knowledge of the code architecture was grasp.
Gerrit is written in Java and some CSS files to deploy the interface. It uses a development toolkit called GWT to build browser-based applications.

4.1.4 Validation

Validation meeting was held to validate the use cases if they were available and reasonable. The Validation meetings were held twice. The first time was held with an administrator of Gerrit and the second time, two more experienced persons were invited to discuss more detail with the use cases. The materials of use cases were sent before meetings to give them some impressions of the meeting if they had some questions, that they could prepare before the meetings. Also, we were welcome to receive their mails if they had some ideas and questions after meetings. Both validation meetings were held around one hour. The meetings went through all the use cases one by one, and the discussion of each use case was held after each explanation. Some helpful and useful advices were collected on the meetings.

4.1.5 Evaluation

The evaluation held in two ways: face-to-face meeting and e-mail evaluation. To get early feedback to continue the implementations, early evaluations held once the features were developed with the mentor by mail. He gave a great feedback on the first feature. The mentor was familiar with the use cases, so he clearly understood what was the purpose of this feature. However, he did not involve in the implementation, so his evaluation was valuable and targeted.
The final evaluation placed after both features were developed. Evaluation mails sent to the Gerrit Developer and the Gerrit project owner in Montreal. In addition, four stakeholders were invited to the evaluation meetings face-by-face. This started with a brief introduction of purpose and background. Then they were given a demo show about the features. After that, they were asked to give the feedback on the strengths and weaknesses for the features.

4.2 Result

This section describes the result of iteration 2 with group discussion (section 4.2.1), use cases (section 4.2.2), evaluation (section 4.2.3) and feature development (section 4.2.4). Use cases are set up according to the result of group discussions. Then the use cases are evaluated with experienced persons. Based on their feedback, two features are developed.

4.2.1 Group discussion

There were two group discussions held to discuss how participants' thought about Gerrit. The first group discussion was taken after a presentation. Over 20 invitations were sent by mail if they were interested in this topic. There were six people attended in total. In the discussion, some ideas were proposed:
"As a user, I cannot create the project as my wish. It needed to be censored by someone else. If I want to create very urgent, I can do nothing except waiting."
"Once we want to organize a public activity, such as Hackathon. We need the administrator to prepare the Git environment in advance to support the activity."
According to this discussion, the strong willing about the project creation was accepted as one of the use cases that can be improved. People also mentioned some about the searching function of Gerrit.
"It is too hard to find a project. I do not think the Gerrit is Inner Source enough to support the collaboration with strangers."
Gerrit has a searching function of some operators to support it, but it still very hard to find a project or a commit by a key word searching. That is a point could be improved as well. However, in this discussion, the search function was talked very

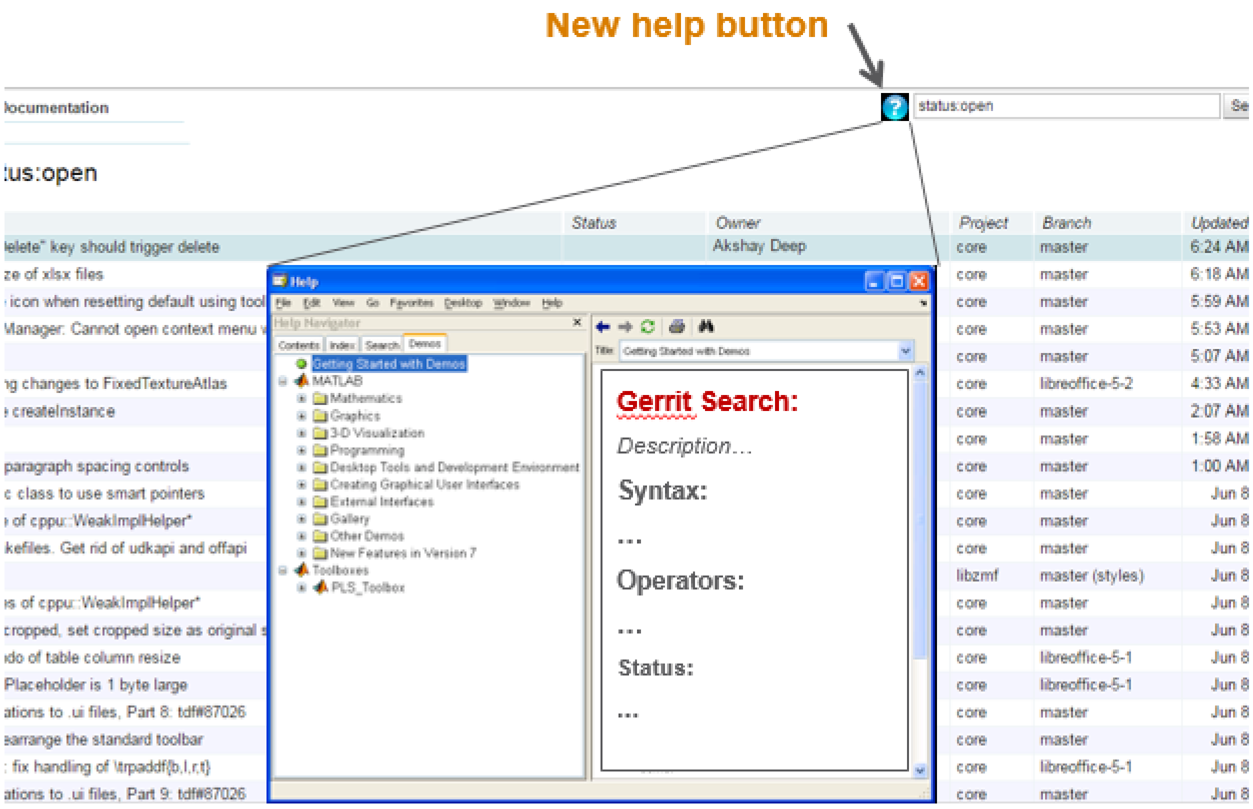
briefly. Thus, the second discussion was brought out.

In the second group discussion, two experienced participants were invited. The discussion started with a short introduction to the topic and an opinion of a search operator helping button.
“I can say the operators are good to help people search the things they want, but they are not good enough. If you cannot keep them all in your mind, it is hard to use them. And if you want to check them all, you need to go to another page. It is not friendly to a user, typically a beginner who even knows nothing about the operators.”
“In Gerrit, it pays attention on the changes, not the projects.”
“It is hard to search a project you are interested in as some Open Source tools do. You can find the projects or the changes only if you know the basic information. For example, the project number.”

4.2.2 Use cases

Gerrit is a good code review tool and Git version controlling for the repository management. It seems is far from an Inner Source platform, but it can be improve to make users gain more benefits from it after improving. For innovative and start-up projects when resources are more loosely coupled or when people just want to find if there are projects others worked on that could support them in their daily work or they could build on or contribute to outside their normal main project, Gerrit does not give so much support. The purpose is to improve Gerrit capabilities to work as an inner source platform, where users can not only work with the projects they know about, but also browse all projects to explore if there is anything that interests them. There are tools today supporting this, like Github. This would facilitate a more innovative and collaborative way of working with smaller projects where the contributors are not pre-defined.
This section is aimed to list the use cases could be improved on Gerrit. Use cases are raised according to the requirements in section 3.3..3

Use case 1: Help button for search operators
This use case is according to QR1 to improve the usability of search function of Gerrit.
Why: Today it is possible to get information about the operators by reading the documentation. However, as a user, I would like to h not understand how to use the search engine in the most powerful way. With a help button, more users would understand and use the operators to filter the information appropriately.
User Story: As a Gerrit user, I would like to find what operators I can use to refine my search for the existing search functionality by pressing a help button. So that it will be easier to make use of the search functionality.
How: Briefly describe: The button, appearance and placement.


New help button
locumentation
status:open
Se
Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Contents Index Search Demos
Getting Started with Demos
MATLAB
Mathematics
Graphics
3-D Visualization
Programming
Desktop Tools and Development Environment
Creating Graphical User Interfaces
External Interfaces
Gallery
Other Demos
New Features in Version 7
Toolboxes
PLS_Toolbox
Title Getting Started with Demos

tus:open

	Status	Owner	Project	Branch	Updated
lelete" key should trigger delete		Akshay Deep	core	master	6:24 AM
ze of xlsx files			core	master	6:18 AM
icon when resetting default using tool			core	master	5:59 AM
Manager: Cannot open context menu v			core	master	5:53 AM
			core	master	5:07 AM
ng changes to FixedTextureAtlas			core	libreoffice-5-2	4:33 AM
createInstance			core	master	2:07 AM
			core	master	1:58 AM
paragraph spacing controls			core	master	1:00 AM
c class to use smart pointers			core	master	Jun 8
of cppu::WeakImplHelper*			core	master	Jun 8
kefiles. Get rid of udkapi and offapi			core	master	Jun 8
			libzmf	master (styles)	Jun 8
s of cppu::WeakImplHelper*			core	master	Jun 8
cropped, set cropped size as original s			core	master	Jun 8
ido of table column resize			core	libreoffice-5-1	Jun 8
Placeholder is 1 byte large			core	libreoffice-5-1	Jun 8
ations to .ui files, Part 8: tdf#87026			core	master	Jun 8
earrange the standard toolbar			core	master	Jun 8
: fix handling of \trpaddf{b,l,r,t}			core	libreoffice-5-1	Jun 8
ations to .ui files, Part 9: tdf#87026			core	master	Jun 8

Gerrit Search:

Description...

Syntax:

...

Operators:

...

Status:

...

Figure 4.1: Help button prototype.

What happen when you press the button:

- A pop-up window with text.
- Does it contain a search field?
- What information will be displayed?
  - Brief description of what can be searched
  - Search syntax
  - Operators – explanation, examples
  - How to stack multiple operators?

Use case 2: Improved inner source Capabilities of searching and information sharing parts in Gerrit

The following sub user cases are included:

- Search function for project names and descriptions.
- Create new projects.
- Add field for homepage (or url) for each project.
- Add full project description.
- Update the project filter to new format.

Use case 2-1: Search function for project names and descriptions

This use case is according to FR1 to improve search function of Gerrit.
Why: To collaborate in a inner source community, as a user it is important to be able to explore the existing projects to evaluate if:

1. the user could contribute to an existing project,

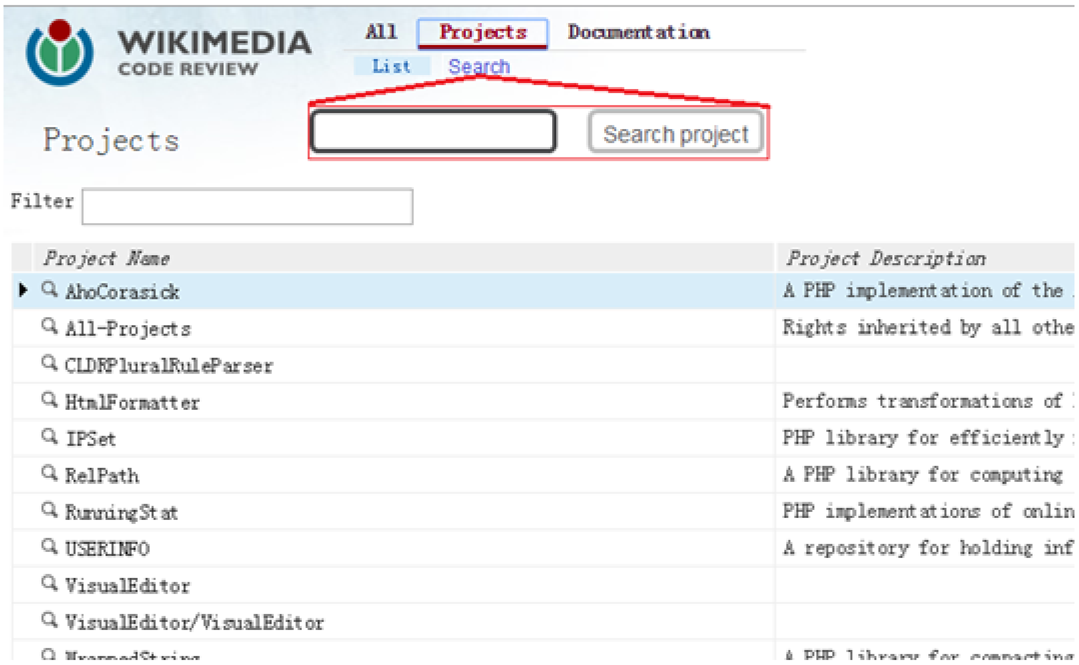
2. base their development on an existing baseline,
3. the user need to start a new project.

Searching from project descriptions and comments will help users to understand more about the related projects, since the title of a project only contains limited information.

User Story: As a Gerrit user, I would like to be able to search for project descriptions, comments and any related information. So that I can understand if this is a program/project/function I would like to use or contribute to developing further.

How: Briefly describe:

- Click the search button in 'project' tab, a dropdown search field will appear for project searching.
- Enter one or several key words, press “search” button, the result list will show with available projects containing the words.
- What kind of information can be considered as the key words: Project description, Comments, Members, etc.



Project Name	Project Description
AhoCorasick	A PHP implementation of the
All-Projects	Rights inherited by all othe
CLDRPluralRuleParser	
HtmlFormatter	Performs transformations of
IPSet	PHP library for efficiently
RelPath	A PHP library for computing
RunningStat	PHP implementations of onlin
USERINFO	A repository for holding inf
VisualEditor	
VisualEditor/VisualEditor	
WrappedString	A PHP library for compacting

Figure 4.2: Project search bar prototype.

Use case 2-2: Create new projects

This use case is according to FR6, which is going to improve the sharing capability of Gerrit.

Why: In Gerrit, users are not allowed to create a new project without permission from the administrator. The reason is to keep good control of the projects so contributors do not destroy others' work. To work in a faster collaborative way, we want users to be able to create new projects themselves in special sandboxes or tools repro. This might need a change in the administration of user permission levels. In Ericsson PC it is solved by using a functional user with rights to some special repros.

User Story: As a Gerrit user, I would like to create the projects by myself, instead of getting permission from administrator. So that it is easier and more efficient to

create new projects.
How: Briefly describe:

- Make users create project directly.
- Users can be identified with different authorities if it needs a high authority to create new projects.
- Administrator can check the new projects after created if needed.
- After pressing the button, users should fill the information on project and a new project created.

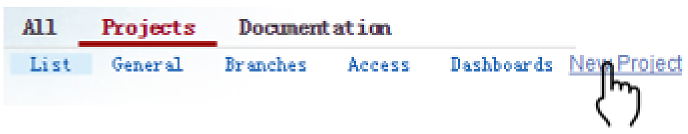
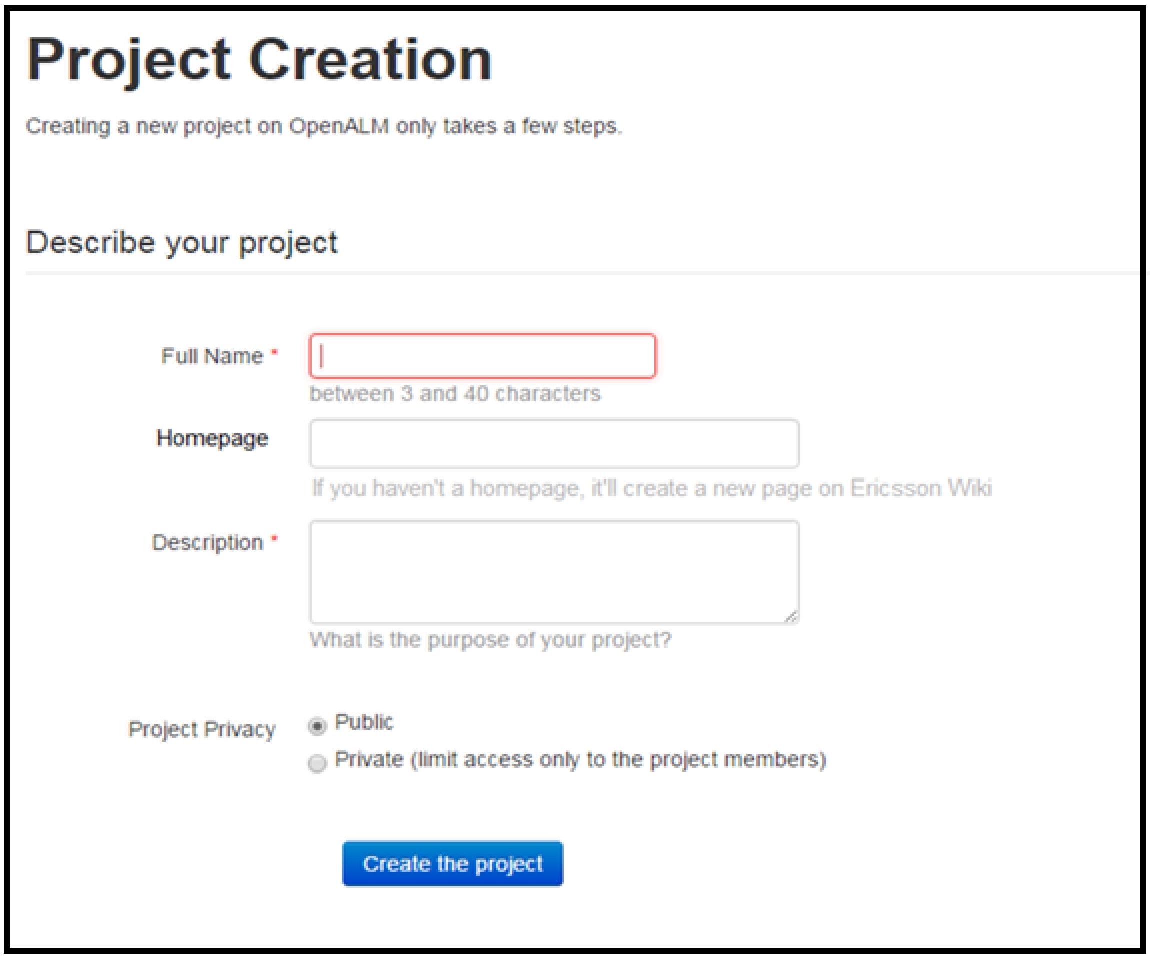


Figure 4.3: New project creation button.


Project Creation

Creating a new project on OpenALM only takes a few steps.

Describe your project

Full Name *
between 3 and 40 characters

Homepage
If you haven't a homepage, it'll create a new page on Ericsson Wiki

Description *
What is the purpose of your project?

Project Privacy
[✓ Public]
Private (limit access only to the project members)

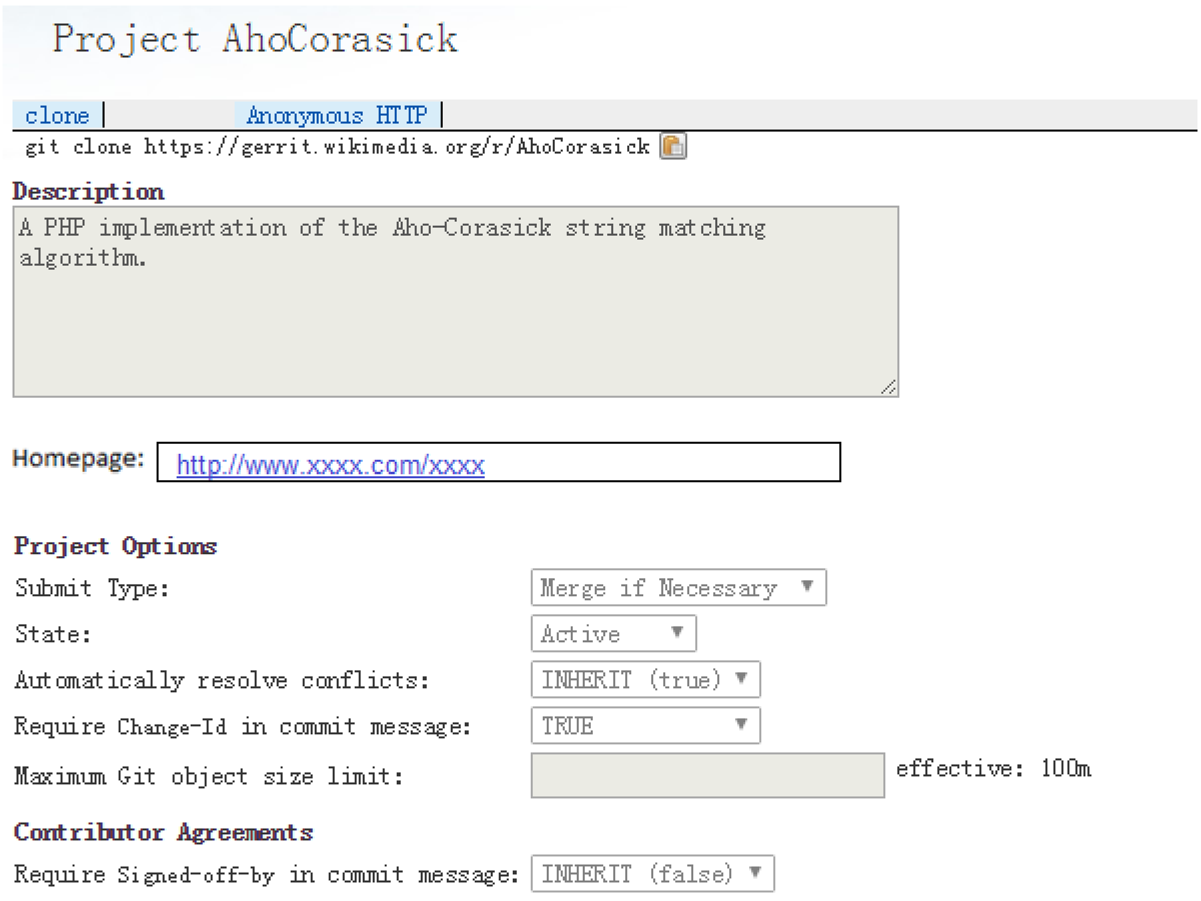
Create the project

Figure 4.4: Project creation page prototype.

Use case 2-3: Homepage (or URL) for each project
This use case is based on FR2 and FR6 to improve the sharing feature and make

the project information more complete.
Why: A good way to get more information about a project is to visit their (main) homepage. To support better utilization of homepages connected to projects in Gerrit, it would be beneficial if the user could add a link to a homepage, in a separate field, at creation of a project instead of entering it in the comments field. If there is a separate field for homepages, it would be easy to present this information when a user searches for information about specific projects.
User story: As a Gerrit user, I would like to have a field linking to a homepage for my project. So that it is possible for interested users to get a more extensive understanding of what my project is doing and how to use and contribute to it.
How: Briefly describe:

- Homepage can be created on for example Ericsson Open Wiki.
- Create a project homepage after a new project was created or input an url that refers to an existing homepage.
- On the project page, users can press the link to go to the homepage and review project information.
- Homepage information should contain: description, project owner, members, language been used in the project and environment, etc.


Project AhoCorasick

clone | Anonymous HTTP |

git clone https://gerrit.wikimedia.org/r/AhoCorasick

Description

A PHP implementation of the Aho-Corasick string matching algorithm.

Homepage: http://www.xxxx.com/xxxx

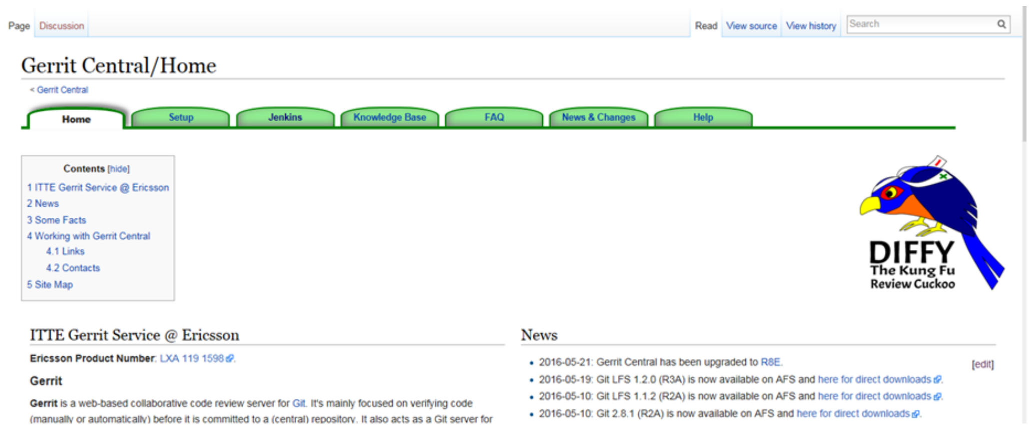
Project Options

Submit Type:	Merge if Necessary
State:	Active
Automatically resolve conflicts:	INHERIT (true)
Require Change-Id in commit message:	TRUE
Maximum Git object size limit:	effective: 100m

Contributor Agreements

Require Signed-off-by in commit message:	INHERIT (false)

Figure 4.5: Project page (homepage blank prototype).



Gerrit Central/Home

< Gerrit Central

Contents [hide]

1 ITTE Gerrit Service @ Ericsson
2 News
3 Some Facts
4 Working with Gerrit Central
4.1 Links
4.2 Contacts
5 Site Map

ITTE Gerrit Service @ Ericsson

Ericsson Product Number: LXA 119 1598.

Gerrit

Gerrit is a web-based collaborative code review server for Git. It's mainly focused on verifying code (manually or automatically) before it is committed to a (central) repository. It also acts as a Git server for

News [edit]

- 2016-05-21: Gerrit Central has been upgraded to R8E.
- 2016-05-19: Git LFS 1.2.0 (R3A) is now available on AFS and here for direct downloads.
- 2016-05-10: Git LFS 1.1.2 (R2A) is now available on AFS and here for direct downloads.
- 2016-05-10: Git 2.8.1 (R2A) is now available on AFS and here for direct downloads.

Figure 4.6: An example for homepage.

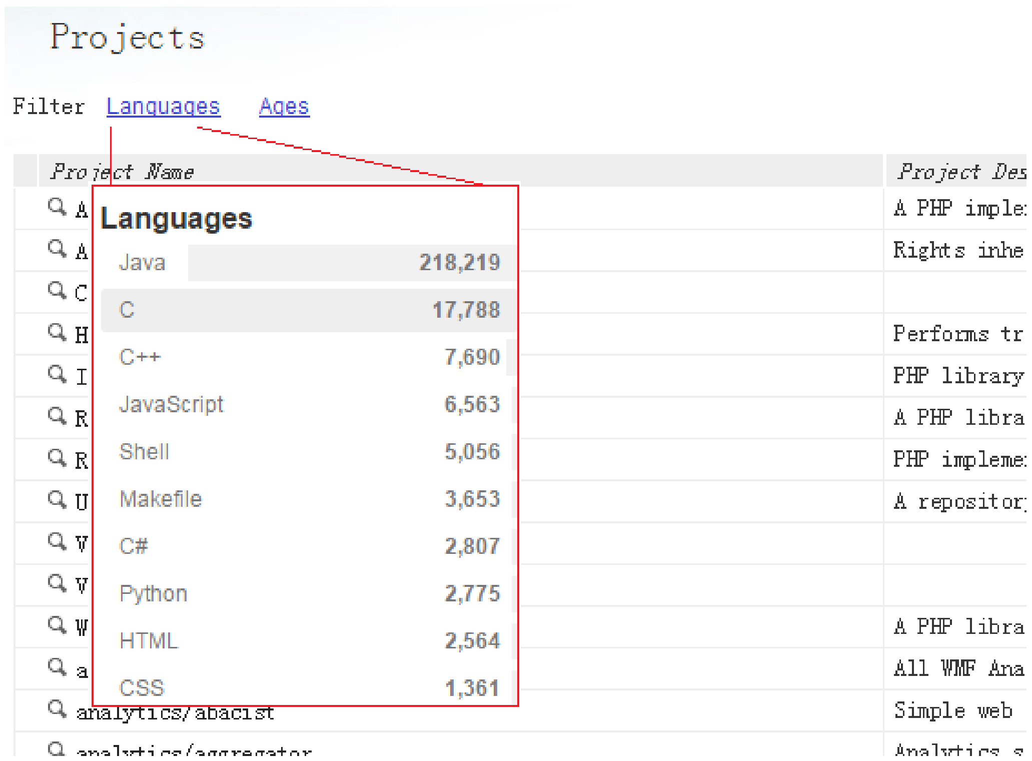
Use case 2-4: Add full description of project
This use case is based on FR2 to make the information of project more complete.
Why: There is a brief description on project page currently. However, many projects are empty of their description. As a project in Inner Source community, it needs description to attract other persons to review the project. If there is no description, others have no idea of what is the project about and how the project works. With complete description, other users can understand the projects and use or contribute the projects.
User Story: As a Gerrit user, I would like to be able to have a project description in a "readme.md" file for wiki files about fully information on the projects. So that it is possible for other persons to read about my projects and understand if they would like to contribute or use my projects.
How: Briefly describe:

- Change the description as a mandatory element when start a project, which force the owner gives some text about that new project.
- Add some wiki files on homepage to guide others to involve into the project.

Use case 2-5: Update the project filter to new format
This use case is based on FR1 to improve the search feature of Gerrit.
Why: There is a text free filter in project list. However, it is more like a key word search of project names. If users have no idea about what the exact name, it is impossible to find the project. With visible and understandable described filters, users can pick the filters to do advanced search.
User Story: As a Gerrit user, I would like to see some choices to filter the projects after I get a long list of searching results. So that I can find more related projects what I interested in more easily.
How: Briefly describe:

- Some preset filters are given to filter the search result instead of a free type box.
- There are two level categories of filters:
  - When click the first level category, the second level category, the second level category will dropdown.

- After pressing different filters of second level category, some results are thrown out from the result list.


Projects

Filter Languages Ages

Languages	
Java	218,219
C	17,788
C++	7,690
JavaScript	6,563
Shell	5,056
Makefile	3,653
C#	2,807
Python	2,775
HTML	2,564
CSS	1,361

Project Name	Project Des
A	A PHP imple
A	Rights inhe
C	
H	Performs tr
I	PHP library
R	A PHP libra
R	PHP impleme
U	A repositor
V	
V	
W	A PHP libra
a	All WMF Ana
analytics/abacist	Simple web
analytics/aggregator	Analytics s

Figure 4.7: Filters of project searching prototype (The second category of language dropdown list is taken from Github.com)

4.2.3 Validation

The evaluation meetings were set up twice. The first time was held with the administrator of Gerrit in Lindholmen. On the first evaluation meeting, the two use cases with their sub use cases were sent in advance. But all the use cases introduced on the meeting as well. He was opposed to the use case 2-4 (Add full description of project) so much.
"You cannot force the users to write something about description. They can choose they leave something to attract people or they do not want to share."
His feedback was reasonable and acceptable. So the use case 2-4 was planned to be removed from the use cases list. To discuss more further, he invited his two another colleague who had contributed Gerrit before.
On the second evaluation meeting, they showed the disagreement with 2-4 as well which strengthened the determination to remove the use case 2-4 of forcing the users to add full description when initial the project. Instead, a warning panel will emerge to point the importance of the description if the users save the project configuration changes without any description. For other use cases, they were happy with them and shared some experiences to contribute the Gerrit.

4.2.4 Development

Two features according to use cases were implemented. There are the help button of search operators and adding homepage url on the main page of a project. The implement result will be introduced into this section.

To develop the feature, clone project from Gerrit Code Review and build the environment was the first step. To build Gerrit, a Linux system is needed. In Ericsson, NX is used as a remote terminal server with Linux system. With the help from Stellan Bondesson, a sandbox was created and the NX environment was built. The brief steps of development are:

- Set up Linux environment (NX).
- Clone Gerrit projects
- Install the build tool of Gerrit (Buck).
- Build Gerrit.
- Review code.
- Develop the features.

4.2.4.1 Feature 1 - Use Case 1

This feature implemented according to use case 1 (help button for search operators). The aim of this feature is to show the search operator list and their descriptions, which helps users finding the operator they want to use during searching. The original search bar is:

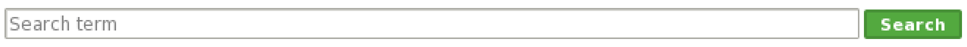
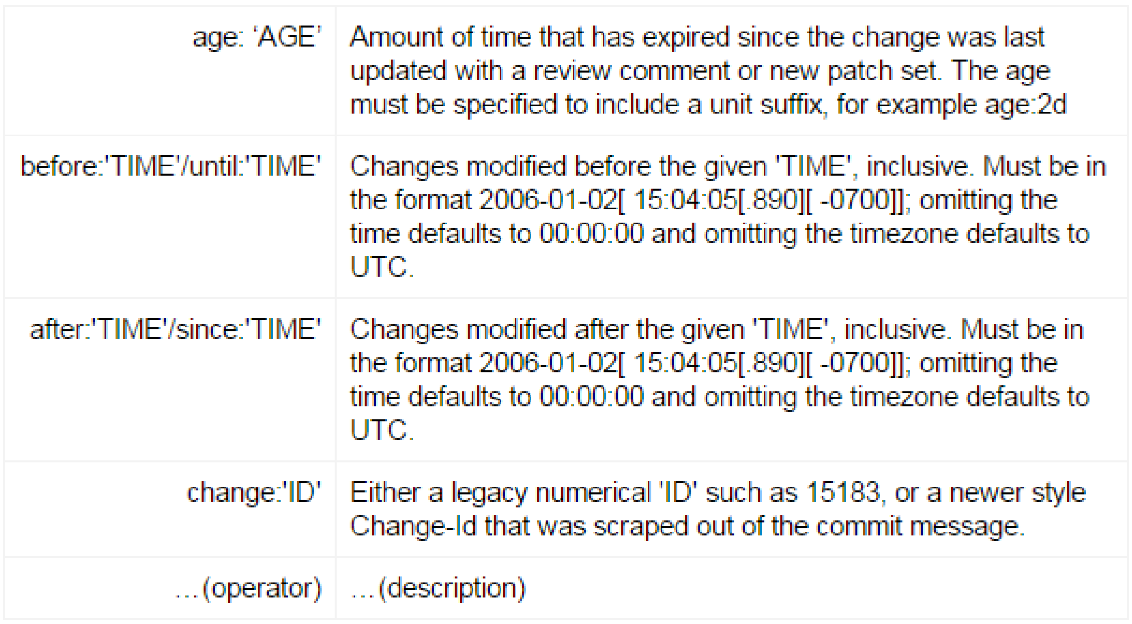


Figure 4.8: Original Search Bar.

The feature can be separated into three parts: help button, operator list and description of each operator. The initial idea of this feature was to list all the operators with their description of a popup table to help users to know the meaning of each operator. The primary design was:

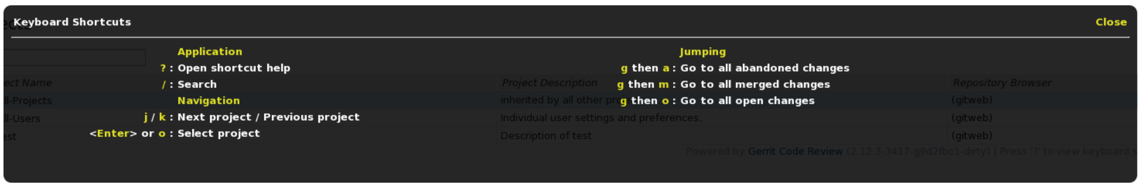


age: 'AGE'	Amount of time that has expired since the change was last updated with a review comment or new patch set. The age must be specified to include a unit suffix, for example age:2d
before:'TIME'/until:'TIME'	Changes modified before the given 'TIME', inclusive. Must be in the format 2006-01-02[15:04:05[.890][-0700]]; omitting the time defaults to 00:00:00 and omitting the timezone defaults to UTC.
after:'TIME'/since:'TIME'	Changes modified after the given 'TIME', inclusive. Must be in the format 2006-01-02[15:04:05[.890][-0700]]; omitting the time defaults to 00:00:00 and omitting the timezone defaults to UTC.
change:'ID'	Either a legacy numerical 'ID' such as 15183, or a newer style Change-Id that was scraped out of the commit message.
...(operator)	...(description)

Figure 4.9: Initial design of search operators.

Considering of the amount of operators is a lot, which will cause the table become very large and users need to scroll the panel to get more information. It is not easy for users to find a specific operator and there are too many words on the panel which will make users feel tired to read it. So the design has been changed into the list of operators, and each operator is an anchor. After pressing the operator anchor, the detail description of normal description, format and constrains will popup. The motivations of operator popup design are:

- The operators are meaningful and readable. Users can guess little bit from the words of operators, then read the detail of the operator to check if the operator is the one they want.
- Simple list of all the operators makes the panel clearer, which reduce the reading quantity when looking for an operator.
- All the operators are classified into different categories, which makes users find the operator more easily.
- Gerrit has a help popup panel of keyboard shortcuts shows as Figure 4.10. To follow the simple design and harmonize design, the same appearance and the same colors will be used in the operator help popup panel.



Keyboard Shortcuts Close

Application
? : Open shortcut help
/ : Search
Navigation
j / k : Next project / Previous project
<Enter> or o : Select project

Jumping
g then a : Go to all abandoned changes
g then m : Go to all merged changes
g then o : Go to all open changes

Figure 4.10: Keyboard Shortcuts help popup panel.

The help button is used as a single question mark before search bar. The initial idea is to use the same button as the search button looks like. However, it will contain too many letters which makes the button too long. Then the button is designed as a grey, simple question mark as the Figure 4.11 shows. The color grey is chosen to distinguish from search button, and this color is not abrupt to destroy the interface.

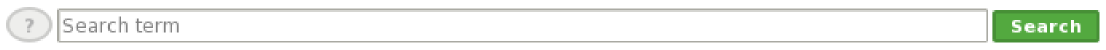
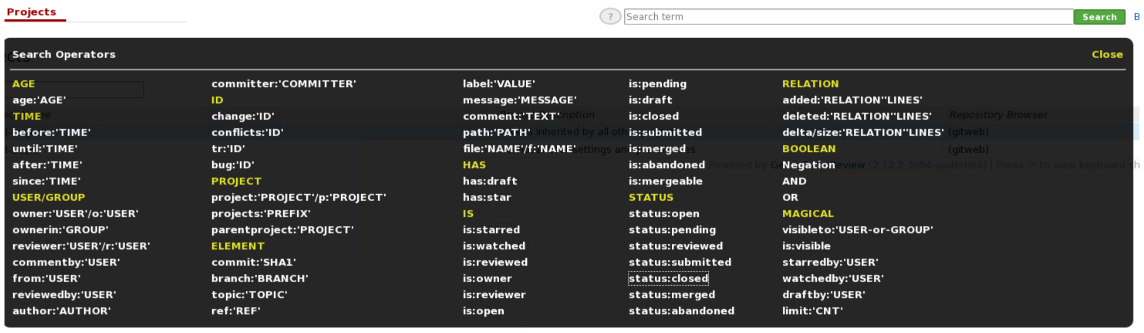


Figure 4.11: Operator Help Button with search bar.

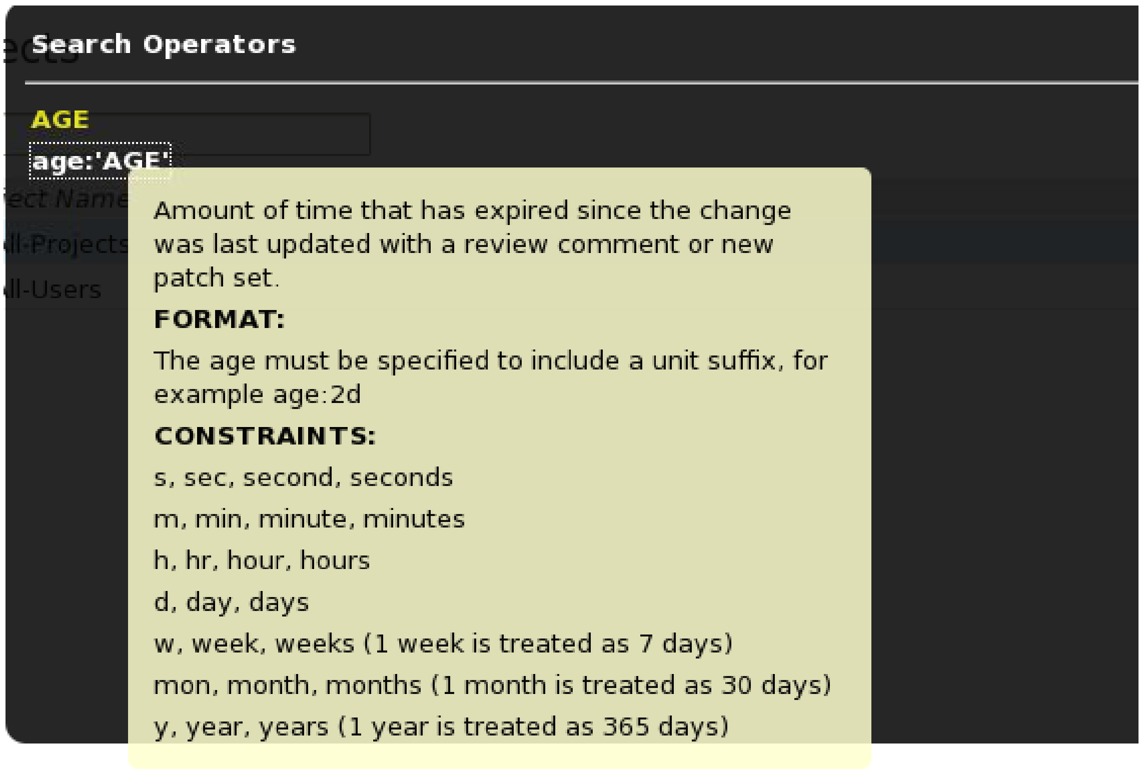
The implementation of operator helps button shows as Figure 4.12 and Figure 4.13. Figure 4.12 is the Operator popup panel and Figure 4.13 describes an example of the operator description.



Search Operators

AGE	committer:'COMMITTER'	label:'VALUE'	is:pending	RELATION
age:'AGE'	ID	message:'MESSAGE'	is:draft	added:'RELATION''LINES'
TIME	change:'ID'	comment:'TEXT'	is:closed	deleted:'RELATION''LINES'
before:'TIME'	conflicts:'ID'	path:'PATH'	is:submitted	delta/size:'RELATION''LINES'
until:'TIME'	tr:'ID'	file:'NAME'/f:'NAME'	is:merged	BOOLEAN
after:'TIME'	bug:'ID'	HAS	is:abandoned	Negation
since:'TIME'	PROJECT	has:draft	is:mergeable	AND
USER/GROUP	project:'PROJECT'/p:'PROJECT'	has:star	STATUS	OR
owner:'USER'/o:'USER'	projects:'PREFIX'	IS	status:open	MAGICAL
ownerin:'GROUP'	parentproject:'PROJECT'	is:starred	status:pending	visibleto:'USER-or-GROUP'
reviewer:'USER'/r:'USER'	ELEMENT	is:watched	status:reviewed	is:visible
commentby:'USER'	commit:'SHA1'	is:reviewed	status:submitted	starredby:'USER'
from:'USER'	branch:'BRANCH'	is:owner	status:closed	watchedby:'USER'
reviewedby:'USER'	topic:'TOPIC'	is:reviewer	status:merged	draftby:'USER'
author:'AUTHOR'	ref:'REF'	is:open	status:abandoned	limit:'CNT'

Figure 4.12: Operator popup panel.



Search Operators

AGE

age:'AGE'

Amount of time that has expired since the change was last updated with a review comment or new patch set.

FORMAT:

The age must be specified to include a unit suffix, for example age:2d

CONSTRAINTS:

s, sec, second, seconds

m, min, minute, minutes

h, hr, hour, hours

d, day, days

w, week, weeks (1 week is treated as 7 days)

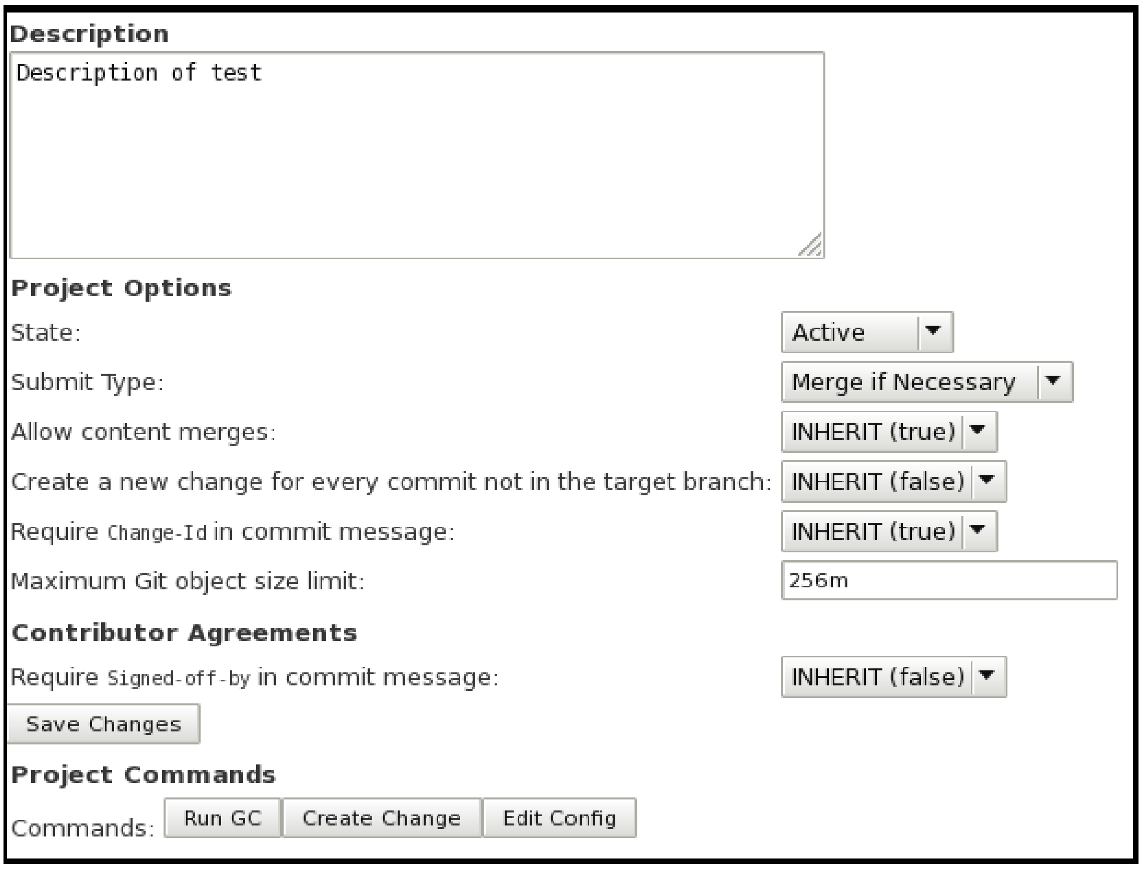
mon, month, months (1 month is treated as 30 days)

y, year, years (1 year is treated as 365 days)

Figure 4.13: An example of the operator description.

4.2.4.2 Feature2 - Use Case 2-3

This feature is implemented according to use case 2-3. This is a feature of adding a homepage blank on the project main page that users can refer the projects to their own homepage outside the Gerrit. Since the project homepage on Gerrit contains very few information on project itself. Thus, if a project has its own homepage to keep more information, this feature gives them a place to refer the link. As the original project main page shows in Figure 4.14, it pays attention to configuration so much. Howeve, to improve the Inner Source capability of Gerrit, more information of project is needed.


Description

Description of test

Project Options

State:	Active
Submit Type:	Merge if Necessary
Allow content merges:	INHERIT (true)
Create a new change for every commit not in the target branch:	INHERIT (false)
Require `Change-Id` in commit message:	INHERIT (true)
Maximum Git object size limit:	256m

Contributor Agreements

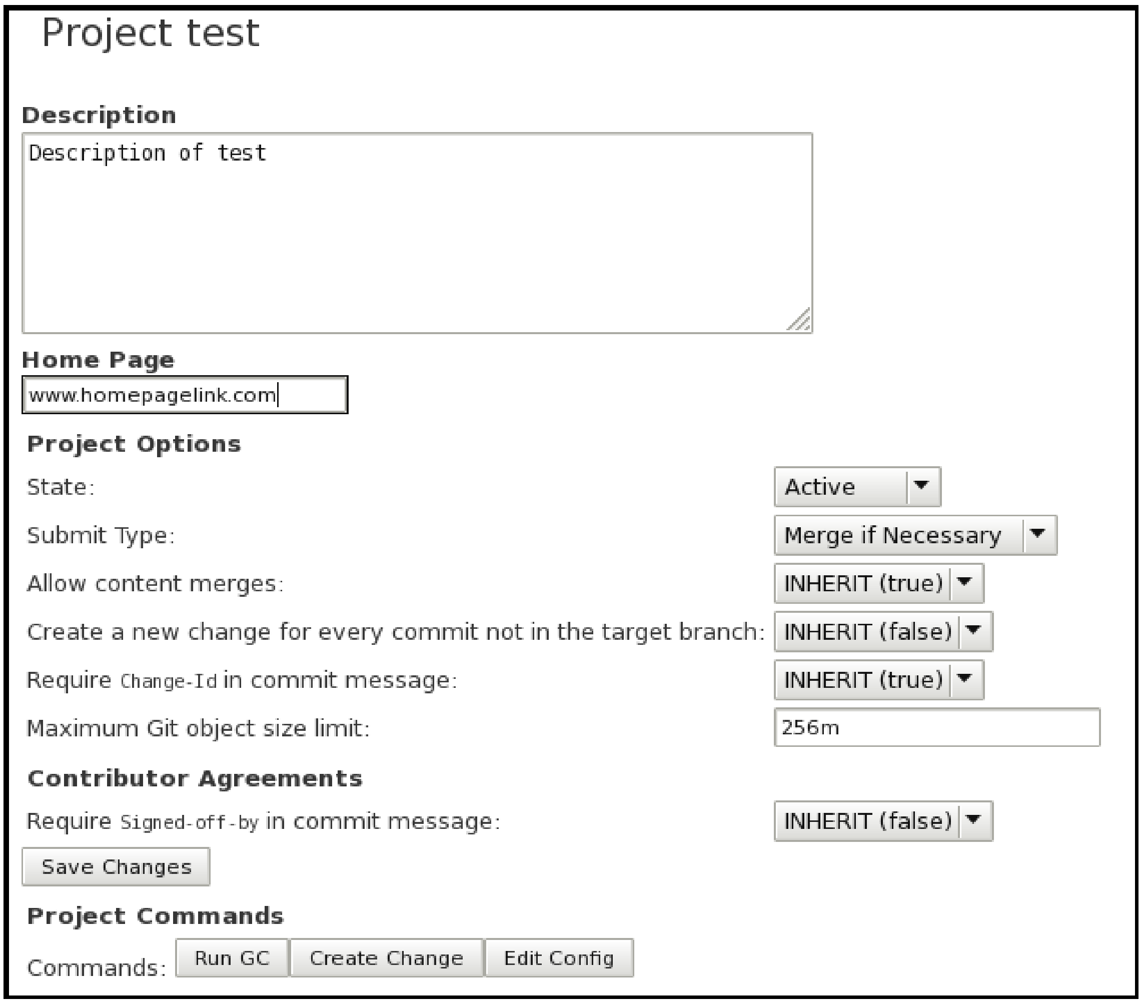
Require `Signed-off-by` in commit message:	INHERIT (false)

Project Commands

Commands:

Figure 4.14: Original project main page.

This feature has less challenge of design than the first feature, but it has more operations need to be fixed behind. The interface design as Figure 4.15 shows. The motivation of placing the homepage blank between description and project options is that description and homepage can be classified into one type (information), others are more like configurations of the project. After entering the link, pressing the 'Save Changes' button, the homepage link can be saved in store.


Project test

Description

Description of test

Home Page

www.homepagelink.com

Project Options

State:	Active
Submit Type:	Merge if Necessary
Allow content merges:	INHERIT (true)
Create a new change for every commit not in the target branch:	INHERIT (false)
Require Change-Id in commit message:	INHERIT (true)
Maximum Git object size limit:	256m

Contributor Agreements

Require Signed-off-by in commit message:	INHERIT (false)

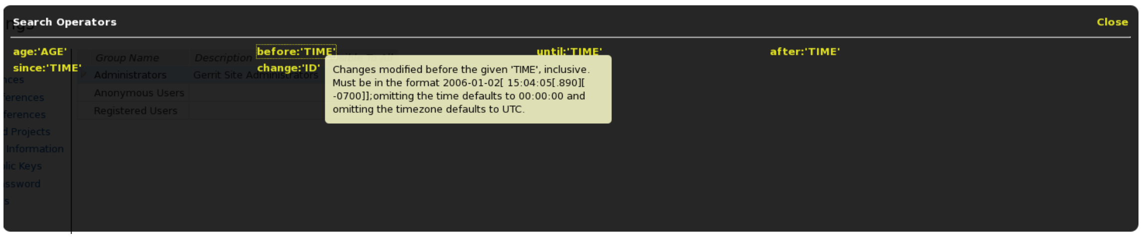
Figure 4.15: Design of project homepage blank.

Gerrit transports data by REST API with Java. For REST API, two project endpoints and two Json Entities were modified. The two project endpoints are 'Set Config' and 'Get Config'. The ConfigInput entity also describes a configuration for a new project. A ConfigInfo entity supplies the configuration to the effective project. This entity is also used to return the configuration called by Get Config, which the request is sent to get a particular configuration of a project. Some fields of ConfigInfo are not available for the users who have no access (refers to refs/meta/config), but for this feature, the access dose not need to be considered. Set Config sets the configuration of a project. A ConfigInput entity is needed in the request body to provide the new modified configuration, and a ConfigInfo entity is returned to response the new configuration. An example of the request and response to both endpoints refers to the Appendix A. To achieve this feature following steps are needed:

- Add a blank on the UI.
- Add a new field in both ConfigInfo entity and ConfigInput entity.
- Set the transport ports of Get Config and Set Config.

4.3 Evaluation

From the early evaluation communication with the mentor, he gave a good feedback to organize the operators and the information. Before that, the operator list and description was shown as Figure 4.16. As the figure shows, the operators were all in yellow, and had not been classified into different categories. The description was not in classifying and copied from operator documentation without formatting.



Changes modified before the given 'TIME', inclusive. Must be in the format 2006-01-02[15:04:05[.890][-0700]];omitting the time defaults to 00:00:00 and omitting the timezone defaults to UTC.

Figure 4.16: Operator list and description (old demo).

His suggestion was to format the description by "Format" and then "Constraints", group operators and reduce the words of description. With his suggestion, operators were all in group and the descriptions were format into a good way (as Figure 4.12 and Figure 4.13 show in Section 4.2.4).
In the final evaluation, unfortunately, the evaluation mails which were sent to the Gerrit Developer and the Gerrit project owner in Montreal did not get response. Otherwise, there are evaluation would be valuable. For the evaluation meetings, the aim was to collect their opinions of the new features if they were friendly performance, understandability and functionality. Since the new features evaluations were not complex tasks, so the evaluation did not take a long time. Not only one participant asked a question that
“What is the added value of having a separate field for the project home page rather that just putting it in the description? ”
The aim of the homepage field was to give users a place to put their homepage link. In addition, because Gerrit only provides a description field for project to introduce itself, and normally when people see a description field, they only think about the description. The homepage field reminds users to create a homepage for their projects to provide more information.
The strengths and weaknesses for both features collected in the meeting were:
Help button for search operators
Strengths:

- Increase the understandability of the search function.
- Convenient to find an operator.
- Friendly Performance with tags.

Weakness:

- Cannot enter the operator into search bar by pressing it.
- No explanation about what the operator means.

Homepage (or URL) for each project
Strengths:

- Increase the functionality of the project controlling.

- Increase the capability for an Inner Source community.

Weakness:

- Cannot press the link to jump to the homepage.

5 Discussion

This section discusses the discussion for each research question, validity and the challenges to this thesis work. The validity section discusses with internal validity, construct validity and external validity.

5.1 Discussion for Research Question

This section discusses the results of the finding of each research question. RQ1.1 is described firstly, and the summary finding of RQ 1 will be presented later. Then the discussion of RQ 2 will be followed.

5.1.1 RQ 1-How does inner source development support team work?

This question is aimed to find a solution to improve the Inner Source environment. To find the solution, understand how Inner Source Development could support people working and how people organized their team work. Thus, this question has a sub-question to get those knowledge. In the semi-structured interviews, interviewees showed their high innovation of how the future tool could support their work, and their opinions of the tools they used. This question is answered in section 3 as the first iteration of this research, to make a decision to lead a precondition of iteration 2. In the sub-question, stakeholders' requirements are specified according to the interviews results. Mining the requirements of stakeholders is a cornerstone of this research. All the improvements were formulated according to those requirements. There are 6 functionality requirements and 1 non-functionality requirements are set up from stakeholders to help improve the Inner Source environment. During the semi-structured interviews, many stakeholders showed their enthusiasm for Github, which caused a big impact on the final decision to import Github Enterprise into the company. However, with the fund problem and ITTE's suggestion, the final decision to achieve the goal was to improve Gerrit, which is in use.

5.1.2 RQ1.1-What are the needs of stakeholders on inner source development?

This question is the main part to answer RQ 1, and it is also important for the whole research. This question is answered in section 3.3.1 and 3.3.3 by the semi-structured interviews and requirement specifications. The

5.1.3 RQ2-How to improve the inner source development environment?

This question is for implementing the improvement based on the answer of RQ 1. This question is answered in section 4, as the second iteration of this research. To implement the improvement, the use cases are formulated with some experts of Gerrit. Simultaneously, those use cases are evaluated by some stakeholders who have experiences of contributing Gerrit. After evaluation, two features of use cases are implemented and evaluated. One of the features would be submitted into Open Source community of Gerrit and another would be submitted into Inner Source community of Gerrit in Ericsson. The rest of features will be implement in the future, which would be discussed in section 6.

5.2 Validity

All the study has its validity. In this study, four validities are discussed in this section. Most validities were described in Wohlin et. al [49].
Nonresponse bias[42] occurs when the meeting invitation was sent or a mail survey was sent without response. This happened many times during this research. Since there were many meetings (semi-structured interviews, group discussions and evaluation meetings) need many participants to attend in this research. All the invitations and some surveys were sent by mail, some positive responses were received, but still many negative responses or even nonresponse happened. Typically, the evaluation mails to Montreal were sent without any response which disappointed us so much. However, in some unofficial communication, those features were discussed. So it will not affect the result so much. And with the negative response from other meetings, there were other participants involved. Nonresponse bias may also happen if the participants don not give any answer during meetings or discussions because they have no knowledge of that topic. To reduce the nonresponse bias, the agendas and topics of the meetings were sent before the meetings, that the participants can prepare a little bit for that. Furthermore, brief introduction was presented at beginning of each meeting and some questions were asked to check if the participants understood or not.

Insight shortage happens if the participants are in a small group, which will lead the others' insights are ignored. In this research, invitations were only sent to the person who gave response. And the invited people were limited by recommending. There is a huge number of employees in Ericsson. It is impossible to do a research which covered all the people, and receive all the responses. We tried our best to collect as many participants as we could. The insight shortage still affected the result, more requirements could be mined if more participants involved in. However, we got a good result as we expected.

Selection validity [49] can happen if the wrong volunteers were chosen to enroll in the research. This validity can occur in many researches. Since we have no idea if the people we choose are the suitable or not before the meetings. In contrast,

the right person we invited may cannot attend to the research. In this research, we cannot promise that all the participants were the "correct" people, but at least, all the participants were related stakeholders. All of them had some knowledge and opinions of this topic. Those who had no idea of this topic or no experience of the topic were not involved in this research.
Specific requirements are one more external validity of this research. Since the research was in the range of Ericsson, all the participants were selected in Ericsson. The stakeholders in the Open Source community were not involved, which led the requirements collected were specific for Ericsson. This validity has related to last validity (Insight shortage). Because the effect of both validities lack wide insights to provide more requirements.

5.3 Challenges

There are some challenges in this research. During semi-structured interviews, some of the interviewees hesitated to accept new tool. In the interviews, a question asked that if they had some recommend for new tool to be imported into the company. Some of the interviewees who had some experiences on Open Source community, gave some recommends, for example, Github, Gitlab, etc. However, some of the other interviewees hesitated to try a new tool.
"Gerrit is fine for me. If there's a new tool coming, I must take a part of time from my normal work to learn it. It can lead many problems because of the users are fresh. We need to fix the problems, and it wastes time."
The main argument was that the new project would change their work style, and they need a training for the new tool. As they have a similar tool, the new tool importation was meaningless. Howeve, they could accept that some more new features to be contributed to the existing tool which makes it become better. The plan for iteration 2 was to contribute Gerrit. The challenges to iteration 2 were hard to get help from others. Since in Ericsson in Lindholmen, I worked alone without anyone else contribute the same features of the tool. The only one that I can get help from was in Montreal. Because of the different time zones, normally I got his response after 5 p.m.(working hour), and it is hard to solve some problem by mail. Thus, a problem always took several days to be solved, or even I lost contact with him. Especially, the final evaluation mail was sent to him without any response in one month. He was supposed to give a good feedback by his experience, and his evaluation should be meaningful. If we start a new similar research in the future, I would find some contributors else in Open Source community, for example, in other companies nearby or in the online group of Gerrit development.

6
Conclusion

The Inner Source Development is used widely in software-related companies with its collaboration, code sharing, version controlling and member management. This thesis was aimed to ameliorate the tool to support the Inner Source community as a target company of Ericsson.

In Ericsson, they have their own way to control product projects, but with some small or medium projects, the project controlling tool cannot support them very well. To learn the requirements of stakeholders, some semi-structured interviews were held after a basic learning of Inner Source by reviewing some literatures and other related research. With the results of interviews, we got the requirements from stakeholders, and some recommended tools they used which could be imported into the company. Three attempts were raised to achieve the research goal:

- contribute the most popular existing tool in company,
- import the most suitable tool in the market, and
- set up a prototype of a new tool based on requirements.

With the company expectation and the results of the interviews, they preferred to have a tool could be used directly, than a prototype full of concepts. Thus, the last attempt was abandoned. Since Github was discussed many times during interviews, we did some researches to compare different tools and tried to import Github. However, with hesitation to the fund problem on the evaluation meeting and feedback from IITE, the second attempt was rejected as well. The only way left was to contribute Gerrit, which was widely used in the company. After got decision to improve Gerrit, several use cases were set up to be implemented. Two of the features were implementing in the end to make Gerrit more 'Inner Source'.

This research is divided into two iterations: the first one was to find out a best way to achieve the purpose which was presented in section 3, the second iteration was to implement the improvement which was described in section 4.

The main contributions of this thesis is to compare 5 platforms with Inner Source requirements and find the suitable one to improve according to the needs of Ericsson situtaion. We have verified that semi-structured interview is a good way to get the knowledge of stakeholders' requirements and get stakeholders' experiences. This thesis gives an example to make a good choice to find a good tool to solve the problem of an company, which matched the requirements with the platform comparison and picked the suitable platform to meet the most stakeholders' requirement through discussion and negotiation. We practical improve the platform we picked with formulating use cases according to the requirements first. As company aspect, we alternative platforms have been compaired and a most suitable tool has been summed up to work with to improve the environment in the company.

6.1 Limitation

The limitation of this research was that all the activities and researches were Ericsson oriented, which lead the voice outside Ericsson was ignored. All the participants were picked from Ericsson, and all the requirements were set for Ericsson. This research result may not suitable for other organizations with different situations and background.

6.2 Future work

Two of use cases were implemented, the rest of the use cases could be developed in the future. Especially, the key words searching function of the projects which is a good point for Gerrit. Since Gerrit has a poor performance of project searching function. As this paper discussed before, Gerrit pays a great emphasis on changes more than the projects. The documentation controlling and project searching is weak in Gerrit. We hope Gerrit could become a good Inner Source tool for Ericsson with fixing the project aspects.
The evaluation in this research was held in the company, but for the feature which will be submitted into Open Source community. To judge it whether it is worth or not, the evaluation for those improvements could be expanded upon an Open Source community. To let more people involve in this project.
In the future, Gerrit capability improvement could be a topic on Hackathon which held among all the employees in Ericsson. A short introduction to the topic could be presented to attract more people in this company to involve in it. Then, they can have the Hackathon days to do some tasks to improve Gerrit to help them working more effective and in a more collaboration way.

Bibliography

[1] Eric von Hippel (2001). Innovation by User Communities: Learning from Open-Source Software. MIT SLOAN MANAGEMENT REVIEW

[2] Martin Höst, Klass-Jan Stol and Alma Oručević-Alagić(2014). Inner Source Project Management. Software Project Management in a Changing World. Chapter 14, Page 343-369. ISBN 978-3-642-55034-8

[3] Nithya A. Ruff (2016). Trends in Corporate Open Source Engagement. Article on Opensource.com

[4] Gaughan G, Fitzgerald B, Shaikh M (2009) An examination of the use of Open Source software 832 processes as a global software development solution for commercial software engineering. In: 833 35th Euromicro conference on software engineering advanced applications (SEAA), pp 20–27

[5] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, Daniela Damian (2008). “Selecting Empirical Methods for Software Engineering Research” in Guide to Advanced Empirical Software Engineering, pp 285-311. ISBN 978-1-84800-044-5

[6] Klaas-Jan Stol. Supporting Product Development with Software from the Bazaar. PhD thesis, University of Limerick, 2011.

[7] Klaas-Jan Stol, Paris Avgeriou, Muhammad Ali Babar, Yan Lucas, and Brian Fitzgerald. 2014. Key factors for adopting inner source. ACM Trans. Softw. Eng. Methodol. 23, 2, Article 18 (March 2014), 35 pages. *DOI: http://dx.doi.org/10.1145/2533685*

[8] Dirk Riehle, John Ellenberger, Tamir Menahem, Boris Mikhailovski, Yuri Natchetoi, Barak Naveh, and Thomas Odenwald (2009). Open collaboration within corporations using software forges. IEEE Softw. 26, 2, 52–58.

[9] Klaas-Jan Stol, Brian Fitzgerald (2014). Inner Source—Adopting Open Source Development Practices in Organizations. In IEEE Software 32(4), doi: 10.1109/MS.2014.77

[10] Dirk Riehle, Maximilian Capraro, Detlef Kips, Lars Horn (2016). "Inner Source in Platform-Based Product Engineering", IEEE Transactions on Software Engineering, no. 1, pp. 1, PrePrints PrePrints, doi:10.1109/TSE.2016.2554553

[11] Smith, P., Garber-Brown, C. (2007, August). Traveling the open road: Using open source practices to transform our organization. In Agile Conference (AGILE), 2007 (pp. 156-161). IEEE.

[12] Torkar, R., Minoves, P., Garrigós, J. (2011). Adopting free, libre, open source software practices, techniques and methods for industrial use. Journal of the Association for Information Systems, 12(1), 88- 122.

[13] Dinkelacker, J., Garg, P. K., Miller, R., Nelson, D. (2002, May). Progressive open source. In Proceedings of the 24th International Conference on Software Engineering (pp. 177-184). ACM.

[14] Melian, C., Ammirati, C. B., Garg, P., Sevon, G. (2002). Building Networks of Software Communities in a Large Corporation. Technical Report. Hewlett Packard.

[15] Neus, A., Scherf, P. (2005). Opening minds: Cultural change with the introduction of open-source collaboration methods. IBM Systems Journal, 44(2), 215-225.

[16] Vitharana, P., King, J., Chapman, H. S. (2010). Impact of internal open source development on reuse: participatory reuse in action. Journal of Management Information Systems, 27(2), 277-304.

[17] Martin, K., Hoffman, B. (2007). An open source approach to developing software in a small organization. Ieee Software, (1), 46-53.

[18] Gurbani, V. K., Garvert, A., Herbsleb, J. D. (2006). A case study of a corporate open source development model. In Proceedings of the 28th international conference on Software engineering (pp. 472- 481). ACM.

[19] Gurbani, V. K., Garvert, A., Herbsleb, J. D. (2010). Managing a corporate open source software asset. Communications of the ACM, 53(2), 155-159.

[20] Lindman, J., Rossi, M., Marttiin, P. (2008). Applying open source development practices inside a company. In Open Source Dev

[21] Lindman, J., Rossi, M., Marttiin, P. (2010). Open Source Technology Changes Intra-Organizational Systems Development-A Tale of Two Companies. In ECIS.

[22] Lindman, J., Riepula, M., Rossi, M., Marttiin, P. (2013). Open Source Technology in Intra-Organisational Software Development— Private Markets or Local Libraries. In Managing Open Innovation Technologies (pp. 107-121). Springer Berlin Heidelberg.

[23] Wesselius, J. (2008). The bazaar inside the cathedral: business models for internal markets. Software, IEEE, 25(3), 60-66.

[24] van der Linden, F. (2009). Inner source product line development. In Proceedings of the 13th International Software Product Line Conference (p. 317). ACM.

[25] van der Linden, F., Lundell, B., Marttiin, P. (2009). Commodification of industrial software: A case for open source. Software, IEEE, 26(4), 77-83.

[26] van der Linden, F. (2013). Open source practices in software product line engineering. In Software Engineering (pp. 216-235). Springer.

[27] Riehle, D., Ellenberger, J., Menahem, T., Mikhailovski, B., Natchetoi, Y., Naveh, B., Odenwald, T. (2009). Open collaboration within corporations using software forges. Software, IEEE, 26(2), 52-58.

[28] Crowston, K., Annabi, H., Howison, J. Masango, C. (2004) Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development, Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research (WISER), pp. 18-26, Newport Beach, California, USA, doi:10.1145/1029997.1030003.

[29] Yunwen Ye, Kouichi Kishida (2003). Toward an Understanding of the Motivation of Open Source Software Developer, ICSE '03 Proceedings of the 25th

International Conference on Software EngineeringPages 419-429, ISBN:0-7695-1877-X

[30] Getting Started with InnerSource. O'Reilly Media, LSI: 978-1-491-93758-7

[31] Guy Martin, Andrew Aitken (2013). Inner Source Webinar Series: Open Source Community Development Methods.

[32] Eric S. Raymond. 2001. The Cathedral the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly Media.

[33] Beck K., Cockburn A., Jeffries R., Highsmith J. (2001). "Agile manifesto", http://www. agilemanifesto.org, 12-4-2002.

[34] Highsmith J., Orr K., Cockburn A.(2000). "Extreme programming", in: E-Business Application Delivery, pp. 4–17. Available: http://www.cutter.com/freestuff/ead0002.pdf.

[35] Conny Johansson, Christian Bucanac (1999). The V-Model.

[36] Ken Schwaber (2004). Agile Project Management with Scrum (Developer Best Practices) ISBN-10: 073561993X

[37] Stefan Koch (2004). Agile Principles and Open Source Software Development: A Theoretival and Empirical Discussion. Extreme Programming and Agile Processes in Software Engineering Volume 3092 of the series Lecture Notes in Computer Science pp 85-93.

[38] Klaas-Jan Stola, Muhammad Ali Babarb , Paris Avgeriouc , Brian Fitzgerald(2011). A comparative study of challenges in integrating Open Source Software and Inner Source Software. Information and Software Technology, 2011, volume 53, issue 12, pages 1319-1336, doi:10.1016/j.infsof.2011.06.007

[39] Klaas-Jan Stola, Brian Fitzgerald (2014). Inner Source–Adopting Open Source Development Practices in Organizations: A Tutorial, in IEEE Software 32(4). DOI: 10.1109/MS.2014.77

[40] J. Dinkelacker et al., "Progressive Open Source," Proc. 24th Int'l Conf. Software Eng. (ICSE 02), 2002, pp. 177–184; doi: 10.1145/581339.581363.

[41] Fiona Fylan (2005). Semi structured interviewing. A Handbook of Research Methods for Clinical Health Psychology. Chapter 6, Page 65-78, ISBN: 97-0-19-85275-5

[42] Thomas Krenzke, Wendy Van de Kerckhove, and Leyla Mohadjer Westat (2005). Identifying and Reducing Nonresponse Bias throughout the Survey Process. ASA Proceedings of the Joint Statistical Meetings.

[43] Nicholas Clifford, Shaun French, Gill Valentine (2010). Semi-structured Interviews and Focus Groups. Key Methods in Geography page 103-113, ISBN9780857025364

[44] Dave Gruber (2014). Five ways to bring a more social, open development environment to your company.

[45] Taghi Javdani Gandomani, Hazura Zulzalil, Abul Azim Abd Ghani and Abu Bakar MD Sultan (2013). A Systematic Literature Review on relationship between agile methods and Open Source Software Development methodology, arXiv:1302.2748 [cs.SE].

[46] Liz Barnett (2004). Applying Open Source Processes In Corporate Development Organizations. Forrester Research.

[47] About Inner Source. Access: https://innersource.accenture.com/about

[48] Andrew Stellman (2009). Requirements 101: User Stories vs. Use Cases.

[49] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell Anders Wesslén (2000). Experimentation in Software Engineering, pp. 66 - 73, ISBN:0-7923-8682-5.

[50] Inner Source Strategy Adopting Open Source Methods in the Enterprise. https://mc-3004-1805356660.us-east-1.elb.amazonaws.com/consulting/inner-source

A
Appendix

A.1 Get Config

A.1.1 Request

```
GET /projects/myproject/config
```

A.1.2 Response

```
HTTP/1.1 200 OK
Content-Disposition: attachment
Content-Type: application/json; charset=UTF-8
)]}'

{
"description": "demo project",
"use_contributor_agreements": {
"value": true,
"configured_value": "TRUE",
"inherited_value": false
},
"use_content_merge": {
"value": true,
"configured_value": "INHERIT",
"inherited_value": true
},
"use_signed_off_by": {
"value": false,
"configured_value": "INHERIT",
"inherited_value": false
},
"create_new_change_for_all_not_in_target": {
"value": false,
"configured_value": "INHERIT",
"inherited_value": false
},
"require_change_id": {
"value": false,
"configured_value": "FALSE",
"inherited_value": true
},
"max_object_size_limit": {
"value": "15m",
"configured_value": "15m",
"inherited_value": "20m"
},
"submit_type": "MERGE_IF_NECESSARY",
"state": "ACTIVE",
"commentlinks": {},
"plugin_config": {
"helloworld": {
"language": {
"display_name": "Preferred Language",
"type": "STRING",
"value": "en"
}
}
},
"actions": {
"cookbook hello-project": {
"method": "POST",
"label": "Say hello",
"title": "Say hello in different languages",
"enabled": true
}
}
}
```

A.2 Set Config

A.2.1 Request

```
PUT HTTP/1.0
Content-Type: application; charset=UTF-8

{
"description": "demo project",
"use_contributor_agreements": "FALSE",
"use_content_merge": "INHERIT",
"use_signed_off_by": "INHERIT",
"create_new_change_for_all_not_in_target": "INHERIT",
"enable_signed_push": "INHERIT",
"require_signed_push": "INHERIT",
"reject_implicit_merges": "INHERIT",
"require_change_id": "TRUE",
"max_object_size_limit": "10m",
"submit_type": "REBASE_IF_NECESSARY",
"state": "ACTIVE"
}
```

A.2.2 Response

```
HTTP/1.1 200 OK
Content-Disposition: attachment
Content-Type: application; charset=UTF-8

)]}’
{
"use_contributor_agreements": {
"value": false,
"configured_value": "FALSE",
"inherited_value": false
},
"use_content_merge": {
"value": true,
"configured_value": "INHERIT",
"inherited_value": true
},
"use_signed_off_by": {
"value": false,
"configured_value": "INHERIT",
"inherited_value": false
},
"create_new_change_for_all_not_in_target": {
```

```
"value": true,
"configured_value": "INHERIT",
"inherited_value": false
},
"require_change_id": {
"value": true,
"configured_value": "TRUE",
"inherited_value": true
},
"enable_signed_push": {
"value": true,
"configured_value": "INHERIT",
"inherited_value": false
},
"require_signed_push": {
"value": false,
"configured_value": "INHERIT",
"inherited_value": false
},
"reject_implicit_merges": {
"value": false,
"configured_value": "INHERIT",
"inherited_value": false
},
"max_object_size_limit": {
"value": "10m",
"configured_value": "10m",
"inherited_value": "20m"
},
"submit_type": "REBASE_IF_NECESSARY",
"state": "ACTIVE",
"commentlinks": {}
}
```