Selection and Evaluation of Open Source Components

Marina Marinela Gerea
Supervisor: Carl-Fredrik Sørensen
Co-advisor: Reidar Conradi

Department of Computer and Information Science
NTNU Norwegian University of Science and Technology
Abstract

Open-source-software (OSS) becomes more and more used in the software products developed by the industry. The motivation of this can be to reduce the development/maintenance costs and to shorten time-to-market. There are several technical processes for selecting and evaluating OTS (Off-the-Shelf) components, but the research shows that these methods are used very little. OTS components come in two types: COTS (Commercial-Off-The-Shelf) and OSS (Open-Source-Software); this report focuses on OSS. The amount of systems with open source components is large and there are many questions without answers such as: Where and how to find such components? How to evaluate them? How to learn the components? How to take care of the knowledge about the chosen component? When should one use and how to make best use of OSS? What component’s version should be chosen and what could be done with respect to the maintenance and the new versions of the component? Without an effective way to select and evaluate the OSS components, the time spent choosing the correct components may offset the advantages of using them.

The report presents the current state-of-the-art, a research design, and then presents the results of an explorative survey performed in collaboration with a few participants from the Norwegian COSI (Co-development using inner & Open source in Software Intensive products) project and other Norwegian software companies. The state-of-the-art focuses on investigating the existing research related to software development processes for component based-systems and OTS selection processes with the goal of finding questions for research interviews. The results of the interviews may help software companies to understand the existing practices and improve their own processes.

This research has two explorative research goals and five research questions. The research goals are: 1) to explore the following company role: user of open source components and 2) to explore their development processes with focus on selection and evaluation processes.

The contributions of this work include a literature study and new knowledge. The new knowledge refers to the qualitative and quantitative answers to the research questions.

**Keywords:** Open Source, Open Source Selection and Evaluation, Structured Interviews, Survey.
Preface

This report has been written during Autumn 2006, as part of the course TDT4735 Software engineering, depth study at the Norwegian University of Science and Technology (NTNU). The main goal of the course is to give an advanced introduction into software engineering through a selected project within the domain. The context of the work is within the European ITEA project, Norwegian COSI (Co-development using inner & Open source in Software Intensive products). The Norwegian COSI wants to enable the Norwegian IT sector to fully exploit the benefits and advantages of open source software (OSS).

This work consists of a literature study, and a survey about development processes with OSS components. In particular, it’s interesting to discover the actual process used in industry when it comes to selection and evaluation of OSS components.

The work was performed under the supervision of Dr. Carl-Fredrik Sørensen and Professor Reidar Conradi. I want to thank them for providing insightful input and valuable feedback. The group of people working on the COSI project at IDI, NTNU was very enthusiastic and eager to provide feedback any time I needed. I was glad to be part of the group, even for a short period of time, and I am grateful to them. I would like to thank to Øyvind Hauge and Claudia Ayala who helped me and encouraged all the time.

Together with Erik Løvland, who was also taking the course TDT4735 Software engineering, depth study at NTNU, we have performed a common interview in the autumn 2006. We would like to thank all the respondents who took time to answer the interview: Audun Jensvoll from Keymind Computing, Per Meland from SINTEF ICT, Skjalig Stilson Arstad from FAST ASA, Harald Wesenberg from Statoil ASA, Vidar Langseid from eZ Systems, Jørgen Austvik from SUN, and Thomas Malt from Linpro.

Trondheim, December 15, 2006

__________________________________________
Marinela Gerea
Contents

Part 1 - Introduction ......................................................................................................................................... 1

1. Introduction ............................................................................................................................................... 1
   1.1 Problem Outline ................................................................................................................................. 1
   1.2 The Purpose of this Thesis ................................................................................................................. 2
   1.3 Contributions .................................................................................................................................... 3
   1.4 Report Structure ............................................................................................................................... 3

Part 2 – Pre-study .......................................................................................................................................... 5

2. An introduction to open source ................................................................................................................ 5
   2.1 Definition of Open Source .............................................................................................................. 5
   2.2 Definition of COTS ......................................................................................................................... 5
   2.3 Organizational Issues ...................................................................................................................... 6
      2.3.1 The Roles in OSS Community and Open Source Projects .................................................. 6
   2.4 Summary ......................................................................................................................................... 7


4. OTS Selection Processes ........................................................................................................................... 11
   4.1 Phases of the OTS Selection and Evaluation Process .................................................................... 11
      4.1.1 Phases of COTS Selection and Evaluation Process .............................................................. 11
      4.1.2 Phases of OSS Selection and Evaluation Process ............................................................... 12
      4.1.3 Differences between COTS and OSS .................................................................................. 13
   4.2 General Considerations and Criteria in an OTS Selection Process ............................................. 13
   4.3 Open Source Licenses .................................................................................................................... 16
   4.4 COTS Selection Methods .............................................................................................................. 16
   4.5 Summary ....................................................................................................................................... 22

Part 3 – Research ....................................................................................................................................... 23

5. Research Agenda .................................................................................................................................... 23
   5.1 Empirical Strategies ........................................................................................................................ 23
      5.1.1 Choice of Research Methods ................................................................................................. 24
   5.2 Research Questions ........................................................................................................................ 24
   5.3 Interview Definition ....................................................................................................................... 25
      5.3.1 Object of Study ......................................................................................................................... 25
      5.3.2 Purpose ................................................................................................................................... 25
      5.3.3 Quality Focus ............................................................................................................................ 25
      5.3.4 Perspective ............................................................................................................................... 26
      5.3.5 Research Context ..................................................................................................................... 26
   5.4 Interview Planning .......................................................................................................................... 28
      5.4.1 Context Selection ..................................................................................................................... 28
      5.4.2 Respondents ............................................................................................................................ 29
      5.4.3 Interview Design ..................................................................................................................... 29
List of tables

Table 1: The content of this report ................................................................. 3
Table 2: Summary of methodologies dealing with COTS selection .................. 21
Table 3: Activities and Roles in OTS Components Selection .......................... 25
Table 4: The corresponding questions in the interview guide for each research question 31
Table 5: Our hypotheses .............................................................................. 40
Table 6: RQ2, H1 and H2 .............................................................................. 41
Table 7: RQ3, H3, H4, H5, H6 and H7 ........................................................... 41
Table 8: RQ4, H8 and H9 .............................................................................. 42
Table 9: RQ5, H10, H11 and H12 ................................................................ 42

List of figures

Figure 1: OTS, OSS and COTS components .................................................... 5
Figure 2: The roles in open source projects ...................................................... 7
Figure 3: Phases of OTS selection process ....................................................... 22
Part 1 - Introduction

1. Introduction

1.1 Problem Outline

More and more companies use OTS (Off-the-Shelf) components in their daily software development. The OTS components can be distinguished in two types: COTS (Commercial-Off-the-Shelf) and OSS (Open-Source-Software). As described in [H.K.N.Leung 2003], “use of commercial-off-the-shelf (COTS) products is becoming an acceptable software development method. With the increased number of available COTS components, the time spent on choosing the appropriate COTS products could easily offset the advantages of using them”. Therefore, it is very important to effectively select and evaluate COTS components.

NTNU is participant in the Norwegian COSI, providing experience and knowledge, primarily related to empirical research in software engineering.

Our contributions to the Norwegian COSI project are: a literature review, a research design, a survey, all these with the goal of discovering better methods to select and evaluate OSS components that can help companies in their development processes. It is necessary to empirically investigate how software companies select and evaluate components, in different projects and different application domains.

The PhD thesis of Jingyue Li from September 2006 shows that:

• Open source software is often used as closed source
• Formal selection processes were rarely used
• Functional completeness was regarded to be more important than architectural compliance in COTS component selection
• The actual OTS-based development process was the traditional process with OTS-specific activities. The development process was dominated by existing company/department rule instead of the decision of using OTS components.
• The actual OTS component selection can be done in different phases. The main evaluation processes are familiarity-based or hands-on-trial-based. The phase to select OTS component depends on the project members’ familiarity to components.

The master thesis at NTNU of Øyvind Hauge and Andreas Røsdal, “A Survey of Industrial Involvement in Open Source” shows that:

• Open source components are primarily regarded as commodity software
• High availability of components, source code, and information related to the components are the most important motivations for using OSS components
• Selection and evaluation of OSS components are primarily done as a process based on previous knowledge, informal searches, and subjective evaluations of the components.
An exploratory study based on structured interviews performed in India and US between June and September 2003 [Madanmohan and De2004], shows that:

- The teams had no need to look at or modify the source and did not have enough resources (knowledge, skills, or manpower) to do so.
- If the open source component offers the best solution and reliability for the price, then it is the most appropriate.
- From the perspective of a commercial firm, the license must explicitly permit the distribution of software built from modified source code.

They therefore proposed that further investigation is needed regarding the OTS development process and the OTS component selection. This depth project can also be seen as a continuation of their research.

1.2 The Purpose of this Thesis

The purpose of the research performed in this project, is described in terms of the project description, the overall design goal and several research questions.

The project description contains two clear tasks: to do a literature study, and to prepare a survey. Since industrial partners are involved in the COSI project, the literature study focuses not only on purely academic papers, but also on papers containing aspects of the practices in the software industry. A special focus was on papers describing the selection and evaluation processes of OSS and COTS. We investigated both because they have things in common and it is interesting to see the difference.

As survey type, we have performed an explorative study. It is used as a pre-study to a more thoroughly investigation that is planned to be performed in the spring semester 2007.

Research goal:
Explore the company role: user of open source components.
Explore their development processes with focus on selection and evaluation processes.

Research questions:
RQ1: Who initiates and performs the work related to OSS harvesting and when in the development process?
   RQ1.1: Who initiates the OSS harvesting in a company?
   RQ1.2: Who does the work related to the harvesting?
   RQ1.3: Who takes the final decision about integrating an OTS component?
   RQ1.4: When in the development process does one select and evaluate OTS-components?
RQ2: What are the most important issues when searching for OSS components: business, technical, organizational or personal issues?
RQ3: What is the current process of selecting and evaluating OSS components?
RQ4: What versions are considered and how to deal with new versions?
RQ5: How to maintain the knowledge about the processes of selection and evaluation of OSS components?
1.3 Contributions

The contributions of this work can be divided into two parts: the literature study and new knowledge regarding the industrial practice about selection and evaluation of OSS components.

The first contribution is the literature study. This study provides descriptive information about the state of the art.

The second and most important contribution is new knowledge. This knowledge refers to the qualitative and quantitative answers to the research questions. We have proposed 12 hypotheses based on our results. The results and the hypotheses are presented in chapter 6.

Our work creates a platform for future work mainly for us, but also for others who have similar interests.

We have developed an interview guide and performed several structured interviews to assure/improve it. The improved interview guide gives a good starting point for a more through study.

1.4 Report Structure

This report is structured into several chapters.

Chapter 1, Introduction, is this part of the document.

Chapter 2, 3 and 4 presents the Pre-study. They contain the results from the literature review and give an overview over the existing selection and evaluation methods described in literature for Off-the-shelf (OTS) components.

Chapter 5 presents the Research design. It presents the project timeline, the research questions, the empirical strategies used, the interview design, and the research context.

Chapter 6, 7 and 8 present the Results, Discussion, and Conclusions of the study.

After the references, the last part contains several appendices like: Glossary, The Open Source Definition, and the Interview Guide.

An overview of the content of this report can be found in Table 1.

<table>
<thead>
<tr>
<th>Part</th>
<th>Chapter</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>Pre-study</td>
<td>2</td>
<td>An introduction to open source</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Software development processes for CBS</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>OTS Selection processes</td>
</tr>
<tr>
<td>Research</td>
<td>5</td>
<td>Research agenda</td>
</tr>
<tr>
<td>Results, Discussion and</td>
<td>6</td>
<td>Results</td>
</tr>
<tr>
<td>Conclusions</td>
<td>7</td>
<td>Discussion</td>
</tr>
<tr>
<td>References</td>
<td>-</td>
<td>References</td>
</tr>
<tr>
<td>Appendices</td>
<td>A</td>
<td>Glossary</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>The Open Source Definition</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Interview Guide</td>
</tr>
</tbody>
</table>

Table 1 : The content of this report
Part 2 – Pre-study

2. An introduction to open source

This chapter aims to give the reader a short introduction to open source by presenting the definition of open source and COTS, followed by a short description of organizational issues and software development processes.

As indicated already, there are two types of OTS components: OSS and COTS (see Figure 1).

![OTS, OSS and COTS components](image)

Figure 1: OTS, OSS and COTS components

2.1 Definition of Open Source

Bruce Perens [opensource2006] wrote a definition of open source as a pragmatic reaction to the misinterpretation of free software as “gratis software”. The Open Source Definition is currently available in version 1.9, see appendix B.

2.2 Definition of COTS

Because the terms OSS, COTS, and OTS are used everywhere in the report, we think it is important to provide clear definitions of them for the reader. [Carney and Long2000] consider that the existing labels COTS and OTS are not sufficient, and many people do not understand exactly the meaning of COTS. If a component was obtained rather than made, then it is called Off-the-Shelf (OTS). A Commercial-off-the-shelf (COTS) component can be modifiable or not. The solution is to separate attributes along two axes: source and modification, and locate the component in a graphical space.

Placing a component on the source axis, the component can be:

- Independent commercial item: This is a COTS
- Custom version of a commercial item: Has some degree of COTS;
- Component produced under a specific contract: Has some degree of COTS
- Existing component obtained from external sources (for example, a reuse repository): This is a previously developed component.
- Component produced in-house: This is a totally non OTS component.

Placing a component on the modification axis, the component can be:

- Very little or no modification.
- Simple parameterization.
- Necessary tailoring or customization.
• Internal revision to accommodate special platforms requirements:
  Include code revision to use special operating system protocols or real-time
  constraints.
• Extensive functional recording and reworking:
  A drastic modification, fundamental alterations such as removing safeguards or
  internal access control.

The open source definition is presented in appendix B. OSS components have always
the source code opened but COTS components may have the source code opened or not.
Therefore it is very important to distinguish between a COTS components and OSS
component. Placing a component on the source axis, shows the distinction between a
COTS and a non COTS component (which can be OSS component or a component
produced in house).

2.3 Organizational Issues

2.3.1 The Roles in OSS Community and Open Source Projects

People can take different roles in open source projects like: joiner, newcomer and
developer [Krogh et al.2003]. A joiner is someone who is on the e-mail list but does not
have access to the CVS repository. A newcomer is someone who has just begun to make
changes to the CVS repository. A developer is someone who has moved beyond
newcomer stage and is contributing code to the project.

Usually, a significant period of observation is necessary for a joiner to contribute to the
technical discussion. A joiner lurk silently on the developer list ad learns as much as
possible in order to make a technical contribution, rather than entering into the
developing list asking general questions.

There are different roles in an open source project, but people can often have several
roles, e.g. both developer and user [Berquist and Ljungberg2001].

These roles as shown in Figure 2 are:
• The owner of an OSS: is the person (or the group) who started the project and has
  exclusive right.
• Core developers: it is a group of core developers who write most of the source
  code, design, and take the important decisions about the project
• Developers: a wider group than the core developers who repairs defects.
• Problem reporters: this is even a larger group
• System testers
• User support: this task is performed mainly by some product users voluntarily
  providing answers to the questions of other users
• Users: it is a huge number of users, large organizations have become users. By
  using OSS, these actors are important contributors. Commercial businesses tied to
  open source projects provide additional resources for development and help to
  promote the open source packages and drive them into the mainstream.
2.4 Summary
The part of the chapter refers to the definition of OSS in appendix B. Many times, people are confused about the difference between OSS, COTS or in-house components, mainly because they all may have the source code opened. Therefore, a definition of COTS component and how is different from OSS and in-house component is necessary to be presented.

Regarding the organizational issues paragraph, it is important to see the different roles in open source projects. Our research goal is to explore the company role: user of open source components. This company role belongs to the “users” role in open source projects.

Before reviewing CBS development processes, it is important to list the traditional processes.

The traditional process models are:

- **Code and Fix**: Developers begin work without a set of good requirements or a vision, writing code as long as there is time and money. This approach has no planning and therefore yields early results. High maintenance and rework effort are the result of this model. The code developed with this method has usually poor quality.

- **Waterfall**: This model is a sequential, document-driven methodology. To advance from the current phase to the next phase, the project team must review and release an artifact in the form of a document. [Royce87]

- **V-model**: This model is an extended waterfall model, adding more details on the validation and verification side. For each phase in development, there is an associated phase to verify and validate the result produced. [German V-model]

- **Spiral**: The spiral model breaks the project down into risk-oriented subprojects [Boehm88]. The advantage of this model is the breakdown of development into small pieces, so that the areas of risk can be tackled at the beginning of the project.

- **Evolutionary Prototyping**: In this model, the system starts with an initial idea or proposal, which is then prototyped and released to the customer in incremental releases based on feedback.

- **Adaptive Software Development**: The adaptive software development model is an iterative, risk- and mission-driven, component-based, time-boxes, and change-tolerant process [Highsmith99]. It works well for small teams (four to eight people) when requirements are uncertain or domain knowledge is missing.

The composed process models, which cover several aggregations, are:

- **Unified Process**: The UP is a risk- and use-driven, architecture-centric, iterative and incremental software development model.

- **Agile models**:
  - **Extreme Programming**: The eXtreme Programming (XP) model [Kent99] is a flexible, lightweight, people- and result-oriented development process that allows for requirements changes at any time during development. XP defines 12 core practices that are essential to the success of an XP project.
  - **Scrum**: The characteristics of this process model are: tries to have everybody involved, the requirements from customers are priorities, daily meetings with the customers.

Nowadays there is a significant amount of work that identifies issues or proposes frameworks for pursuing further research for improving new software development processes for component-based system (CBS).

For instance in [Brownsword et al.2000] the changes required to address the CBS development are identified taking into account real-life lessons. Then they articulate a framework for organizing the new and changed process elements.

[Morisio et al.2000] reports about adopted COTS-based processes and proposes a new one.
The process model in Open source software development (OSSD) is different from Closed Source Software Development (CSSD) [Vidyasagar and Chang2004]. CSSD normally follows a spiral or iterative model of development, whereas OSSD follows an evolutionary model for development where the software never reaches a final state and keeps on evolving. OSSD is more a concurrent or parallel process.

In most cases, the process that builds applications on existing OSS is largely similarly with the process of developing COTS-Based Applications (CBA). Selecting correct OSS, adjusting them as needed, and integrating them are essential actions during system development lifecycle [Huang and Yang2005]. The authors describe a process for building OSS Based Applications (OBAs), it’s a process with several activities:

- Identifying High-level Requirements & Candidate OSS;
- Designing high-level Architecture;
- Eliciting Requirements and Establishing Assessment Criteria;
- Assessing OSS according to criteria;
- Adjusting Criteria or High-level Architecture;
- Constructing In-house Products;
- Improving and Integrating OSS;
- Productizing and Transiting applications.

If suitable OSS does not exist, it is necessary to construct in-house products. In the Eliciting Requirements and Establishing Assessment Criteria activity, attributes of OSS are used to build assessment criteria: correctness of functionality, flexibility of interface, availability/robustness, installation/upgrade ease, security, portability, product performance, user interface, etc.

Despite the many differences from proprietary software, it is important to note that OSS usually goes through the same stages as a proprietary product [Murrain2004]. Some key differences between proprietary software and OSS are:

- In open source project development, this process may happen much more organically - starting from a single developer to more developers who gradually become interested in the project
- The pace of open source development can be slower, due to the voluntary nature
- The quality of OSS can be much better than proprietary software (because of many adopters and testers)
- The version numbering of open source software tends to be more conservative.
4. OTS Selection Processes

In a COTS-based process, as well as in an OSS-based process, an effective and efficient COTS product selection process is essential to the delivery of the full potential of CBS development. [H.K.N.Leung2003].

Most studies focus on proposing selection processes and methods to select COTS components. Since both the COTS and OSS components are acquired from third-party companies, the users usually have no control of the provided functionality and evolution of these components [Li2006]. They believe that most proposed process improvement in COTS components-based development can also be used in OSS component-based development.

Everywhere in the report we will refer to the term “selection and evaluation process” because we observed from the pre-study that often few components are selected, which then are thoroughly evaluated. The output of the evaluation is often one selected component which will be integrated in a system or application.

4.1 Phases of the OTS Selection and Evaluation Process

We start the sub-chapter by presenting the phases of COTS selection and evaluation process and the phases of OSS selection and evaluation process.

Even if this depth project is about open source components, we presented both the phases of selection and evaluation process of COTS and OSS because we believe that they have things in common.

4.1.1 Phases of COTS Selection and Evaluation Process

According to [Kunda-Brooks1999], there are three phases of COTS software selection:

**Evaluation criteria definition:** The criteria definition process essentially decomposes the requirements for COTS into a hierarchical criterion set. The criteria include component functionality (what services are provided), other aspects of a component’s interface (such as the use of standards) and quality aspects that are more difficult to isolate, such as components reliability, predictability, and usability. In [Boehm00], four guiding principles tailor a requirements strategy. These ensures that the requirements are value-, shared-vision-, change-, and risk-driven. The value-requirements will show how the system will add value for each stakeholder. The shared-vision helps all the stakeholders quickly adapt to the new situation, while the precise requirements specifications take more effort to change. The change-driven requirements are important because the requirements change in time; also it is important to capture the rationale besides decisions, not only the final decisions. Still it is need for a detailed requirements specification to see possible risks (risk-driven requirements). The author also points out that developers need some guidelines principles rather than inflexible rules to help each project determine the best combination of requirements and flexibility for each technical and organizational situation.

**Identification of candidate components:** The identification of candidate components, also known as alternatives identification, involves the search and screening for COTS candidates that should be included for assessment in the evaluation phase. In general, evaluating and analyzing all the relevant characteristics of COTS candidates take a great
amount of time, typically more than the organization has. Therefore, it is both necessary and cost-effective to select the most promising candidates for detailed evaluation.

**Evaluation of the criteria for these candidates:** There are currently three strategies to COTS evaluation: progressive filtering, keystone identification and puzzle assembly. In the keystone strategy, products are evaluated against a key characteristic such as a vendor or type of technology. Progressive filtering is a strategy whereby a COTS product is selected from a larger set of potential candidates, in which products that do not satisfy the evaluation criteria are progressively eliminated from the products list. In the puzzle assembly model, a valid COTS solution will require fitting the various components of the system together.

### 4.1.2. Phases of OSS Selection and Evaluation Process

In [Wheeler2006], it is described a general process for evaluation of open source software/Free software (OSS/FS) programs. This process is based on four steps: identify candidates, read existing reviews, compare the leading programs’ basic attributes to your needs, and analyze the top candidates in more depth.

**Identify candidates**

The process proposes to first look at lists of OSS programs, lists of “generally recognized as mature” (GRAM) or “generally recognized as safe” (GRAS) OSS programs. Such a link is: [Wheeler2004], where the author made a GRAM list, structured on operating systems, network services, databases, desktop applications, development tool/languages, web applications, GUI infrastructure, security and research tools. Then, he recommends searching on specialized sites as:

- [http://freshmeat.net/](http://freshmeat.net/)
- [http://directory.fsf.org/](http://directory.fsf.org/)
- [http://www.theopencd.org/](http://www.theopencd.org/)
- [http://www.koders.com/](http://www.koders.com/)

Another way is to search using search engines, especially Google’s specialized searches for Linux and BSD. He also recommends some tips about searching for OSS/FS programs and components.

**Read existing reviews**

It is much more efficient first to learn about a program’s strengths and weaknesses from a few reviews than to try to discern that information just from project websites. The simplest way is to use a search engine and search for an article containing the names of all the candidates not identified. Also it is indicated to search for web sites that cover that market or functional area and see if there are published reviews. Three OSS/FS SCM systems got the most discussion in April 2004: CVS, Subversion, and GNU Arch. Market share is an important indirect “review” of a product.

**Compare the leading programs’ basic attributes to your needs**

First, one should read the project web site to get information about the project. Next, one can evaluate the project on a number of important attributes: functionality, cost, market
share, support, maintenance, reliability, performance, scalability, usability, security, flexibility/customizability, interoperability, and legal/license issues.

**Analyze the top candidates in more depth**

This step is, for the most part, done the same way for both proprietary and OSS/FS programs. The same attributes as in the previous step are investigated, but people need to spend more effort by actually trying things out instead of quick reading.

[Madanmohan and De2004] developed an open source component selection model, which involves three basic steps: *collection, incubation* and *revision*.

In the *collection* step, developers search for open source components and explore the environment for new ideas and components. This should be according to their firm’s product roadmap.

In the *incubation* step, developers create and evaluate information about the software component. They question the collected information and evaluate component usability through brainstorming and what-if analysis.

In the *revision* step, developers critically revise existing components through social selection. This revision involves revisiting the existing component library, estimating the value of those components, and deciding whether to add or to extend them in order to meet the customer requirements.

### 4.1.3 Differences between COTS and OSS

The way the steps are performed in an evaluation is different between OSS/FS programs and proprietary programs [Wheeler2006]. Some key differences may be:

- The information available for OSS is usually different than for proprietary programs
- OSS can be changed and redistributed by customers. This difference affects many factors, such as flexibility/customizability, support options.
- No hidden defects or features in OSS
- All the features are available in OSS

When considering open-source based platforms, there are at least two crucial differences when compared to more traditionally proprietary platforms [Derick and West2004]:

- The open source solution uses collaborative R&D and support in cooperation with firms whose role is far less central or defined. The R&D, sales and support for the proprietary solution are the responsibilities of a well-defined profit-making enterprise.
- In OSS, the source code is widely disseminated to all and thus organizations have the opportunity to modify the software to suit their own needs.

### 4.2 General Considerations and Criteria in an OTS Selection Process

In [Alves and Castro2001] four main dimensions should be considered in a COTS selection process: domain coverage, time restriction, cost rating, and vendor guaranties. Some can apply to OSS as well.
• **Domain Coverage**: The components have to provide all or part of the required capabilities, which are necessary to meet core essential customer’s requirements. The domain coverage can also apply to OSS, since both COTS and OSS must anyway fulfill the customer’s requirements.

• **Time restriction**: Software companies usually operate in a very rigid development schedule, on which their competitiveness depends. Selection is a time consuming activity, where a considerable amount of time is necessary to search and screen all the potential COTS candidates. This can also apply to OSS when companies select components. Either it is COTS or an OSS, software companies have always time restriction.

• **Cost rating**: The available budget is a very important variable. The expenses when selecting COTS products will be influenced by factors such as: license acquisition, cost of support, adaptation expenses, and maintenance prices.

• **Vendor guaranties**: An important aspect to be considered in the selection activity is to verify the technical support provided by the vendor. Some issues have to be taken into account, for example: vendor reputation and maturity, number and kind of applications that already use the COTS, and characteristics of the maintenance licenses.

In [Ruffin and Ebert2004], some criteria that OSS must meet, are described:

- Build on and follow a mature and commonly used industry standard such as de facto standard like Linux.
- Have a strong OSS community.
- Be broadly supported by several Independent Software Vendors (ISVs) for distribution, evolution and support.
- Have a clear, indisputable legal status regarding Intellectual Property Rights (IPR) and the right to use it.

They also confirm that: “Alcatel maintains a database, which all its engineers can access, containing experiences with ‘open source-like’ that someone in the company has studied (but not necessary used).” It provides information such as how to distribute the software, whether someone in the company has used it, who the favorite distributors are and how they behaved, how to contribute to OSS initiatives, what liability aspects to anticipate, and what end-of-life conditions to consider.

In [Woods and Guliani2005], the elements of open source maturity (elements that are direct indicators of the potential difficulties one can encounter when using open source) are discussed:

- **Leadership and culture**: It’s very important for an open source project to have good leaders and highly knowledgeable individuals.
- **Vitality of community**: 
The leadership and the vitality of the community are correlated. In a good community, everyone finds something useful to do, and there is a division between the project’s developers and users. The size of the community is also a good indicator of a project’s viability.

- **Quality of end-user support:**
  Active forums, well-maintained FAQs, and documentation that is available through search engine can save a lot of time.

- **Extent and scope of documentation:**
  The quality of documentation is another good clue about a project’s work process. It is normal to expect that the instructions for installing, running, and fine-tuning a piece of software to be written in clear English.

- **Quality of packaging:**
  It is a good indication of maturity for a project to have an installation package that can install the software easily on many different platforms and configurations.

- **Momentum:**
  Often releases indicate that the project is active, the users do not have to wait too long for bug fixes. It is important to check the release history to see if the new releases are mainly significant or more trivial.

- **Quality of code and design**
- **Quality of architecture**

- **Testing practices:**
  Some open source code come with automated, build-in testing facilities as standard features. The presence of unit tests is a key indicator of good design.

- **Integration with other products:**
  All software in the enterprise operate as part of an *ecology*, meaning that a set of interdependencies cause programs to call on each other.

- **Support for standards:**
  The programs need to use standards-based APIs. The problem of dependencies and nonstandard APIs is sometimes addressed in open source projects by having a program download with a pre-selected set of other programs (the programs that make up its local ecology).

- **Quality of project site**
- **License type:** the main license types will be explained on the next sub-chapter.

- **Potential for commercial conflicts**
- **Corporate commitment:**
  Several open source projects, such as Linux operating system and Apache Web Server, have enjoyed support from large, established computer companies, including IBM, Sun, HP, and Dell.

Some maturity models exist to help organizations to successfully implement open source software. An example is the Open Source Maturity Model (OSMM) from Navica [Navica]. OSMM assesses the maturity level of all key product elements:

- **Software**
- **Support**
- **Documentation**
From these different sources we observe that there are three different segments to be considered when selecting open source components:

1. Business aspects
2. Technical aspects
3. Organizational aspects
4. Professional aspects

4.3 Open Source Licenses
Opensource.org ([www.opensource.org](http://www.opensource.org)) lists 21 approved licenses, which are the standards licenses that the Open Source Initiative (OSI) has certified as valid. The base license determines how a developer can use an open source software component. From commercial firm’s perspective, the license must explicitly permit the distribution of software build from modified source code [Madanmohan and De2004].

The most used licenses are:

- **GNU General Public License (GNU GPL)**
  It is the earliest open source license, created in 1980s to distribute the GNU project software. Most open source software to this date has been distributed under GPL. It is a copy-left license. This license is controversial because it requires that all applications that contain GPL software are also released under a GPL license. A modified license, the Lesser GPL (LGPL) was created when this proved impractical. The LGPL differs from the GPL in two ways. Firstly, it is intended for use with software libraries. Secondly, the software may be linked with proprietary code, which is precluded by the GPL.

- **Berkley Software Distribution (BSD)**
  This is also an early license. The company is not required to grant any particular license right. Its main requirement is the retention and acknowledgment of previous contributors’ work. It is a free license. Many commercial companies use this license in their business and software they develop, e.g.: Yahoo, Apple, Microsoft. It is a copy-right license.

- **Mozilla Public License (MPL)**
  This license was created by Netscape when they launched the Mozilla project. The license is a mixture of BSD and GPL. Separate non MLP files may be licensed under other terms. Company must apply MPL terms to any file that contain MPL code.

4.4 COTS Selection Methods
According to [H.K.N.Leung2003], there are three categories of COTS product selection methods: the *intuition* approach, the *direct assessment* approach and the *indirect*
assessment approach. In the intuition approach, software developers select COTS products according to their experience and intuition, it’s a subjective approach. Most of the methods belong to the direct assessment (DA) approach, which selects COTS components directly from their source. The problem with these approaches is that they can be quite inefficient when there are many components to be investigated.

A range of direct assessment methods have been proposed for COTS product selection: [Alves-Castro2001], [Kontio1996], [Maiden and Ncube98], [Morisio et al.2000], [Chung and Kooper2004], [Tran et al.1997], [Fox et al.1997].

**OTSO**

This method is based on direct assessment. Kontio [Kontio96] proposed the Off-The-Shelf-Option (OTSO) selection method. The OTSO method assumes that the requirements of the proposed system already exist. But in practice it is possible that the requirements cannot be defined precisely because the use of some COTS products may require some changes to the requirements. The main principle followed by the OTSO method is that of providing explicit definitions of the tasks in the selection process, including entry and exit criteria. The inputs are: requirement specification, design specification, project plans and organizational characteristics. The outputs are the selected COTS product, the results of the evaluation, and cost models.

The OTSO selection method comprises three phases: searching, screening and evaluation. The searching phase attempts to identify all potential COTS candidates. In the screening phase, it is decided which COTS candidates should be investigated further. In the evaluation phase, COTS candidates undergo a detailed evaluation. Criteria for COTS selection are:

a. functional requirements of the COTS
b. required quality-related characteristics
c. business concerns
d. issues relevant to the software architecture

The OTSO method uses the Analytic Hierarchy Process (AHP) to consolidate the evaluation data for decision-making purposes [Saaty]. AHP is based on the idea of decomposing a complex, multi-criteria decision-making problem into a hierarchy of the selection criteria. It helps decision makers to structure the important components of a problem into a hierarchical structure. The method assumes that the requirements exist; they are used to define the evaluation criteria.

**CISD**

This method is based on direct assessment. Tran, Liu and Hummel have proposed the COTS-based Integrated System Development (CISD) model [Tran et al.1997].

The inputs for the selection process are the system requirements and information about the COTS products, and the outputs include a prioritized list of COTS products and the architecture of the system.

The CISD model consists of three phases: identification (with two sub-phases: product classification and product prioritization), evaluation and integration.

In the evaluation phase, three attributes of the COTS products are examined: functionality, architecture and performance. The integration phase encompasses all of the
development effort that is required to interconnect the different selected COTS products into a single integrated system.

**PORE**
This method is based on direct assessment.
The *Procurement-Oriented Requirements Engineering* (PORE) method [Maiden and Ncube1998], [Maiden et al.2002] integrates techniques from requirements engineering, knowledge engineering, multi-criteria decision-making and feature analysis to guide the selection of COTS packages. It proposes a concurrent development process, in which stakeholders requirements acquisition and COTS package selection occur at the same time.
The method uses a progressive filtering strategy, whereby COTS products initially selected are then progressively eliminated when they do not satisfy the evaluation criteria. The inputs of the method are the attributes of the COTS products, supplier requirements, and information about product branding, open standards, product certification, the development process, reliability, security and dependability. The output is a shortlist of COTS products. In this method it is not clear how requirements are used in the evaluation process and how products are eliminated.

**CRE**
This method is based on direct assessment.
The COTS-based Requirements Engineering (CRE) method [Alves-Castro2001] was developed to facilitate a systematic, repeatable and requirements-driven COTS product selection process. The method focuses on non-functional requirements to assist the process of selection and evaluation of COTS products.
The method has four iterative phases: identification, description, evaluation and acceptance. The identification phase is based on a careful analysis of influencing factors; there are five groups of factors: user requirements, application architecture, project objectives & restrictions, product availability and organizational infrastructure.
During the description phase, the evaluation criteria are elaborated in detail.
In the evaluation phase, a particular COTS product is selected based on estimated cost versus benefits.

The selection criteria for the CRE method are:

- **Domain coverage:**
  Both functional and non-functional requirements are important to meet customer needs. A special focus is on non-functional requirements.

- **Time restriction:**
  The time available for searching and screening all the potential COTS candidates is limited.

- **Cost rating:**
  Acquiring a license, support cost, expenses associated with adapting the product, maintenance cost should all be within the available budget.

- **Vendor guaranties:**
  The technical support provided by vendor is also important.
A disadvantage of the CRE method is that the decision-making process can be very complex.

**IIDA**

Another work presented in [Fox et al. 1997] describes the *Infrastructure Incremental Development Approach* (IIDA) for the development of technical infrastructure using COTS products. This approach is a combination of the classical waterfall and spiral development models in order to accommodate the needs of CBS development. The process of selecting COTS products in the IIDA relies on two phases: analysis prototype and design prototype. In the analysis-prototype phase, COTS candidates are selected from each COTS product family. In the design prototype phase, the best COTS products from the earlier phase are selected and evaluated. Basic evaluation criteria include functionality and performance.

**CARE**

This method is based on direct assessment. The *COTS-Aware Requirements Engineering* (CARE) process is described in [Chung and Cooper 2004]. The CARE approach is characterized as agent-oriented, goal-oriented, knowledge based, and has a defined methodology, or process. The goal is to define a methodology that supports the definition and selection of COTS components from a technical view. The early artifacts for the system under development are accounted for including:

- The agents
- Soft-goals: non-functional goals that are achieved not absolutely but in a “good-enough” sense
- Hard-goals: functional goals
- System requirements
- Software requirements
- Architectural elements

The traceability relationships among the artifacts are also established and maintained (e.g. a soft-goal is refined and traced to specific system requirements).

All these descriptions are information like that found on marketing brochures for existing products. These general descriptions are used to determine if the product appears to be potentially useful.

This methodology departs from the description of the COTS components stored and maintained in a knowledge base, or repository. As the goals are defined, analyzed, and negotiated, the requirements engineer (RE) searches the repository for COTS components that appear to be (possible) matches. As the system develops, goals are defined into requirements and the RE determines if the identified components still seem suitable or if other components seem to be a better fit.

In spite of this methodology pretends to improve reuse by means of a repository of COTS descriptions, it presents 3 obvious disadvantages: (1) it is not clear how to build this repository; (2) the maintainability of the repository is specially difficult taking into account the rapid evolve of the information of each product and the increasing amount of products in the market and this aspect not considered (3) the searching process is not very efficient having to look for components in a widespread range of descriptions.
The authors [Leung03] have developed an indirect method: the Domain-Based COTS-product Selection method, which makes use of the specific domain model to decide the suitability of the COTS product. This approach has the goal to reduce the amount of work required for the selection process. There are two basic strategies for the selection of a COTS product; depending on whether an application development needs the best available COTS product: best-fit strategy and first-fit strategy.
Table 2: Summary of methodologies dealing with COTS selection

<table>
<thead>
<tr>
<th>Methodologies of Process</th>
<th>Selection procedure</th>
<th>Selection criteria</th>
<th>Reuse of knowledge</th>
<th>Descriptive criteria definition</th>
<th>Facility of searching the set of COTS to be evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTSO</td>
<td>-Searching</td>
<td>-Functional</td>
<td>-</td>
<td>√</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-Screening</td>
<td>requirements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Evaluation</td>
<td>-Quality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>characteristics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Business concerns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Relevant software architecture</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CISD</td>
<td>-Product identification</td>
<td>-Functional</td>
<td>-</td>
<td>√</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Architecture/interoperability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Fulfill multiple requirements from the service domain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORE</td>
<td>-Acquires customer requirements</td>
<td>-Development process</td>
<td>-</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Supplier CMM level</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Product/supplier past record</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Reliability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Security</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Dependability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IIDA</td>
<td>-Analysis prototype</td>
<td>-Functional</td>
<td>√</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-Design prototype</td>
<td>requirements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Performance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARE</td>
<td>-Select candidate hard-goals</td>
<td>-The agents</td>
<td>√</td>
<td>√</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Soft-goals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Hard-goals:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-System requirements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Software</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>requirements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Architectural</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>elements</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(√) addresses the issue fully  (*) deals with the issue but not fully (-) means not deal with the issue
4.5 Summary

The first three parts of the chapter presented the phases of COTS selection and evaluation process and the phases of OSS selection and evaluation process. Even if this depth project is about open source components, we presented both the phases of selection and evaluation process of COTS and OSS. We believe that the phases of COTS selection and evaluation process can also apply to OSS in general, even if there are some differences, e.g. in OSS selection and evaluation process, components are not evaluated against a key characteristic such as a vendor. [Wheeler2006] and [Modanmohan and De2004] describe specific phases for OSS selection and evaluation process. Some phases are also common for COTS. For example, the incubation phase for OSS described in [Modanmohan and De2004] is similar with the “identification of candidate components” phase for COTS described in [Kunda-Brooks1999].

From all the sources described in 4.1, we can propose an OTS selection and evaluation process:

```
Criteria definition → Find candidates → Evaluate (test) → Select → Integrate → Maintain
```

Figure 3: Phases of OTS selection process

The general considerations and criteria in an OTS selection process are presented both for COTS and OSS. Again, it is interesting to understand the differences and the similarities between these two. The open source licenses part gives an overview over the most used licenses. The licenses are always considered when people select and evaluate open source components. Finally, the COTS selection methods and a summary of methodologies dealing with COTS selection close this chapter.
Part 3 - Research

5. Research Agenda

This chapter presents the main empirical strategies with focus on an explorative survey since this it was used in our project. Then, the research questions, interview definition, and the research design are presented, followed by the research context.

The main activities of the project are:

- Project start-up
- Literature survey
- Establish research questions
- Prepare interview guide
- Company visit and interviews
- Analyze and present data and results

5.1 Empirical Strategies

There are two types of research paradigms that have different approaches to empirical studies:

- Qualitative research – is concerned with studying objects in their natural setting. It is concerned with discovering causes noticed by the subjects in the study, and understanding their view of the problem at hand.
- Quantitative research – is mainly concerned with quantifying a relationship or to compare two or more groups. The aim is to identify a cause-effect relationship.

In the empirical software engineering field, there are three major types of investigations (strategies) [Robson93]:

- Survey:
  A survey is often an investigation performed in retrospective, when, for example, a tool or technique, has been in use for a while. The primary means of gathering qualitative or quantitative data are interviews or questionnaires. These are done through taking a sample which is representative from the population to be studied. The results from the survey are then analyzed to derive descriptive or explanatory conclusions. They are then generalized to the population from which the sample was taken.

- Case study:
  Case studies are used for monitoring projects, activities or assignments.

- Experimentation:
  Experiments are done normally in laboratory environment, which provides a high level of control. The objective is to manipulate one or more variables and control all other variables at fixes levels.

Survey and Case study can be both qualitative and quantitative, while the experiment is only quantitative.

The general objectives for conducting a survey is either of the following [Babbie90]:

---

23
Explorative surveys are used as a pre-study through investigations to assure that important issues are not overseen. Creating a loosely structured questionnaire and letting a sample from the population answer it could do this. The information is gathered and analyzed, and the results are used to improve the full investigation. In other words, the explorative survey does not answer the basic research question, but it may provide new possibilities that could be analyzed and should therefore be followed up in a more focused or thorough survey.

5.1.1 Choice of Research Methods

Based on the preceding research methods, explorative research goals, and questions, it was decided to perform a structured interview as part of the depth project, followed by a questionnaire as part of the master thesis in the spring of 2007.

The interview was chosen as a qualitative method because it was the best trade-off between usage of time, and its ability to provide detailed textual description for the research questions. The interview was structured as much as possible in order to get comparable results from the respondents. The interview has open-ended questions because of the explorative research question. Some Norwegian software companies agreed to participate to the interview.

An interview can provide sufficiently detailed information in a relatively short period of time.

5.2 Research Questions

The goal of this project is to investigate state-of-the-practice of selection and evaluation of OSS components from both the organizational and process point of view. The motivation is to give guidelines on how to select and evaluate an OSS component, based on experiences and lessons learned from finished projects.

A pre-study was performed with the goal to find out what are the actual processes and selection and evaluation processes used in OSS component-based development projects.

Research goal:
Explore the following company role: user of open source components. [Hauge and Røsdal2006]
Explore their development processes with focus on selection and evaluation processes.

Company role description:
User of Open Source Component is typically a company which develops software using open source tools or components. Some high-level activities directly related with the OTS selection processes and the most relevant required roles are described in Table 3 [Ayala et al.2006].
Table 3: Activities and Roles in OTS Components Selection

<table>
<thead>
<tr>
<th>Activity</th>
<th>OTS Users Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding Candidate OTS</td>
<td>Market Watcher (MW)</td>
</tr>
<tr>
<td>Evaluating OTS Candidates</td>
<td>Quality Engineer (GW)</td>
</tr>
<tr>
<td>Selecting OTS Component</td>
<td>Selector (S)</td>
</tr>
<tr>
<td>Documenting the decision</td>
<td>Knowledge Keeper (KK)</td>
</tr>
</tbody>
</table>

We explore all these OTS company roles in our study. In fact, our first research question tries to relate these roles to the company size.

**Research questions:**

**RQ1:** Who initiates and performs the work related to OSS harvesting and when in the development process?
- RQ1.1: Who initiates the OSS harvesting in a company?
- RQ1.2: Who does the work related to harvesting?
- RQ1.3: Who takes the final decision about integrating an OTS component?
- RQ1.4: When in the development process does one select and evaluate OTS-components?

**RQ2:** What are the most important issues when searching for OSS components: business, technical, organizational, personal?

**RQ3:** What is the process of selecting and evaluating OSS components?

**RQ4:** What versions are considered and how to deal with new versions?

**RQ5:** How to maintain the knowledge about the processes of selection and evaluation of OSS components?

5.3 Interview Definition

5.3.1 Object of Study

The object of study is the selection and evaluation of open source components in industry. The interviews want to explore what are the practices in the industry. This empirical study is only explorative, but the final goal is to find out practices that can be used as input in the decision-making in any improvement seeking software organization [Wohlin et al.2000].

5.3.2 Purpose

The purpose of the interview is to evaluate the processes of selection and evaluation of open source components.

5.3.3 Quality Focus

The quality focus is the effectiveness of the processes. It is very important to discover an effective way to select components, and that would be very useful for the software industry.
5.3.4 Perspective

In this empirical study, the perspective is from master students and researcher’s point of view. The interviews are performed by two master students (see Appendix B) but under the supervision of a researcher. Also, other researchers contributed to the interview guide with useful comments.

5.3.5 Research Context

The depth study was performed in the context of the Norwegian COSI project, which is a sub-part of the international COSI project. A short description of the Norwegian COSI project and the three industrial partners are presented first, followed by short description of the other two software companies who agreed to participate to the interview. The interview was run by two master students (see Appendix B).

The Norwegian COSI project

The Norwegian sub-project is sponsored by the Norwegian Research Council and led by ICT-Norway. In addition to ICT-Norway, there are also four industrial partners (Keymind, Linpro, ITFarm and eZ Systems), who will be described below, and one academic partner, NTNU.

The goal of the Norwegian project is process improvement of development processes related to open and inner source software development. The Norwegian project is further responsible for deliverables to the European project related to development processes. Each industrial partner is responsible for reporting their processes related to open source, and NTNU is gathering these practices and presenting them to the European project.

In the next subsections, the three industrial partners are briefly described, followed by the other two software companies.

Keymind Computing AS

Keymind Computing AS, formed in 1998, is a small consulting company focusing on developing, maintaining and supporting IT solutions. They have six employees, located in three different locations. Their main objective is digital screening within the Norwegian health care.

Keymind is a user of open source components and open source tools. These tools and components are used in the development of their solutions. Keymind believes that using open source can give them further economical benefits than they have today. The cost of maintenance is proportional with the amount of code, and Keymind believes they can cut maintenance costs by using code maintained by a community. Keymind are participating in the COSI project to be able to analyze and further develop their use of OSS. For further information about Keymind Computing, see their website at http://www.keymind.no/.

Linpro

Linpro has since their beginning in 1995 focused on services around the Linux platform and the use of open source software. Linpro is growing and with more than 70 employees, they are probably the largest open source based consulting company in Norway.
Linpro has competence within several areas like Linux, security, project management, network, databases, several programming languages, and operation of IT systems. They provide development and integration services, support, training and courses. They also provide operation of IT systems for several big Norwegian customers like Elkjøp, A-pressen, Netcom and Telenor. They have also their own webshop which is used to sell Linux related products and some server hardware.

Linpro provides services on an open source platform, they develop software using open source tools and components. All this makes Linpro a very interesting partner for the COSI project. Linpro wishes to improve their cost/risk-evaluation, delegate more of their software development to open source communities, and improve their relationship in the open source community. More information about Linpro is available at http://www.linpro.no.

**SINTEF ICT, Software engineering, Safety and security**

SINTEF ICT offers integrated research-based knowledge through access to a broad competence and technology platform within ICT. The Software Engineering, Safety and Security department has focus on increasing and maintaining safety and security of computer based systems, which is vital to our society. Their contribution to making the world a better place is through ensuring that software is developed in a reliable, secure and efficient way, as well as safety assessment of technical systems and increasing the awareness of information security. Their area of expertise include: Software Process Improvement and Knowledge Management, Software Architectures, Intelligent Transport Systems and Services, Information Security, System Safety, and Health Care Informatics.

The department uses open source in different projects. More information about SINTEF ICT, Software engineering, Safety and security is available at http://www.sintef.no/content/page3____6456.aspx.

**FAST ASA**

Their business is Enterprise Search. Since the start of the company in Norway in 1997, they have grown rapidly to become a global organization with offices across six continents. The areas that FAST provides solutions for are: Corporate Operations, eCommerce, Information Management, Market Management, Mobile, Online Media, OEM Search Integration, Risk Management, Surveillance and Enforcement. More information about FAST ASA is available at: http://www.fast.no/.

**eZ Systems**

eZ Systems, established in 1999, has since their start-up had success with their Entreprise Open Source Content Management System, eZ publisher. The company has their main office in Skien, Norway, but they are also located in Oslo (Norway), Ukraine, France, Canada and Germany. They have more than 70 employees from close to 20 countries, and are still growing. Through participating in the COSI project, eZ wishes to gain insight into how they can grow to become a larger global actor than they are today.

Their most important product is eZ publisher. It is an open source and freely available content management solution developed in PHP. eZ Systems provides services, product responsibility, and other related software. eZ publisher is licensed under the GPL license and their own proprietary licenses. Around their product they have a community
consisting of customers from all over the world, and more than 20000 registered users in their forum. More info can be found at their website http://ez.no/.

**Statoil ASA**

Statoil ASA is an integrated oil and gas company with substantial international activities.

Several main vendors provide application maintenance, support and development for Statoil, for example IBM.

More info can be found at their website http://statoil.com

**Sun**

A singular vision (i.e “The Network Is The Computer”) guides Sun in the development of technologies that power the world’s most important markets. Sun’s philosophy of sharing innovation and building communities is at the forefront of the next wave of computing: the Participation Age. Sun can be found in more than 100 countries.

Sun creating open source:

Sun is the biggest single contributor of open source code in the world. Everything Sun creates of software is supposed to be open source over time. Read more at http://www.sun.com/software/opensource/learnmore.jsp.

Sun using open source:

Sun uses and distributes a lot of open source components. OpenSolaris is boundled with tons of open source code. Sun also uses open source internally as tools to develop software.

Maybe one of the latest important news about Sun related to open source is the fact that Sun open sources Java.

Sun believes deeply in creating communities and sharing innovations and technologies to foster more participation. Today in a historic move, Sun is opening the door to greater innovation by open sourcing key Java implementations—Java Platform Standard Edition (Java SE), Java Platform Micro Edition (Java ME), and Java Platform Enterprise Edition (Java EE)—under the GNU General Public License version 2 (GPLv2), the same license as GNU/Linux. More info about this at: http://www.sun.com/2006-1113/feature/

More info about Sun in general can be found at: www.sun.com.

5.4 Interview Planning

The interview definition presented in the previous sub-chapter is input for the interview planning. The output of the interview planning is the interview design.

The interview is designed to provide qualitative answers to the research questions. The design of the interview is explorative with the main objective to discover new aspects of open source, aspects described by the research questions.

5.4.1 Context Selection

The interview is performed by students, but under the supervision of a researcher. The interview addresses real problems of the software industry. However, because of the limited number of respondents, the results are not valid to the general software engineering domain.
5.4.2 Respondents

The interviews were conducted at 7 Norwegian software companies, where managers, software architects or developers were interviewed. The companies were selected using *convenience sampling*, which means that the companies were selected because they participate in the COSI project in Norway, and therefore are available for interview. Other software companies which are not part of the COSI project agreed to participate to the interview. The selected companies have the specific role in open source (i.e user of open source component).

The COSI partners who participated to the interview were: Keymind, Linpro and eZ systems.

Other companies who agreed to participate to the interview are:
- SINTEF IKT (Trondheim), Software Engineering and security department
- FAST ASA (Oslo)
- Sun (Trondheim)
- Statoil (Trondheim)

5.4.3 Interview Design

The interview guide used is presented in the Appendix D. It was written in English, even if was only used to interview Norwegian software companies.

The interview contains both open questions but also closed questions with predefined answers to choose from.

The parts 1, 3, 4 and 5 of the interview guide apply to OTS in general. Only the part 2 of the interview guide is OSS specific.

During the interview it is still possible to ask follow-up questions to further elaborate on areas where more information is needed.

5.4.4 Interview Limitations

The interview was limited to few Norwegian software companies and this is a limitation. Performing interview is a time consuming task and it is difficult to interview a representative number of responding companies. Therefore, the validity of the results will be discussed based on the limited number of responding companies. Many problems might occur when designing and performing an interview. Some problems are related to the sample of respondents answering the interview; other problems are categorized as non-response or response errors. Non-response errors occur when the respondent for some reason will not answer the interview. This reduces the sample size and has impact on the validity of the results. Response errors occur when the reported data does not match the actual true data.

5.5 Collecting Evidence

We will discuss how we collected data and information from our interviews. Erik Løvland conducted two interviews with Linpro and Sun. I conducted five interviews with Keymind, SINTEF ICT, FAST ASA, Statoil and eZ systems. Part 6 about COTS components is not related to this depth project, therefore that data will not be analyzed in this report.

The interview guide was in English and therefore we conducted the interviews in English. To allow the interview objects to be prepared for the interview, we sent them the
questionnaire a few days before the interview. Only the interviews with SINTEF ICT, Statoil ASA and Sun were face-to-face, the others were done by phone. Each interview was conducted by one student who recorded the interview in addition to marking the answers directly on the paper version of the interview guide. Some interview objects started to talk about a specific question without being asked directly, some were asked by us. The average time spent on the interview was about 50 minutes.

5.6 Analyzing the Evidence

When the interviews finished, we analyzed the data by reading the answers but also listening to the audio records. We found both qualitative and quantitative results. To back-up our findings, we included some statements in the thesis as quotes. The results of this study will be used to improve the questionnaire. The findings from this study should be approved by a larger survey, which is expected to be done in the spring of 2007 in a similar context.

5.7 Summary

In this chapter, we presented a summary of the empirical strategies with focus on explorative survey. Defining the research goal and research questions is an essential part of the chapter. We have defined a company role: user of open source components. The user of Open Source Components is a company which develops software using open source tools or components. We have defined our research questions by identifying research problems in literature but also considering what would be beneficial for the software industry. The interview definition and the interview planning are necessary steps before performing the interview.
Part 4 – Results, Discussion and Conclusions

6. Results
This chapter starts by presenting the link between each research question and the corresponding questions from the interview guide in Table 4. For example, the research question 1 is answered with the help of the first questions from the interview guide.

<table>
<thead>
<tr>
<th>Research question</th>
<th>The corresponding questions in the interview guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>2</td>
<td>5, 6, 7, 8, 9</td>
</tr>
<tr>
<td>3</td>
<td>12 – 24</td>
</tr>
<tr>
<td>4</td>
<td>10, 11</td>
</tr>
<tr>
<td>5</td>
<td>25, 26, 27</td>
</tr>
</tbody>
</table>

Table 4: The corresponding questions in the interview guide for each research question

6.1 Create Hypotheses
This study is based on an explorative research goal. Therefore, several hypotheses will be created based on the results to the research questions. In the next section, several observations will be formulated as hypotheses. These hypotheses could be evaluated by other qualitative or quantitative studies.

6.2 RQ1: Who initiates and performs the work related to OSS harvesting and when in the development process?
This research question is answered by the first 4 questions of the interview guide. We have asked if is the IT manager, project manager, software architect, software developer, tester or customer who initiates and does the work related to harvesting in a company.
Regarding the initiation phase, most respondents answered that is the software architect who does it.
Also most respondents answered that it is the software developer who does the work related to harvesting and that it is the project manager or the software architect who takes the final decision about integrating an OTS component.

In small companies, a person can have several roles (e.g. software developer and project manager), as said one of the respondents.
In large companies, not only it exist a clear distinction between software developers, software architects, project managers, but it can also exist different types of architects. Our respondents from a large company said that: “we have multiple software architects responsible for different areas in the company. We have different types of architects that take decisions about selecting and evaluating OTS components.”

The answers for question number 4(i.e “When in the development process does one selects and evaluates OTS-components?”) are quite spread, but the most relevant are:
The selection and evaluation of OSS components are done frequently, often or sometimes on the pre-study, requirement/analysis and overall design, sometimes on the detailed design and coding. One respondent from a small company said that depending on the component size, they select and evaluate large OSS components often on the overall design and often on the detailed design for smaller components. If the company uses Agile development or similar, it can also be done incrementally throughout the project.

**Observation:** Usually it is the software architect who initiates the work related to harvesting. The work of harvesting is usually done by the software developer and the final decision about integrating an OTS component is usually done by project manager or software architect.

### 6.3 RQ2: What are the most important issues when searching for OSS components: business, technical, organizational, personal?

#### Business issues

We listed the following business issues and the respondents indicated how important each specific issue is: licenses are cheaper, installation cost is smaller, maintenance cost is smaller, learning cost is smaller, to be more independent from the big companies’ prices, support, want to support the OSS movement, it is recommended by our partners, our developers work on open source projects in their spare time, we want to increase our skills in this area, the market is looking for this.

The most relevant findings are:
- The *license price* seems to be a bit important (for 5 of 7 respondents).
- The *maintenance cost* was important for three companies (two medium-sizes and one large). For small and medium-size companies, this was a bit important. In big organizations, the applications in which the components are integrated usually have a long life-cycle, have complex architectures therefore the maintenance cost is important, as indicated by one of our respondents.
- Regarding the fact that the *component is recommended by partners*, it seems that the companies care about the partners opinions.

**Observation:** The license price is the most important business issue when selecting and evaluation OSS components, but also the maintenance cost is quite important.

#### Technical issues

We listed the following technical issues: functionality, quality of code and design, quality of architecture, quality of documentation, security, performance, integration with other products, support for standards, and standard compliance.

All the respondents pointed out *functionality* as very important.

The *quality of code and design* and the *quality of architecture* are very important for 4 of 7 respondents. The *quality of documentation* is important for 5 of 7 respondents.
When OSS components are integrated into large and complex applications or systems, then the quality of code and design, of architecture and of documentation are very important: “We have a life-cycle management perspective on all software including open source software. That means that we have to be able to integrate and maintain it along with all the other software that we are using. That makes the quality of code and design, the quality of architecture and the quality of documentation very important for us.”, said one of our respondents from a large company.

When selecting and evaluating OSS components, the security and performance are more or less important, depending on the type of company and the type of the product where the OSS components are integrated.

One respondent from a small company said that security and performance are important only for components which deal with security and performance, it thus depends on the type of component.

Security and performance are not a priority for research products, because they are not sold afterwards, as said by one of our respondents.

Our respondent from a large company said that: “if the OSS components don’t deal with security, they still have to fit into our security architecture.”, so security is very important for them anyway.

Integration with other products, supports for standards and standard compliance are important or very important for 6 respondents.

Observation: The functionality is the most important technical issue when selecting and evaluating OSS components. When the functionality criterion is met, the quality of code and design, of architecture and documentation are also important. Support for standards and standard compliance play also an important role.

Organizational issues
We listed the following organizational issues of the OSS community: vitality of the community, quality of end-user support, momentum of releases, quality of project site, availability of roadmap/plan, and availability of component for test and use.

The most relevant findings are:
Vitality of community and the Momentum of releases (frequency of releases) are a bit important, have some importance, are important or very important.
The availability of component for test and use is very important.

H1: The availability of component for test and use is the most important organizational issue when selecting and evaluation OSS components.

Professional issues
We listed the following professional issues:
• people select components they have used already. All respondents answered that it is important or very important.
• people select components they have heard about. Most of all respondents answered that it has some importance or it is important.
• people select components written in programming languages they know. All respondents answered that it is important or very important.

Observation: The fact that the people have experience with the OSS component or that the component is written in programming languages people know, are important issues when selecting and evaluating OSS components.

Regarding the question 9 (i.e. what makes you discard an OSS component), we wanted to find out what are the properties of the OSS component which make a company to discard it. The answer to this would be very beneficial for industry, so they know which kind of components they should avoid.

The most relevant answers are:
5 of 7 respondents answered that unsuitable license contributes very large. 4 of 7 respondents answered that the bad documentation or no documentation contributes largely to discard an OSS component. Also 4 respondents answered that the bad code quality contributes very largely.

H2: Unsuitable license and bad code quality are the most important properties which makes a company to discard an OSS component.

6.4 RQ3: What is the process of selecting and evaluating OSS components?
This research question is answered by the questions 12 to 24 from the interview guide. Below we present the answers to the questions 12 to 24, and describe new hypotheses when possible.

12. What type of OSS does your company use already?
The most used OSS components are:
• LAMP (Linux, Apache, MySQL, PHP)
• Desktop applications. FireFox and OpenOffice are the most common in this category
• Development tools. Subversion, CVS, Emacs and the GNU Compiler are the most common in this category
• Languages. Perl, Python and Java are the most common here
• Graphical user interface. KDE and GNOME are the most common here
• Security. Open SSH and Open SSL are the most common here

13. How would you describe your software process model?
The results of this question show that a company can have one or several process models, or can move from an old process to a new one, as indicated by one of our respondents (the company was moving from Waterfall to an Agile process model). The results of this question are:
• 5 companies of 7 use Agile methods
• 2 use Waterfall
• 2 use Adaptive Software Development
• 2 use Extreme programming
• 1 uses Evolutionary Prototyping and
• 1 uses Unified Process.
Observation: The most common categories of OSS components used in industry are: LAMP, desktop applications, development tools, languages, graphical user interface and security. Agile methods are the most common used process models.

14. How many selection processes do you perform per year?
The answers are:
- Between 1 and 4: 1 company
- Between 5 and 10: 4 companies
- More than 10: 1 company

15. How much time does it take to perform a selection and evaluation process?
The findings show that a process can take from a few days to one or several months.
- 4 respondents answered that they use 5 to 10 days to select and evaluate a component.
- 1 respondent answered that they use between 1 and 5 days and
- 1 respondent answered that they use more than 10 days for a component.

We believe that if the OSS component is large or complex and it is going to be integrated into a large or complex application, then companies use a lot of time for a proper selection and evaluation process. This was also indicated by one of our respondents from a large company who said that is very important to use time in order to select the proper component, instead of later using much more time because the selected component was not the right one.

16. Does your company use any formal process for selection and evaluation of OTS components?
Only one company answered affirmatively, the others said that they have never heard about formal processes for selection and evaluation of OTS components.

17. Which formal activities do you perform?
Since the companies we have interviewed do not use any formal processes, we didn’t ask which formal activities they performed.

**H3**: The number of selection processes and the time used for selection and evaluation is proportional with the complexity of the component to be integrated

**H4**: Companies do not use any formal processes for the selection and evaluation of OTS components.
This hypothesis is also supported in the PhD thesis of [Li2006].

18. How do you find OTS components?
The answers are:
- 5 of 7 respondents answered that their company search on specialized sites or search using specialized search engines.
- 4 respondents answered that their company also get components from a colleague or a friend.
- 1 respondent answered that they find components by reading papers.
1 respondent answered that their company use an internal repository.

**H5: Most companies find OTS components by searching on specialized sites or by searching using specialized search engines.**

19. Are the OTS components tested before integration?
All the respondents answered that they test the OTS components before integration. This is usually done by making a small prototype and testing it.

**H6: The OTS components are usually tested before integration by making a small prototype.**

20. Have the selected OTS component ever been replaced with a different component at some point during the development process?
6 respondents answered that this happened. Being asked in which phase, 3 of them said that it happened in the coding phase.

21. Have you noticed in the end of the development process that your selected component is not suitable?
5 respondents answered that this happened. Being asked in which phase, their answers are very different.

**H7: Usually, the companies realize in the end of the development process that the selected OTS component is not suitable.**

22. Did you restart the process of selection and evaluation of an OTS component?
Only 3 respondents confirmed this, the other said no or they didn’t know. The 3 respondents who confirmed this usually use 3-5 days to do a new selection and evaluation process. A respondent from a large company confirmed that: “we use only 5 days because we had a heavy evaluation the last time”.

23. Did you start building an internal component instead of searching for an OTS component?
It is difficult to make any conclusion about this issue based on the answers.

24. In which phase of the development process did you decide not to make the component yourself, but acquire an OTS component?
This question is similar with the question number 4, but not exactly. We got a lot of “Don’t know” for this question, which means that the respondents found difficult to understand it. Therefore, this question should be re-phrased.

**6.5 RQ4: What versions are considered and how to deal with new versions?**

Different versions of an OTS component usually are available on different web sites. 3 respondents answered that it is the last version they consider and 3 answered that it is the last stable version they consider when selecting and evaluating OTS components.
To know when to update to a new version of an OTS component is an important decision because it can be a time consuming task and therefore the advantages of having the new component should offset the time consumed for updating. Most of the respondents answered that they update to a new version of the OTS component when additional functional is needed or when bugs need to be corrected.

**H8:** Companies usually consider the last or the last stable version when selecting and evaluating OSS components.

**H9:** Companies update to a new version when additional functionally is needed or when bugs need to be corrected.

6.6 RQ5: How to maintain the knowledge about the processes of selection and evaluation of OSS components?

Regarding the question 25 (i.e “Do you document the decision besides selecting and evaluating OTS components?”), 5 of 7 respondents answered that they do. Usually this is done as part of the project or the product. In the case of a small company, this is done by writing a short document which describes the alternatives. One of our respondents from a large company said that they document the decision as part of the product documentation. Being asked if they have a common documentation across different products/projects, he answered that “there is a corporate web site that is maintained by one of the software architects, which is linked to the product documentation.” Being asked if they reuse this information later, 4 of them answered that they do, in a smaller or larger extent.

**H10:** Usually people document the decision besides selecting and evaluating OTS components and this is usually done as part of the product/project documentation.

Regarding the question 26 (i.e “Do you keep a local knowledge repository about the selected OTS components?”), 5 of 7 respondents answered that they do not. Our respondent from a large company said they have a common framework within the company for each development platform. For example, they have a Java enterprise framework.

**H11:** Small or medium size companies usually do not keep a knowledge repository about the selected OTS components, only the large companies do.

Regarding the question 27 (i.e “Do you have a person who is responsible for the OTS component?”), 5 respondents answered that they do. The other two respondents answered that they do not formally have such a role. Our respondent from a small company said that: “For each component, there is one person responsible in our company but it is not the same person all the time. Their responsibility is to understand each OTS component”. In the case of a large company, there are about 5 persons responsible for all the components.
H12: Usually there are people responsible for the OTS components in each company.
7. Discussion

This chapter will discuss the results from the explorative survey. We will start by presenting the main contributions. Then, a discussion about the validity of the results will follow. Finally, improvements of the survey are suggested.

7.1 Main Contributions

7.1.1 Literature study

The first contribution of this study is the literature study. When performing the literature study, the focus was on creating a short introduction to open source by presenting the relevant issues (e.g. “The Roles in OSS Community and Open Source Projects”). This introduction should help the reader to better understand the rest of the literature study.

Because the selection and evaluation of OSS components happened as part of a software development process, the chapter “Software development processes for Component Based Systems (CBS)” is necessary before presenting the state-off-the-art of selection of OTS components.

The knowledge gained from the literature study has been used to define the research questions. When creating the research questions we always had in mind to define research questions which could be beneficial for the software industry.

7.1.2 New Knowledge

The most important contribution of this thesis is the new knowledge, which has been discovered throughout the study. This knowledge can be divided into: answers to the research questions and hypotheses.

**Answers to the research questions**

The answers to the research questions are provided in chapter 6, as quantitative and qualitative data.

**Hypotheses**

We have proposed 12 hypotheses based on the results of the interviews we performed with 7 Norwegian companies. These hypotheses cannot be considered as definitive proof but can be tested further by other studies. Our hypotheses are presented in Table 5.

<table>
<thead>
<tr>
<th>Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1: The availability of component for test and use is the most important organizational issue when selecting and evaluation OSS components.</td>
</tr>
<tr>
<td>H2: Unsuitable license and bad code quality are the most important properties which makes a company to discard an OSS component.</td>
</tr>
<tr>
<td>H3: The number of selection processes and the time used for selection and evaluation is proportional with the complexity of the component to be integrated.</td>
</tr>
<tr>
<td>H4: Companies do not use any formal processes for the selection and evaluation of OTS components.</td>
</tr>
<tr>
<td>H5: Most companies find OTS components by searching on specialized sites or by searching using specialized search engines.</td>
</tr>
<tr>
<td>H6: The OTS components are usually tested before integration by making a small</td>
</tr>
</tbody>
</table>
H7: Usually, the companies realize in the end of the development process that the selected OTS component is not suitable.

H8: Companies usually consider the last or the last stable version when selecting and evaluating OSS components.

H9: Companies update to a new version when additional functionally is needed or when bugs need to be corrected.

H10: Usually people document the decision besides selecting and evaluating OTS components and this is usually done as part of the product/project documentation.

H11: Small or medium size companies usually do not keep a knowledge repository about the selected OTS components, only the large companies do.

H12: Usually there are people responsible for the OTS components in each company.

Table 5: Our hypotheses

7.1.3 A Platform for Future Work

Our work tries to answer as many questions as possible about selection and evaluation of OTS components (OSS in particular). We have gained new understanding which will be used the next semester for a larger survey. Our work may also be interesting to others.

7.1.4 Results discussion

Below we compare our results with those described in literature if any exists, and we relate the research questions to our hypotheses.

RQ1: Who initiates and perform the work related to OSS harvesting and when in the development process?
   RQ1.1: Who initiates the OSS harvesting in a company?
   RQ1.2: Who does the work related to harvesting?
   RQ1.3: Who takes the final decision about integrating an OTS component?
   RQ1.4: When in the development process does one select and evaluate OTS-components?

We did not find a specific description of this in literature, therefore we found it useful to know which person/role in a company is the most appropriate for initiating component selection, doing the work related to harvesting, and in the end taking the final decision whether to acquire components. Discovering when in the development process is best to select and evaluate is also important to know more about since we could not find any relevant empirical studies in the literature.

We could not make any hypotheses based on our results, but we have some observations that can at a later stage be transformed to hypotheses. This will, however, require further studies to find and express the context of the observations.

RQ2: What are the most important issues when searching for OSS components: business, technical, organizational, personal?
Section 4.2 of the report presents the dimensions considered when selecting COTS, the criteria that OSS must meet, and some elements of open source maturity. These are described in academic papers or books and helped us to define four categories of issues that need to be considered.

Based on the answers to this research question, we have proposed the following hypotheses:

| RQ2 | H1: The availability of component for test and use is the most important organizational issue when selecting and evaluation OSS components. |
|     | H2: Unsuitable license and bad code quality are the most important properties which makes a company to discard an OSS component. |

Table 6: RQ2, H1 and H2

**RQ3:** What is the process of selecting and evaluating OSS components?

Section 4 of the report presents the relevant literature findings about COTS, OSS, and the common and different issues. This study is also based on the findings of other empirical studies, which are described in the section 1.1 of the report.

Based on the answers to this research question, we have proposed the following hypotheses:

| RQ3 | H3: The number of selection processes and the time used for selection and evaluation is proportional with the complexity of the component to be integrated. |
|     | H4: Companies do not use any formal processes for the selection and evaluation of OTS components. |
|     | H5: Most companies find OTS components by searching on specialized sites or by searching using specialized search engines. |
|     | H6: The OTS components are usually tested before integration by making a small prototype. |
|     | H7: Usually, the companies realize in the end of the development process that the selected OTS component is not suitable. |

Table 7: RQ3, H3, H4, H5, H6 and H7

**RQ4:** What versions are considered and how to deal with new versions?

We could not find any empirical studies regarding which version of the OSS components it is best to integrate and how to deal with new versions. Several versions of an OSS component are available, usually on a web site. It is important to know which version to integrate and then how to deal with new versions.

Based on the answers to this research question, we have proposed the following hypotheses:

| RQ4 | H8: Companies usually consider the last or the last stable version when selecting and evaluating OSS components. |
|     | H9: Companies update to a new version when additional functionally is
Table 8: RQ4, H8 and H9

**RQ5:** How to maintain the knowledge about the processes of selection and evaluation of OSS components?

One of the papers, [Ruffin and Ebert2004], shortly describes a database of a large company. This database contains experiences with open source software. The results of this question confirmed that large companies store and maintain such information.

We also thought that would be beneficial to know if companies document the decisions besides selecting and evaluation OSS components.

Based on the answers to this research question, we have proposed the following hypotheses:

<table>
<thead>
<tr>
<th>RQ5</th>
<th>H10: Usually people document the decision besides selecting and evaluating OTS components and this is usually done as part of the product/project documentation.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H11: Small or medium size companies usually do not keep a knowledge repository about the selected OTS components, only the large companies do.</td>
</tr>
<tr>
<td></td>
<td>H12: Usually there are people responsible for the OTS components in each company.</td>
</tr>
</tbody>
</table>

Table 9: RQ5, H10, H11 and H12

7.2 Validity

Validity is related to how much we can trust the results. [Wohlin et al.2000] state that adequate validity refers to that the results should be valid for the population of interest. First of all, the results should be valid for the population from which the sample is drawn. Secondly, it may be of interest to generalize the results to a broader population.

The four types of validity described in [Wohlin et al.2000] are: *conclusion, internal, construct and external validity*.

7.2.1 Conclusion Validity

Conclusion validity is concerned with the relationship between the treatment and the outcome [Wohlin et al.2000].

The first threat is *low statistical power*. Because we had only 7 respondents, the statistical power of this work is very low. This is a problem we are aware of and therefore a more thorough study needs to be performed in the future.

Another threat is *fishing and error rate*. The researcher may influence the result by looking for a specific outcome. We do not think that we have influenced in any way the results.

Several other threads are presented in [Wohlin et al.2000], but are not relevant for us.
The student’s lack of experience creating the interview guide and conducting interviews is potentially another problem.

7.2.2 Internal Validity
If a relationship is observed between the treatment and the outcome, we must make sure that is a causal relationship, and that it is not a result of a factor of which we have no control or have not measured [Wohlin et al.2000].
The relevant threats are:
- Maturation
Subjects may be affected negatively during the interview. In our case, subjects may be tired (some interviews were performed late on the day) and some subjects may not be so motivated to answer.
- Instrumentation
This is the effect caused if the artifacts of the interview are badly designed. We realized that a part of the interview guide was not very well designed, therefore this may negatively affect the interview.
- Selection
Depending on how the subjects are selected from a larger group, the selection effects can vary [Wohlin et al.2000].
Using convenience sampling, as we did in our research, may reduce the internal validity of the results. Our group of respondents is not representative for the whole population and this may affect the results. It would have been preferable to use some kind of randomization to select our sample, but this was not possible.

7.2.3 Construct Validity
Construct validity concerns generalizing the results of the experiment to the concept or theory behind the experiment [Wohlin et al.2000].
[Jacobsen2005] mentions measuring the right thing as one problem.
Questions and statements in the interview may have been misunderstood or misinterpreted because they were improperly phrased. If the respondents misunderstood something, they may have answered something else.
We realized that some sentences could be better phrased. Since the interview was done by phone or face-to-face, when people had something unclear, we gave them more information.
[Rabson2003] mentions testing and reviews as the best ways of ensuring construct validity. Our supervisor and other researchers have reviewed the interview guide but we did not test the interview guide before the interviews.

7.2.4 External Validity
The external validity is concerned with generalization. If there is a crucial relationship between the construct of the cause, and the effect, can the result of the study be generalized outside the scope of our study? [Wohlin et al.2000]
There are three types of interactions with the treatment: people, place and time. The relevant for us is the people.
Having a subject population not representative for the population we want to generalize, may affect the result. We had only 7 respondents and they are possible not representative for a larger population.

7.3 Improvements
We think is important to describe ideas of improvement which will be useful both for us when performing a new study next semester, but also for others.

General improvements
- Increase the sample size
- Use random-sample technique if possible
- Increase the motivation of the respondents
- Differentiate between “large” and “small” components and tools.

Improving the interview guide
- Add a new item to question 7: Availability of Bugzilla or other bug-issue-tracking system on the project site
- Change the first item of the question 9 to: “Is ill-suited for the technical skills of people responsible for integrating and maintain it”
- Add Subversion as Development tool at question 12
- Add Java as Language at question 12
- More focus on standard compliance, research is missing about it.
- Improve the questions in the part 4 of the interview guide: “Process of selecting and evaluating OTS components”
  - If the companies do not use any formal process to select and evaluate OTS components, ask anyway which activities they perform.
  - We think that many respondents did not understand very well the question 24, therefore it should be redefined or removed.
- Rewrite the question 21, should be: “Have you noticed in the end of the development process that your selected component is not suitable?”
8. Conclusions and Future Work

8.1 Conclusions
Empirical research is performed to verify theories, develop new theories or extend the existing ones, and improve practice. This study is mainly used to improve the software development practice in industry, particularly to improve the practice of selection and evaluation of OSS components.

We have used the role: user of open source, because this is the most appropriate for the research we have performed. More and more companies integrate open source components in their products because the benefits are large. Therefore, improving the practice about the selection and evaluation of OSS components may help software companies to decrease the time spent. If the time spent is too large, this can offset the advantages of integrating OSS components.

The results of the interview are presented and 12 hypotheses are formulated based on this. These hypotheses will be tested by a larger survey next semester.

The literature study was very useful to gain understanding about the state-of-the-art but also to define the research questions.

8.2 Future Work
As stated already, this explorative study will be used as input for a larger survey which will be performed next semester in a similar context but with a larger sample size.
References


[Li2006] Li, J. “Process Improvement and Risk Management in Off-The-Shelf Component Based Development” PhD theses at NTNU, 2006:84, pp 29


## Appendix A: Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBS</td>
<td>Component-based system</td>
</tr>
<tr>
<td>Copy-left</td>
<td>Copy-left is a general method for making a program or other work free and requiring all modified and extended versions of the program to be free as well.</td>
</tr>
<tr>
<td>COSI</td>
<td>Co-development using inner &amp; Open source in Software Intensive products</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial off the shelf</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent Versions Systems</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>Hacker</td>
<td>A person who is very skilled at computer programming and spends a lot of time programming</td>
</tr>
<tr>
<td>IDI</td>
<td>Department of Computer and Information Science</td>
</tr>
<tr>
<td>ITEA</td>
<td>Information Technology for European Advancement</td>
</tr>
<tr>
<td>NTNU</td>
<td>Norwegian University of Science and Technology</td>
</tr>
<tr>
<td>OTS</td>
<td>Off-the-Shelf</td>
</tr>
<tr>
<td>OSS</td>
<td>Open Source Software</td>
</tr>
<tr>
<td>Qualitative</td>
<td>Concerned with information as text</td>
</tr>
<tr>
<td>Quantitative</td>
<td>Concerned with information as numbers</td>
</tr>
</tbody>
</table>
Appendix B: The Open Source Definition

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the
program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

*10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.
Appendix C: Interview guide

Autumn 2006

Marinela Gerea: Selection and Evaluation of Open source components

Erik Løvland: Studies of component-based development
Definitions the reader may read before the interview start:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTS</td>
<td>Off-The-Shelf</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial-Off-The-Shelf</td>
</tr>
<tr>
<td>OSS</td>
<td>Open Source Software</td>
</tr>
<tr>
<td>In-House-Build</td>
<td>Components build by your company.</td>
</tr>
</tbody>
</table>
Part 1. Who initiates the OTS harvesting and when in the development process?

1. Who initiates the work related to OTS harvesting in a company?

Please select one or several alternatives.
- [ ] IT Manager
- [ ] Project manager
- [ ] Software architect
- [ ] Software developer
- [ ] Other (please specify): ________________

2. Who does the work related to OTS harvesting in a company?

Please select one or several alternatives.
- [ ] IT Manager
- [ ] Project manager
- [ ] Software architect
- [ ] Software developer
- [ ] Other (please specify): ________________

3. Who takes the final decision about integrating an OTS component?

Please select one or several alternatives.
- [ ] IT manager
- [ ] Project manager
- [ ] Software architect
- [ ] Software developer
- [ ] Tester/Quality assurance
- [ ] Customer
- [ ] Other (please specify): ________________
4. When in the development process does one select and evaluate OTS-components?

☐ Please check it if it applies only for selecting OTS components
Mark one alternative per row

<table>
<thead>
<tr>
<th>Phase</th>
<th>Frequently</th>
<th>Often</th>
<th>Sometimes</th>
<th>Seldom</th>
<th>Never</th>
<th>Don’t know</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-study</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements/Analysis</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall design</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detailed design</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incrementally throughout the project</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Part 2. Issues considered when searching for an OSS component

5. What are the important business issues you consider when selecting and evaluating OSS components in general?

Mark one alternative per row

<table>
<thead>
<tr>
<th>Issue</th>
<th>Not at all important</th>
<th>A bit important</th>
<th>Some importance</th>
<th>Important</th>
<th>Very important</th>
<th>Don’t know</th>
</tr>
</thead>
<tbody>
<tr>
<td>Licenses are cheaper</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation cost is smaller</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Maintenance cost is smaller</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learning cost is smaller</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>To be more independent from the big companies’ prices, support.</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Want to support the OSS movement</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>It is recommended by our partners</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Our developers work on open source projects in their spare time</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>We want to increase our</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>skills in this area</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The market is looking for this</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. What are the important technical issues you consider when selecting and evaluating an OSS component?

Mark one alternative per row

<table>
<thead>
<tr>
<th></th>
<th>Not at all important</th>
<th>A bit important</th>
<th>Some importance</th>
<th>Important</th>
<th>Very important</th>
<th>Don’t know</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>Quality of code and design</td>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality of architecture</td>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality of documentation</td>
<td></td>
<td>1</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration with other products</td>
<td></td>
<td>1</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support for standards (have standard-based APIs)</td>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard compliance</td>
<td></td>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. What are the important organizational issues of the OSS community you consider when selecting and evaluating an OSS component?

Mark one alternative per row

<table>
<thead>
<tr>
<th></th>
<th>Not at all important</th>
<th>A bit important</th>
<th>Some importance</th>
<th>Important</th>
<th>Very important</th>
<th>Don’t know</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vitality of community</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality of end-user support</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Momentum of releases</td>
<td></td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Quality of project site</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability of roadmap/plan</td>
<td></td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The availability of component for test and use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

By Vitality of community we mean: Activity on mailing lists/email, Number of downloads, Number of page views, Frequency of new releases, Time to fix bugs, Time to implement new functionality.
8. What are the important professional issues you consider when selecting and evaluating an OSS component?

Mark one alternative per row

<table>
<thead>
<tr>
<th>People select components they have used already</th>
<th>Not at all important</th>
<th>A bit important</th>
<th>Some importance</th>
<th>Important</th>
<th>Very important</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>People select components they have heard about</th>
<th>Not at all important</th>
<th>A bit important</th>
<th>Some importance</th>
<th>Important</th>
<th>Very important</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>People select components written in programming languages they know</th>
<th>Not at all important</th>
<th>A bit important</th>
<th>Some importance</th>
<th>Important</th>
<th>Very important</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. To what extent do the following properties make you discard an OSS component?

Mark one alternative per row

<table>
<thead>
<tr>
<th>Is ill-suited for the technical skill set of people responsible for running it</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Little community activity</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Does not have momentum</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Has unsuitable license</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>With bad or no documentation</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No commercial company supports it</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Is dependent from too many other components</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Without road map</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Without reputation</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Has bad code quality</th>
<th>Not at all</th>
<th>Small</th>
<th>Some</th>
<th>Large</th>
<th>Very large</th>
<th>Don't know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part.3 Versions issues OTS

10. Which version of the OTS component do you normally consider when selecting and evaluating?

Please select one:
- The last version
- The version that most people use
- The most stable version
- The version that provides more functionality
- Other (please specify): ______________________

11. When do you update to a new version of the OTS component?

Please select one or several:
- Each time you release a new version of the system containing one or more OSS components
- Each time a new version of the OSS component is coming
- When additional functionality is needed
- When bugs need to be corrected
- Other (please specify): ______________________

Part 4. Process of selecting and evaluating OTS components

12. What type of OSS does your company use already?

- LAMP (Linux, Apache, MySQL, PHP)
- Desktop applications (Evolution, FireFox, OpenOffice, etc)
- Development tools
  - Bugzilla
  - CVS
  - Emacs
  - GNU Compiler
  - Other: ______________________
- Languages
  - Perl
  - Python
  - Other: ______________________
- Web Application Development/Content Management (Zope, Plone, Midgard, eZ Publish, etc)
- Graphical user interface (GNOME, KDE, XFree86, etc)
- Security (Nessus, Nmap, Open SSH, Open SSL, Snort, etc)
- Research tools (Octave, R, etc)
13. How would you describe your software process model?

☐ Code and fix (writing code as long there is time and money, without a set of requirements)
☐ Waterfall (a sequential, document-driven methodology)
☐ V-model (extended waterfall model, adding details on the validation and verification phase)
☐ Spiral (development into small pieces)
☐ Evolutionary Prototyping (the model starts with an initial idea, which is then prototyped and released to the customer)
☐ Adaptive Software Development (iterative, risk- and mission-driven, component based, time-boxed process)
☐ Unified Process (risk- and use-driven, architecture-centric, iterative and incremental software development process)
☐ Extreme Programming (flexible, people- and result-oriented development process)
☐ Agile
☐ Other (please specify): ________________

14. How many selection processes do you perform by year?

____ processes.

15. How much time does it take to perform a selection and evaluation process?

____ days

16. Does your company use any formal process for selection and evaluation OTS components?

☐ No
☐ Yes
If yes, please answer the next question, else skip it.
If yes, which method and how did you find that method?
17. Which formal activities do you perform?

Mark one alternative per row

<table>
<thead>
<tr>
<th></th>
<th>Frequently</th>
<th>Often</th>
<th>Sometimes</th>
<th>Seldom</th>
<th>Never</th>
<th>Don’t know</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inspection of functional requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inspection of non-functional requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compare different candidates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

18. How do you find OTS components?

Please select one or several:

- Get it from a colleague
- Look at “generally recognized as mature” OSS programs (i.e. [http://www.dwheeler.com/gram.html](http://www.dwheeler.com/gram.html))
- Search on specialized sites (i.e. [http://freshmeat.net](http://freshmeat.net), [http://osswin.sourceforge.net](http://osswin.sourceforge.net))
- Search using specialized search engines (i.e. Google’s specialized searches for Linux)
- Other (please specify): ______________________

19. Are OTS components tested before integration?

- No
- Yes

If yes, please describe how? ____________________________________________________________

20. Have the selected OTS component ever been replaced with a different component at some point during the development process?

- No
- Yes

If yes, in which phase? ______________________________________________________________

21. Did it happen to realize late in the development process that your selected component is not suitable?
☐ No
☐ Yes
If yes, in which phase? ___________________________________________________

If yes, please answer the next two questions, else skip it.

22. Did you restart the process of selection and evaluation of an OTS component?

☐ No
☐ Yes
If yes, how much time did you spend? ______ days.

23. Did you start building an internal component instead of searching for an OTS component?

☐ No
☐ Yes
If yes, how much time did you spend? ______ days.

24. In which phase of the development process did you decide not to make the component yourself, but acquire an OTS component?

Mark one alternative per row

<table>
<thead>
<tr>
<th>Phase</th>
<th>Frequently</th>
<th>Often</th>
<th>Sometimes</th>
<th>Seldom</th>
<th>Never</th>
<th>Don’t know</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-study</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Requirements/Analysis</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall design</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Detailed design</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Coding</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Incrementally throughout the project</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Part 5. Activities after the selection of OTS components

25. Do you document the decision besides selecting and evaluating OTS components?

☐ Yes
☐ No
If yes, please specify how: ________________________________________________

If yes, do you reuse this information later?  ☐ Yes  ☐ No

26. Do you keep a local knowledge repository about the selected OTS components?

☐ Yes
☐ No
If yes, please describe shortly: ______________________________________________

If yes, do you reuse this repository later?  ☐ Yes  ☐ No

27. Do you have a person who is responsible for the OTS-component (e.g. a knowledge keeper)?

☐ No
☐ Yes, we have such a person in our company

Part 6. COTS-components

28. How many COTS-components...

were a part of the final product? ___
had the source code available? ___
29. If the source code was available did you...

<table>
<thead>
<tr>
<th></th>
<th>Frequently</th>
<th>Often</th>
<th>Sometimes</th>
<th>Seldom</th>
<th>Never</th>
<th>Don’t know</th>
</tr>
</thead>
<tbody>
<tr>
<td>read the source code?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>change the source code?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

30. Which categories would your COTS-components fit into?

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of components fitting to the category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Aided software engineering</td>
<td></td>
</tr>
<tr>
<td>Database Management</td>
<td></td>
</tr>
<tr>
<td>Knowledge Management</td>
<td></td>
</tr>
<tr>
<td>E-Learning</td>
<td></td>
</tr>
<tr>
<td>Human Machine Interface</td>
<td></td>
</tr>
<tr>
<td>Industrial Application</td>
<td></td>
</tr>
<tr>
<td>Internet</td>
<td></td>
</tr>
<tr>
<td>Middleware</td>
<td></td>
</tr>
<tr>
<td>Multimedia</td>
<td></td>
</tr>
<tr>
<td>Office/Business</td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td></td>
</tr>
<tr>
<td>Real Time/Embedded Systems</td>
<td></td>
</tr>
<tr>
<td>Scientific Engineering</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

31. Select how much you agree these statements apply to your project.
<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product was build around one or more central COTS-components.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The COTS-components mainly provided additional functionality which we did not have time to make ourselves.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The COTS-components mainly provided functionality which we did not have the expertise to make ourselves.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The COTS-components made it easier to build the final product.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COTS-components were domain specific.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COTS-components had more functionality than we needed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration cost exceeded the estimated cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The component did not work as expected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COTS-components were of high quality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COTS-components gave the end product high quality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
32. What is important when evaluating COTS-components?

<table>
<thead>
<tr>
<th></th>
<th>Very Important</th>
<th>Important</th>
<th>Average</th>
<th>Slightly Important</th>
<th>Not Important</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality of the component</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Familiarity with the component</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor is CMMI certified</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor reputation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ease of integration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

33. Did the use of COTS-components...

<table>
<thead>
<tr>
<th></th>
<th>Extremely Likely</th>
<th>Likely</th>
<th>Not Sure</th>
<th>Unlikely</th>
<th>Extremely Unlikely</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce development cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Improve quality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shorten time to marked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add additional complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part 7. About the company

1. What is the name of your company or business unit?

Company name: ________________________________________________

2. What is the type of company?

[Mark one alternative]
☐ Stand-alone: The highest reporting entity with no parent organization above it
☐ Subsidiary: An independent entity with majority interest held by a parent.

3. What is the ownership of your company?

[Mark one alternative]
☐ Publicly traded on a stock exchange
☐ Privately held company
☐ Government, education, or non-profit organization

4. What is the staff size of your mother company in your own country? (full- & part-time persons)______?

5. What is the staff size of your local business unit (full- & part-time persons)______?
[This staff size may be the same for small and medium sized companies]

6. What is the main business area of your company?
[Mark one alternative]
☐ IT/Telecom industry (Ericsson, Nokia, NERA, etc)
☐ Telecom service provider (Telenor, Telefonica, Orange, etc)
☐ Software House / Software Vendor (Opera, Microsoft, etc)
☐ Software / IT consulting company (Accenture, CapGemini E&Y, TiedoEnator, etc)
☐ Retail product with large degree of embedded software (Philips, Sony, etc)
☐ Other software-intensive company (please specify): ________________________________
Part 8. About the respondent

7. How old are you?
   □ <25
   □ 26-30
   □ 31-35
   □ 36-40
   □ 41-50
   □ 51-60
   □ >60

8. What is your current position in your business unit?
   □ IT Manager
   □ Project manager
   □ Software architect
   □ Software developer
   □ Web designer
   □ Tester/Quality assurance
   □ Other: ________________________________

9. How long have you been working in the current business unit?
   Years: __________________

10. How long have you been working with software development?
    Years: __________________

11. How long have you been working with OSS?
    Years: __________________

12. How long have you been working with COTS?
    Years: __________________

13. How many projects using OSS components have you been involved in?
    Number: __________________

14. How many projects using COTS components have you been involved in?
15. What is your highest completed education?

- Bachelor (BSc)
- Master (MSc)
- Ph.D.
- Other education: _____________________________

16. Is your degree software-related (informatics/computer science/telecommunications)?

[Mark one alternative]
- Yes
- No