Open Source Software Use and Development

- A Case Study regarding Business Perspectives and Tetra Pak Software Management

Master Thesis Report

<table>
<thead>
<tr>
<th>Document Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Name</strong></td>
</tr>
<tr>
<td><strong>Supervisors</strong></td>
</tr>
<tr>
<td><strong>Examiner:</strong></td>
</tr>
<tr>
<td><strong>Document Version</strong></td>
</tr>
<tr>
<td><strong>Authors</strong></td>
</tr>
</tbody>
</table>
Abstract

Open Source Software development has evolved from being an underground movement for hackers, to become a phenomenon interesting to commercial organisations. The possibility to control the code, avoid vendor lock-in and accomplish cost savings makes Open Source Software attractive. This Master Thesis includes a study of ten organisations relating to Open Source Software in different ways. The study resulted in analysis and documentation of certain gains and risks associated with Open Source Software use and development. To succeed, some prerequisites need to be fulfilled. A clear strategy is needed and a role concept must be defined in order to assign tasks and responsibilities. There must also be a strong advocate to support the project and support from management is crucial. In addition to this it is important to know how to cooperate with the open source community.

The Master Thesis work was performed at Tetra Pak R&D in Lund with the purpose of investigating if Tetra Pak can make use of Open Source Software in packaging and filling machines. Many other large organisations have been forced to adapt to Open Source Software. This suggests that Tetra Pak needs to stay updated with the evolution of this matter. Possible gains with using OSS are shorter development time, cost savings regarding licences and software better adapted to Tetra Pak needs. If Tetra Pak chooses to start using Open Source Software, improved procedures for OEM software management are required.
Acknowledgements

First of all, we would like to thank our great supervisors. Josef Nedstam at the Department of Communication Systems for his enthusiasm and useful advice. At Tetra Pak R&D, Mattias Wallinius, for introducing us to open source and Bertil Schaedel for encouraging our feminine side and arguing the importance of visual presentation. Our room-mate Maria Strömberg has also been of great support and has cheered us up with her positive attitude.

We would also like to thank all organisations that contributed to the result of this Master Thesis. Fredrik Norrman from Axis, Ruud de Bruijckere from Borland, Tord Andersson from Combitech Systems, Per Emanuelsson from Ericsson, Michael Pyschny from IBM, Greger Andersson and Joakim Nilsson from Nohau, Peter Neubauer from Scancoin and Åse Stiller from Teleca.

Finally, we would like to thank all the employees at Tetra Pak that have taken their time to answer our many questions.
# Table of contents

1 **Introduction** .................................................................................................................. 11
  1.1 Purpose and Delimitations of the Master Thesis ............................................................. 11
  1.2 Main Questions at Issue ................................................................................................. 12
  1.3 Outline ............................................................................................................................ 12

2 **Background** .................................................................................................................... 13
  2.1 Tetra Pak .......................................................................................................................... 13
  2.2 Research & Development at Tetra Pak .......................................................................... 14
  2.3 Software in Tetra Pak Machines ..................................................................................... 15
    2.3.1 Control System ......................................................................................................... 15
    2.3.2 Tetra Pak Operator Panel ......................................................................................... 16
  2.4 Challenges in Software Management ............................................................................... 16
    2.4.1 Mechanical Engineering and Automation ................................................................. 17
    2.4.2 Documentation and Knowledge Management .......................................................... 17
    2.4.3 Management and Organisational Structure .............................................................. 17
    2.4.4 Machine Product Portfolio ....................................................................................... 17
    2.4.5 Automation System and Suppliers ............................................................................ 18

3 **Method** ........................................................................................................................... 21
  3.1 Data Collection ................................................................................................................. 21
  3.2 Quality Assurance .......................................................................................................... 22

4 **Open Source Software, OSS** ............................................................................................ 23
  4.1 Free/Open Source Software Organisations ...................................................................... 23
    4.1.1 Free Software Foundation ....................................................................................... 23
    4.1.2 Open Source Initiative ............................................................................................. 24
    4.1.3 The Open Source Definition from OSI ..................................................................... 25
    4.1.4 Free Standards Group ............................................................................................... 26
  4.2 The Open Source Community ......................................................................................... 26
    4.2.1 Rules and Manner in the Community ....................................................................... 27
    4.2.2 Why Give Away Source Code? .................................................................................. 27
    4.2.3 Tools .......................................................................................................................... 29
  4.3 Security and Quality ......................................................................................................... 29

4.4 Successful Open Source Projects ...................................................................................... 30
  4.4.1 Linux ............................................................................................................................ 30
  4.4.2 Eclipse .......................................................................................................................... 31

4.5 Computer Software Licences .............................................................................................. 31
  4.5.1 GNU General Public License, GPL ........................................................................... 32
  4.5.2 GNU Lesser General Public License, LGPL ................................................................. 33
  4.5.3 Berkeley Software Distribution, BSD ........................................................................... 33
  4.5.4 MIT License ................................................................................................................ 34
  4.5.5 Apache Software License, ASL .................................................................................... 34
  4.5.6 Mozilla Public License, MPL ..................................................................................... 34
5 Commercial Use of OSS

5.1 OSS Applications in Business

5.1.1 Scancoin .......................................................... 37
5.1.2 Axis Communications .......................................... 38
5.1.3 Combitech Systems ............................................. 39
5.1.4 IBM ................................................................. 39
5.1.5 Ericsson ............................................................ 40
5.1.6 Nohau ............................................................... 40
5.1.7 MontaVista ....................................................... 41
5.1.8 Teleca ............................................................... 42
5.1.9 Borland ............................................................. 42
5.1.10 Beaumont Hospital ........................................... 43

5.2 Lessons Learned from OSS

5.2.1 Development .................................................. 44
5.2.2 Security & Quality ............................................ 44
5.2.3 Support ........................................................... 45
5.2.4 Procurement .................................................... 47
5.2.5 Licences & Legal Issues .................................... 47
5.2.6 Strategic & Organisational Aspects .................... 48

5.3 Relations to the Community .................................... 49

5.4 Business Models

5.4.1 Product-Related Business Models ....................... 51
5.4.2 Service-Related Business Models ....................... 51

6 Financial Aspects

6.1 A Cost Model – Total Cost of Ownership .................. 53

6.2 Cost Analysis of Software in Tetra Pak Operating Panel

6.2.1 InTouch & Windows .......................................... 55
6.2.2 Linux .............................................................. 58
6.2.3 Open Source Development Project Initiated by Tetra Pak 60
6.2.4 Cost Comparison .............................................. 63
6.2.5 Conclusions .................................................. 64

7 Analysis and Discussion

7.1 General Findings about Commercial Use of OSS

7.2 Risk Assessment for Tetra Pak

7.2.1 Development .................................................. 66
7.2.2 Security & Quality .......................................... 66
7.2.3 Support ........................................................... 66
7.2.4 Procurement .................................................... 66
7.2.5 Licences & Legal Issues .................................... 67
7.2.6 Strategic & Organisational Aspects .................... 67

7.3 Possible Gains for Tetra Pak

7.3.1 Development .................................................. 67
7.3.2 Security & Quality .......................................... 68
7.3.3 Support ........................................................... 68
7.3.4 Procurement .................................................... 68
7.3.5 Licences & Legal Issues .................................... 68
7.3.6 Strategic and Organisational Aspects ................ 68
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4</td>
<td>Is Tetra Pak Prepared for OSS?</td>
<td>69</td>
</tr>
<tr>
<td>7.5</td>
<td>Validity</td>
<td>70</td>
</tr>
<tr>
<td>7.6</td>
<td>Future of OSS…</td>
<td>71</td>
</tr>
<tr>
<td>8</td>
<td>Conclusions</td>
<td>73</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>A.</td>
<td>Appendix – Open Source Dictionary</td>
<td>79</td>
</tr>
<tr>
<td>B.</td>
<td>Appendix – Where to Start with OSS</td>
<td>83</td>
</tr>
<tr>
<td>C.</td>
<td>Appendix – Cost Drivers</td>
<td>85</td>
</tr>
</tbody>
</table>
1 Introduction

The concept of Open Source Software, further on referred to as OSS, is fairly new in the business environment. For OSS, the source code of the software is available, and this makes it possible for a software developer to understand how it is structured. The developer is permitted to make changes and to distribute the code if desirable. OSS is usually developed in a so-called developer community, where developers from all over the world are collaborating primarily through the Internet. The open source development process has been described by the open source author Steven Weber (2003) as:

“...collective action where one person puts in a chicken, someone else puts in carrots, another person puts in some onions...and everyone can take out delicious stew.”

For a long time OSS was not considered to be useful outside the hacker community, but today several organisations use OSS in a commercial context. One example is Red Hat, an organisation distributing Linux. Even larger enterprises such as IBM and Sun Microsystems are nowadays investing in OSS projects. Companies that are not primarily in the IT business also start to gain interest in OSS.

Tetra Pak is a large international corporation that provides its customers with packaging and processing solutions for food. In a big organisation like Tetra Pak it takes time to make changes, and therefore it is important to stay ahead of competitors when it comes to innovative technologies and new ideas. It is also important to have a cost effective organisation. This Master Thesis aims to investigate the possibilities of profiting from OSS mainly within Tetra Pak Research & Development (R&D). In order to find out if OSS is feasible to implement at R&D and how Tetra Pak could profit from it, the authors have studied literature in the area and have performed interviews with employees at Tetra Pak as well as with other companies where OSS has been introduced in various ways. Possible gains and risks have been identified from the perspective of these organisations, as well as the different ways they relate to the open source communities.

1.1 Purpose and Delimitations of the Master Thesis

The purpose of this Master Thesis is to investigate the conditions for commercial use of OSS in general and at Tetra Pak in particular.

The scope within Tetra Pak is the software development at the R&D department in Lund. The thesis covers mainly software management regarding OEM licences and not desktop environments, such as Microsoft Office and similar software. Since the examples of companies comparable to Tetra Pak in this context are very limited, the thesis concerns different kinds of organisations involved in OSS. The aim is to present findings as a knowledge base for continuous work at Tetra Pak but also to draw some general conclusions about OSS development and use. Implementation of any suggested solution is not within the scope of this Master Thesis.
The focus will be on licences certified by the organisation Open Source Initiative, which are described in Section 4.5, and not on other types of Free Software.

1.2 Main Questions at Issue
To establish the focus of the Master Thesis the authors have defined a set of main questions at issue.

1. Are there successful business models with OSS today? Does anyone use OSS for embedded or automation applications?
The aim is to look at other organisations using OSS, and learn from their experiences. What parallels can be drawn between them and Tetra Pak?

2. What gains can be made from OSS and what are the risks?
By looking at other organisations using OSS, potential gains and risks will be identified. A cost model is utilised to get an economical view of OSS gains and risks.

3. Can OSS be part of a long-term strategy?
Is the fast changing OSS community development possible to combine with Tetra Pak’s need for long-term thinking? To resolve this issue, Tetra Pak’s prerequisites and the risks of OSS management are investigated.

4. What are the legal implications and restrictions if OSS is to be sold as a part of Tetra Pak products?
The most common forms of licences are investigated. The way other organisations have resolved this matter will be considered.

5. How can Tetra Pak cooperate with OSS communities?
The rules and conditions of OSS communities are explored. The way other organisations have resolved this matter will be considered.

6. What prerequisites are needed to introduce OSS?
Experiences from other organisations are investigated. The prerequisites at Tetra Pak for OSS introduction are examined.

1.3 Outline
In Chapter 2 some facts about Tetra Pak are presented. The Master Thesis work procedure is detailed in Chapter 3. Chapter 4 introduces the concept of OSS and the free software/open source movement. Findings from case studies of a number of organisations are presented in Chapter 5. Chapter 6 discusses financial aspects concerning OSS use and development. An evaluation of gains and risks for Tetra Pak regarding OSS is carried out in Chapter 7. Chapter 8 summarises the conclusions. Appendix A is a small dictionary for software terminology, and Appendix B shows some useful sources for further reading. Finally, Appendix C presents cost drivers from the model presented in Chapter 6.
2 Background

This chapter contains a short presentation of Tetra Pak and of how the Research and Development department is organised. A description of software in Tetra Pak packaging and filling machines is included. Some challenges regarding software management are discussed. This provides the necessary facts to understand how open source can be seen from a Tetra Pak perspective.

2.1 Tetra Pak

Dr Ruben Rausing founded Tetra Pak in 1943 to develop a milk package that required a minimum of material (Tetra Pak, 2004). This was when the work with developing the famous tetrahedron-shaped carton package started. In 1951, Tetra Pak AB in Lund was established, and research and improvement regarding packaging and processing solutions for food continued. Today, Tetra Pak is a large international enterprise, which provides the world food industry in 165 different countries with a large variety of packages in different shapes and material. Tetra Pak also provides customers with systems for packaging, processing and distribution of liquid foods. One of the latest innovations is Tetra ReCart, the first package in the world that can withstand the high temperatures and pressure needed for food, which have traditionally been packed in cans or glass jars.

Tetra Pak has formulated a collection of core values, as seen in Figure 2.1, that are stated to function as guidelines for employees in the absence of detailed rules and regulations (Tetra Pak, 2004). The first one is “Freedom with Accountability” that expresses the employee’s freedom to take initiatives, as long as it is with Tetra Pak’s best interest in mind. “Partnership with Customers, Suppliers and Colleagues” communicates that Tetra Pak puts an effort in listening and learning from others, as well as acting proactively to deliver high quality and competitive advantage. The third core value, “Long-term Perspective” states that Tetra Pak focuses on learning and experimentation, and seeks to develop long-term commitments to customers and suppliers. “Innovation and Creativity” talks about setting challenging goals to strive for new and better solutions, and that Tetra Pak follows market trends and technical developments to gain the competitive edge. The last core value is named “Commitment and Fun” and says that by having fun at work it is easier to get good results, and that it is rewarding to achieve objectives.

Tetra Pak is organised in a number of different subsidiaries. Tetra Pak Carton Ambient, Tetra Pak Carton Chilled and Tetra Pak Processing System take care of the different packaging solutions. There is also the Market Operations subsidiary and supporting subsidiaries such as Human Resources and Finance Control.
2. Background

Figure 2.1 Tetra Pak Core Values.

“Long-term Perspective”

“Freedom with Accountability”

“Partnership with Customers, Suppliers and Colleagues”

“Innovation and Creativity”

“Commitment and Fun”

2.2 Research & Development at Tetra Pak

The R&D organisation consists of a number of departments, as seen in Figure 2.2. Research on new technologies is mainly performed at Tetra Pak Research and Development (R&D) AB. This subsidiary focuses on supporting the Business Areas of different packages in their ongoing projects and ensures continued innovation and development of packaging and processing technology. This is summarised in the R&D key accountabilities:

1. Support the Business Areas in delivering their new product plan
2. Keep Tetra Pak at the leading edge of chosen key technologies and competencies
3. Lead projects outside the remit of the Business Areas
4. Lead projects beyond the time horizon of the Business Areas

Figure 2.2 The Tetra Pak R&D organisation chart

There are four main departments. Food Safety & Quality, situated in Stuttgart, focus on meeting the increasing demands from customers and retailers. Material Technologies situated in Lund and in Romont in Switzerland, work with
packaging materials and new technologies to develop, convert and evaluate these. Package Development in Lund is responsible for developing new innovative packages. Finally Machine Technologies, which is also situated in Lund, work with the development of filling machines. The department also performs research on new technologies to improve cost, quality and capacity in the process. A supporting structure consists of Human Relations, Financial Management and Project Management.

2.3 Software in Tetra Pak Machines

Software developers at Tetra Pak are distributed all over the organisation. Within each subsidiary, one or several automation engineers are operating to ensure the function of automation for that specific area. The largest group of software developers working together is found within Machine Technologies at R&D. Another large group of software developers work in Plant Engineering, a subsidiary working with customized turn-key solutions, where a complete filling and packaging factory is constructed on site. Most of the machines Tetra Pak produces have a similar software structure, including an automation system and an operator panel.

2.3.1 Control System

The control system contains one or two Programmable Logic Controllers (PLC), which control multiple inputs and outputs with the help of a CPU. The PLC is run by a real-time operating system (RTOS), which is communicating with different device drivers for the hardware.

Tetra Pak R&D has designed a number of specific hardware components to satisfy special needs which cannot be fulfilled with standard solutions. The Tetra Pak Multipurpose Compact Controller (TMCC) is a self-sustaining microprocessor with its own memory and I/O units. The TMCC is a complement to the PLC, due to high time requirements on certain functions. A PLC can perform a process cycle in one millisecond at the best, while the TMCC can perform a cycle in the tenth of a millisecond. There is no specific function for a TMCC, instead the application program decides what the TMCC is to do. Tasks can for example be to keep track of the packaging design position to make sure the package is sealed in the right places, or to control the filling process to assure the right amount of liquid in each package. The program running on the TMCC is written in C, and involves some critical aspects such as real-time functions and close-to-hardware programming. The hardware supplier provides a compiler and a development environment. The Intelligent Power Module (IPM) is a part of the Temperature Control Impulse Sealing (TCIS). IPM is the control unit in the packaging sealing system and contains fewer I/O than the TMCC but in other aspects it contains about the same hardware. Both the TCIS and the TMCC communicates with the PLC through field buses and serial communication. Approximately two persons at R&D are responsible for developing components like the TMCC and the IPM.

About 50 developers are involved in PLC programming at Tetra Pak. The automation applications running on the PLC are usually written in one of the five programming languages within the international IEC 1131 standard for automation industry. These languages are: Sequential Function Chart (SFC), Instruction List (IL), Ladder Diagram (LD), Function Block Diagram (FBD) and Structured Text (ST) (Olsson and Rosen, 2003). All of the languages are
2. Background

intended for functional programming. At Tetra Pak, most of the PLC programming is conducted in Ladder Diagrams.

2.3.2 Tetra Pak Operator Panel

The control system communicates with the Tetra Pak Operator Panel (TPOP), shown in Figure 2.3, where a machine operator can check the status of the machine and make actions to correct a malfunction or modify performance if needed. The TPOP replaced the old MIMIC panel, where a board with diodes showed the information in a more primitive way. The TPOP is running on Windows, and a supplier provides the Supervisor Control and Data Acquisition (SCADA) application program called InTouch. All machines in serial production are delivered with InTouch. Besides InTouch there are some other applications running on the TPOP, for example the purchased programs SBTETRA and GENSP that handle communication between InTouch and the PLC. The in-house developed Packaging Line Monitoring System (PLMS) Logger produces logging files with runtime history, for example if any alarms have gone off. The PLMS Logger was from the beginning written in C++, but is now migrated to Visual Basic. Another in-house developed application is PLMS Send, which forwards information from the machine upwards to the office via serial communication or to Tetra Pak via modem.

In one of the current development projects a Tetra Pak specific GUI, written in Visual Basic and .NET, has been created instead of using the purchased software InTouch. This is part of a new initiative at Tetra Pak R&D, called Software House. The aim of this project is to make automation programs at Tetra Pak more structured. An important part of the project is a collection of libraries containing basic functions. According to the initiators there are several advantages with a planned software architecture. Among other things it is possible to define interfaces for device drivers. A lot of programming effort has been invested in this project and the initiators hope that the function libraries can be used in future projects.

2.4 Challenges in Software Management

Based on interviews conducted with people from different Tetra Pak departments, some software issues were identified. These issues are presented in the following sections. Most of them originate in Tetra Pak culture, organisational structure and products, but a few are related to suppliers and clients. A summary is presented in Table 1.
2.4.1 Mechanical Engineering and Automation
Traditionally, mechanical engineers have developed Tetra Pak’s machines. Today automation activities are an important part of the development, but people working within this area consider the interface between mechanical and automation development to be troublesome from time to time. It is expressed that automation is involved too late in projects and programmers sometimes have to work overtime to be able to complete their tasks in time. In addition to this there is no standard for software architecture or a defined software process within Tetra Pak. Due to the lack of standards and processes, programmers can claim “artistic freedom”, something that is not appropriate for producing coherent, easy-to-understand software.

2.4.2 Documentation and Knowledge Management
According to some Tetra Pak R&D employees, documentation regarding software can be improved. Programs developed in-house sometimes lack documentation and this can cause problems when a developer decides to leave Tetra Pak. There are examples of occasions when this has happened. When there is no documentation it is difficult to handle changes and debugging since it is hard to understand the structure of the program for someone other than the original creator. This way, Tetra Pak risks losing valuable knowledge that could have been preserved with improved documentation. There might also be a risk of losing knowledge when hiring consultants to do certain tasks. Another difficulty is that changes, made while developing, are sometimes only saved in a handwritten document. Occasionally there are different types of documentation, not containing the same information, for a certain part or program.

2.4.3 Management and Organisational Structure
The Tetra Pak organisational structure is seen as informal, which sometimes makes it difficult to understand how it works. Software development and procurement are activities present within different departments at Tetra Pak. Developers in different subsidiaries are generally not coordinated. Software vendors working with Tetra Pak would prefer to be in contact with only one person, but this is not possible since there is no organisational structure for comprehensive software management within Tetra Pak. This might be a disadvantage when analysing needs and when trying to keep costs low. On a related note, one person involved in development believes risk analysis is not a priority within Tetra Pak. When it comes to project management, some employees think that the structure of a certain project depends on the project leader. It is expressed that it would be useful to establish a method for coordinating groups.

2.4.4 Machine Product Portfolio
Tetra Pak has a wide range of different products with long lifecycles. Once a product is developed, changes are undesirable since serial production is applied. Software, provided for a certain type of machine, is not upgraded when the supplier releases new versions. There is usually no need for new functionality and it would be expensive to integrate the new program and to provide support for different software versions to the same type of machine. Therefore Tetra Pak tries to hold on to the originally installed version of a program as long as possible. Software providers are generally not adapted to this kind of demands. Most of their clients are willing to upgrade to have bugs corrected and to get
better functionality. To have a few clients that stick to an old version, while most clients upgrade is expensive for the supplier. For this reason, it is sometimes difficult for Tetra Pak to get support or even to get licences for old versions of software.

Once a machine is in production at a client’s plant, service technicians handle problems that arise. When developing machines, it is therefore important to understand the service technician’s problems with maintaining the machine. During interviews it was expressed that this ability needs to be improved. It is important that a service technician can make required changes without corrupting the code in the machines. It is also a delicate matter to control and to register changes made by service technicians. These changes are often just written by hand in the file containing the machine’s documentation.

2.4.5 Automation System and Suppliers

Large automation suppliers provide PLC systems for automation in Tetra Pak’s machines. The systems contain both hardware and software, including development environments. These are specific for each supplier, and once Tetra Pak has adapted to one system, there is a risk of vendor lock-in. Learning a new system and adapting machine technology demands excessive work, and generates large costs. The programming languages used for automation at Tetra Pak all conform to the standard IEC 1131, since this is what the suppliers use in their systems. The classical automation languages are function-oriented, which result in large, complex programs with many lines of code. Due to the large code base, programs are often difficult to maintain when it comes to debugging and updating. It has been suggested by certain Tetra Pak employees that by using object-oriented programming languages instead, these problems might be eliminated.
Table 1. Challenges for Tetra Pak Software Management

<table>
<thead>
<tr>
<th>Area</th>
<th>Challenges</th>
</tr>
</thead>
</table>
| **Mechanical Engineering and Automation** | • PPC/Automation involved late in the projects.  
• Programmers have to work extra hours.  
• Difficulties in interface mechanics/automation.  
• No standard for software architecture.  
• Developers want “artistic freedom”.  |
| **Documentation and Knowledge Management** | • Need to improve Configuration Management and documentation.  
• Risk losing knowledge when hiring consultants.  
• Difficulties when a developer leaves Tetra Pak.  |
| **Organisational Structure** | • It depends on the project leader if a project is structured or not.  
• Need method to control groups.  
• Difficult to understand the informal organisation.  
• Software vendors would like to have one contact  
• Risk analysis is not a priority.  |
| **Machine Product Portfolio** | • How can code be validated and protected when service technicians need to correct errors?  
• Need for better understanding of the service technician’s maintenance problems.  
• Difficult to introduce new machine models.  
• Problems with getting support for old versions.  |
| **Automation System and Suppliers** | • No object oriented programming language for automation.  
• Vendor lock-in. Systems for automation are specific to a certain supplier.  |
3 Method

The following chapter covers the way this Master Thesis work was performed and the research and data collection methods involved, with some discussion regarding quality.

Since the phenomenon of OSS in a commercial context is fairly new, there is not much research performed on the subject. The authors have therefore chosen a qualitative approach to this Master Thesis. The qualitative school is inductive, which means that it is used to build theory by drawing general conclusions from single case studies (Nedstam, 2004). It is assumed that everything depends on its context, and it is impossible to lift the phenomenon out of this context to perform an objective experiment.

This Master Thesis was partly conducted as a multiple case study, where one case was to study the prerequisites for introducing OSS at Tetra Pak, and also to investigate current software costs. The other case studies were based on a number of organisations already using open source to some extent.

3.1 Data Collection

It is important to know that a qualitative study can never be statistically significant (Nedstam, 2004). The sample is not selected to present a larger population; instead subjects are sampled with the purpose of finding the persons most likely to provide useful information about the phenomenon. Since this Master Thesis was focused on a qualitative approach, conclusions were drawn mainly from interview results. Interviewing people is a flexible way to gather information (Robson, 1993). The large advantage is that the questioner has an ability to modify questions continuously during the interview, and easily investigate further if an interesting area comes up. A disadvantage is the amount of time needed to carry out interview sessions and review the material to find a way of reaching scientific results.

To attain a broad picture of Tetra Pak’s organisation and of the way people work at R&D, a number of semi-structured interviews were carried out. Interviews were performed at Tetra Pak Packaging Process Control (PPC), Tetra Pak Machine Design, and Tetra Pak Carton Ambient. The questions concerned work routines and focused on software development management. All interviewees were given the opportunity to validate a written summary of the interview.

Furthermore, work included a different set of semi-structured interviews with the purpose of investigating open source from the starting point of the information gathered at Tetra Pak. These interviews were essentially based on the Main Questions at Issue mentioned in the introduction chapter. The primary purpose was to collect experience regarding work with OSS in a commercial context from external sources, such as academics and organisations involved in OSS today. The organisations were chosen through snowball or chain sampling (Patton 1990). When using this method people within certain areas are asked to recommend other persons that would be valuable sources of information. For every interview, new recommendations of other people are asked for, and the accumulated material grows, like a snowball. After a certain time the study will converge when the same recommendations occur. When applying this method to
people with open source knowledge in Swedish companies, the names of recommended persons were soon repeated. One reason for this might be that open source is not widely spread in Swedish organisations.

A third interview round was performed with structured open-ended questions derived from a cost model based on Total Cost of Ownership, and this involved a handful of people involved in software management at Tetra Pak. In addition to interviews, studies of written material were also a part of the data collection. In this Master Thesis some internal documents from Tetra Pak were studied in addition to the references to provide facts and figures about the company.

3.2 Quality Assurance

To assure quality in the project, the Master Thesis work procedure contained a number of recurrent processes. The authors had weekly meetings with the mentors at Tetra Pak as well as with the mentor at Lund Institute of Technology. The report was thoroughly reviewed every second week. A daily diary was kept to enable the authors to evaluate actions and to act as a resource for traceability of events.
4 Open Source Software, OSS

To investigate possible ways of using open source at Tetra Pak it is necessary to know what OSS means and to understand how the open source movement has evolved. This chapter includes a presentation of some important open source interest organisations such as the political Free Software Foundation and the more business friendly Open Source Initiative. Some theories of how an open source project works are also presented. Finally some legal aspects regarding different kinds of open source licensing are treated.

To understand the concept of OSS it is necessary to know the meaning of source code. Source code is basically any series of statements written in some computer programming language before being processed to be understood by the computer. In modern programming languages the source code, which constitutes a program, is usually represented in several text files. OSS gives access to the source code of a program, and the principles of the program are revealed. To have the source code also means that it is possible to make changes in it and thereby change the function of the program.

In the beginning of software development, software was used mainly by scientists in an academic environment, and was freely shared among developers. Low-level programming was difficult and time consuming. Once a program worked it was shared to everyone who needed it. When computer technology evolved, applications for business started to appear. The need for business solutions created a market for software companies and proprietary software was born. These companies protected the source code of their programs and sold only a compiled version. However source code was still shared among academics and enthusiasts.

4.1 Free/Open Source Software Organisations

Some of the more famous contributors in the open source movement history are presented in the following sections.

4.1.1 Free Software Foundation

The Free Software Foundation, FSF, was founded in 1986 and claims to promote the computer users’ right to use, study, copy, modify, and redistribute computer programs (FSF, 2004). The founder Richard Stallman used to work at the Artificial Intelligence Laboratory at MIT. He was convinced that software should be free and resigned from MIT to support the development of free software. The word free is not intended to mean free of charge, but refers to free in the sense that anyone should have a certain freedom to, for example, change or redistribute software.

The term “Free Software” was officially introduced in the GNU manifesto, which was written by Stallman in 1985 to ask for participation and support in the GNU-project. GNU stands for GNU’s Not Unix and the project aims to develop a complete non-proprietary operating system. FSF also promotes the development and use of Free Software and documentation in general and works to spread awareness of the ethical and political issues of freedom in the use of software. This is the somewhat political side of FSF, for which the organisation is sometimes considered to be anti-commercial or even hostile towards software.
business. On the GNU website it is stated that the development of the GNU-operating system aims to eliminate the need to use proprietary software. However, FSF does not, in principle, question charging for software development. Most FSF funds do in fact come from its distribution of CD-ROM’s and manuals for GNU-software.

4.1.2 Open Source Initiative

While some people involved in Free Software are of the opinion that conventional software business has nothing to do with Free Software, others want to promote the advantages of software with open source code to commercial organisations. In 1998 the term “Open Source” was coined, to describe basically the same thing as the Free Software label used by FSF. The purpose of introducing the concept of Open Source Software was to create a sort of trademark (OSI, 2004). This could be used to encourage commercial organisations to open their eyes to the development work within the hacker community and the advantages of an OSS licence. Several front figures in the hacker world supported this new initiative. One of the initiators was Eric Raymond. He is, among other things, known for the article “The Cathedral and the Bazaar” (Raymond, 1998) where he compared traditional software development strategies to development performed by the hacker community. Another initiator was Bruce Perens, a former Debian/GNU Linux project leader, who is the primary author of the Open Source Definition, which is presented on the next page. Linus Torvalds, the father of the Linux kernel, also expressed his support for the project.

The Open Source Initiative, OSI, is the organisation that stands behind the OSS concept. OSI is a not-for-profit corporation, based in California, US. Its main concern is to administrate the Open Source Definition, which is derived from the Debian Free Software Guidelines. Debian is an open source project organising the work on the free Debian operating system, based on the Linux kernel. The Debian guidelines were revised and some Debian specific references were removed to produce the Open Source Definition document.
4.1.3 The Open Source Definition from OSI

The following nine principles form the definition of open source according to OSI. This organisation certifies a large number of licences that conform to the Open Source Definition. Software with one of these licences is said to be OSI Certified. This is a registered certification mark. OSI certified software might also carry the graphic symbol of OSI, as seen in Figure 4.1.

1. Free Redistribution
OSI Certified software may be redistributed in any way without any claims from the original distributor.

2. Source Code
The software must be distributed as source code or as a compiled program with access to source code.

3. Derived Works
Modified versions of the software are allowed and can be redistributed on the same terms as the original version.

4. Integrity of The Author’s Source Code
If the author wishes that the original source code should be visible when redistributed after modifications, it is possible to only allow modification with patch files\(^1\) that change the code at build time. Another alternative is to request that a new version must change name or version number.

5. No Discrimination Against Persons or Groups
It is not allowed to discriminate any persons or groups in the OSI licence. It is for example not possible to state that members of a certain organisation are forbidden to use the software.

6. No Discrimination Against Fields of Endeavour
The software cannot be restricted from any field of application. For example, it is not allowed to forbid someone to use OSI certified software in a weapon system.

7. Distribution of License
When software is redistributed, the new user must have all the rights originally attached to the software.

8. License Must Not Be Specific to a Product
The rights attached to the software must not be tied to a certain distribution. It is for example not possible to state that someone who uses an OSI certified program only is allowed to do this in a Red Hat Linux distribution.

9. License Must Not Contaminate Other Software
The software licence must not imply any restrictions regarding other software that might be distributed together with the software.

\(^1\) A patch is a software update meant to fix problems with a computer program. This can range from fixing bugs to replacing graphics to improving the usability or performance of a previous version (Wikipedia 2004).
4. Open Source Software

4.1.4 Free Standards Group

The Free Standards Group is an organisation working to grow the use and acceptance of OSS by developing and promoting standards, mainly for Linux. Both open source development communities and the IT industry are engaged in the organisation. Standards produced by the Free Standards Group are available to anyone and are published under open source licences. Organisations involved in the Free Standards Group include Caldera, Conectiva, Debian, Dell, Hewlett Packard, Hitachi, IBM, Miracle Linux, The Open Group, Oracle, Red Hat, Sun, SuSE and TurboLinux (Free Standards Group, 2004).

4.2 The Open Source Community

According to Feller and Fitzgerald (2002), the Open Source Software process involves:

“...large, globally distributed communities of developers collaborating primarily through the Internet.”

The development process in these communities is usually characterised by parallel work, independent peer review, prompt feedback, highly motivated and talented developers, increased user involvement and rapid release schedules (Feller and Fitzgerald, 2002).

The first well-known article studying the phenomenon of communities developing software is The Cathedral and the Bazaar (1997) written by Eric Raymond. He argues that traditional software development is like building a cathedral with experts and strict rules while development in communities is more like an oriental bazaar where everyone communicates and exchanges ideas and material without any strict rules. Raymond was surprised that this chaotic work approach resulted in stable, fast evolving software. Among other things he refers to Linus Torvalds’s law, “given enough eyeballs, all bugs are shallow” as one of the reasons why the communities were successful in their development. When thinking the cathedral way bugs are assumed to be difficult to find and a few experts are assigned this task. In the bazaar, both users and developers might search for bugs.

Open source communities do not always correspond to the picture described by Raymond. They might differ much in size as well as in activity. Large open source projects like the development of the Linux kernel or the Apache web server are likely to profit from abundant peer review and feedback. They can also easily attract skilled developers. However this is not necessarily true for all open source communities. Krishnamurthy (2002) presents a study of 100 mature OSS projects found on Source Forge, which is a large repository of OSS programs. The aim of the study was to see if these projects conform to the bazaar model by investigating the number of developers engaged in each project. One of the conclusions was that there are few developers in most of the projects. The mean value was 6.61 developers per project. Another surprising result was that most projects did not generate much discussion. This contradicts the general stereotype of an open source project as a babbling bazaar. Krishnamurthy does not claim his findings to be general for all open source projects but his study does show that a community might take different forms. It is not possible do
draw conclusions about open source projects in general by studying for example the Linux kernel development.

4.2.1 Rules and Manner in the Community

Even though open source communities can take different forms there are a few informal, social rules that are generally accepted. These rules often serve to avoid chaos when developers are spread all over the world and never meet face-to-face. Feller and Fitzgerald (2002) describe some of these rules. It is for example important to avoid forking of a project. Forking means to split the project so that developers work in different directions. If this happens, there is not a clear statement on where the project is going and developers will soon lose interest. Another important rule is to not take credit for someone else’s work. Acknowledgement is a key motivation factor for developers in open source projects and therefore it is important that everyone is recognised for what he has done.

It is also of essence that leaders of open source projects are humble. If the original developer gives the impression that he himself can solve all problems, it is not likely that he will be able to attract other developers to work on the project. Anyone who likes to participate in an open source community must also respect that there is a low threshold for noise. Questions that are irrelevant or that has already been answered in the newsgroup must not be asked since this means wasting the time of others. It is not common that a formal development process is used in open source projects. However it is necessary to handle version and release management to avoid chaos. For this purpose a set of OSS tools are used, as will be presented in Section 4.2.3.

The way tasks are divided in an open source community differs from how this matter is solved in commercial organisations, where a manager usually assigns a task to a certain developer. In open source communities a task is either spontaneously picked by a developer or chosen in mutual agreement during discussion in the community (Siefert and Wieland, 2003). Either way, the task is chosen voluntarily and no one can force another developer in a community to perform a particular task. Therefore motivation is essential in OSS development and it is very important to understand the factors that inspire developers to contribute to open source, or more simply put; why give away source code?

4.2.2 Why Give Away Source Code?

For people who are not committed to an open source project it may be difficult to understand why someone would freely give away code that he or she has put a significant effort into. Bergquist and Ljungberg (2001) discuss different types of motivation in the so-called gift economy. The giving away of software is compared to gifts in primitive cultures and also to the academic way of sharing knowledge.

In primitive cultures gifts contribute in building social relationships between people and is a way to maintain power structures. A gift in these cultures can be seen as a process following a set of rules, which is a symbolic translation of the social structure in the society. This can to some extent also apply to the open source community. Even though no object changes owner when the source code is released, certain social relations are created in the process. Some developers give away information to indirectly gain information; others seek emotional
support and a sense of belonging to a group. In the giving process of the open source community, the rules of the society are revealed. If someone is breaking the rules, he or she will soon know this by reading the comments afterwards in the discussion forum.

Parallels can also be drawn between academic practices and the functions of an open source community. In the academic world it is necessary to publish findings, to achieve recognition and advance a career. No economic compensation is provided for articles submitted to scientific journals. The community works in about the same way regarding source code. Another thing communities have in common with the academic world is the characteristics of peer review. Before publishing an academic article it is reviewed by other scientists working in the same area and it is not published unless it contains relevant scientific results. In the open source community, it is customary to earn respect and appreciation by providing high quality code. When code is released to the community, other developers scrutinize it and inadequate code is cruelly rejected. Once the software is released, many users act as beta testers and report the bugs they find. Motivation to provide feedback is high, since users know that the bugs they report are likely to be corrected rapidly.

Another important motivation factor for OSS developers originates from the fact that work is done on a voluntary basis. The consulting firm Boston Consulting Group conducted a survey in 2002 regarding the motivation and background of participants in OSS projects. Out of 684 respondents, 58% were working as IT-professionals, as seen in Figure 4.2 For some people it might be surprising to see that such a large part of OSS developers are IT-professionals. Why work in your spare time with something that you get paid for at work? When a manager in a commercial organisation suggests tasks that have to be done, many of them are considered to be tedious. In an open source project each developer is free to choose what tasks to complete. This gives an opportunity to do challenging work and to gain knowledge in a specific area. The lack of timetables in open source projects also gives a possibility to develop ideas that cannot be explored at work because of time and budget constraints.

Figure 4.2. Current occupation of open source developers in the Boston Consulting Group hacker survey.
If developers are motivated to participate in OSS projects by challenging tasks, then who does the important, but somewhat tedious, work like debugging or writing documentation? According to Raymond (1998) this type of work “...is more praiseworthy than cherry-picking the fun and easy hacks.”

This means that performing dull work is more appreciated by other community members. In the cases where the community’s praise and glory is not enough to get all the necessary work done, a hybrid model is proposed by Bonaccorsi and Rossi (2002). Commercial organisations can gain revenues by supporting users of OSS, with the ‘non-sexy work’ such as packaging, consultancy, maintenance, updating and training.

4.2.3 Tools

When administrating an open source project it is necessary to keep track of different versions and releases. It is important to be able to handle software requirements if tasks are to be divided, and also to manage user relations to take advantage of the user base for feedback. A survey conducted by Siefert and Wieland (2003) investigates which tools are used in 15 large OSS projects, such as Debian, Python, Zope and MySQL. The results showed that tools for design were rare in open source development. Developers in OSS communities are often of the opinion that “code rules” which means that they prefer to express their ideas directly in terms of code rather than using modelling techniques. The most common tools found in the survey are presented in Table 2 below.

<table>
<thead>
<tr>
<th>Area of usage</th>
<th>Tools used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Text editor, Wiki</td>
</tr>
<tr>
<td>Use case collection</td>
<td>Text editor, Wiki (if at all)</td>
</tr>
<tr>
<td>Design</td>
<td>Umbrello, mailing list</td>
</tr>
<tr>
<td>Configuration</td>
<td>Automake, autoconf, ZConfig</td>
</tr>
<tr>
<td>management</td>
<td>CVS, mailing list</td>
</tr>
<tr>
<td>Testing</td>
<td>JUnit, Bugzilla</td>
</tr>
<tr>
<td>Packing</td>
<td>Zip, ant, tar, RPM</td>
</tr>
<tr>
<td>CRM</td>
<td>Bugzilla, Usenet, mailing list</td>
</tr>
<tr>
<td>Bug tracing</td>
<td>Bugzilla, PHPBBugtracker</td>
</tr>
<tr>
<td>Project management</td>
<td>Web, mailing list</td>
</tr>
</tbody>
</table>

One of the most important tools in an open source project is the mailing list. As seen above, mailing lists are used both for project management and for configuration management. A common definition of a mailing list is a special usage of email that allows for widespread of information to a large group of Internet users (Wikipedia 2004). In open source projects the mailing list takes the form of a discussion list, where a subscriber can use the list to send messages to all the other subscribers, who may answer in the same way.

4.3 Security and Quality

There are different opinions on whether OSS is more or less secure and reliable compared to proprietary software. According to OSI, closed source does not protect against hacker attacks, since it creates a false sense of security. This
makes it more difficult to find holes and fix them and it is also harder to
distribute trustworthy fixes when a hole is revealed. In short this is expressed:

“Security through obscurity does not work”

Those who promote closed source argue that only trusted insiders can examine
security vulnerabilities in proprietary software and therefore holes are less likely
to be found by hackers. Siefert and Wieland (2003) argue that open source
projects have a better quality control since developers do not have write access to
the repository. Contributions from different developers are always evaluated
before being integrated in the software. David Wheeler claims that the design of
OSS generally makes it harder to spread viruses since they do not support start-
up macros or execution of mail attachments that can be controlled by hackers.
Many OSS also have write protection on system directories. (Wheeler, 2004)

Paul Vixie writes about open source development and software engineering in
Voices from the Open Source Revolution (Vixie, 1999) He concludes that open
source developers prefer to implement software, rather than defining
requirements or testing. He believes that open source quality assurance is
unorganised, but that extensive field-testing helps to improve quality. Another
factor affecting the safety of software is that commercial software, and
Microsoft’s products in particular, are more often targets of the hackers who
create viruses. Software that is exposed to more virus attacks has to have a
higher security standard to be as safe as other software. This is also something to
consider when evaluating open source and proprietary alternatives.

4.4 Successful Open Source Projects

In the sections below, two very successful open source projects are presented.
The first one is Linux, a classic example of open source development that started
as early as 1991, initiated by Linus Torvalds. Eclipse is a rather young project,
initiated by IBM in 2001 and is run by a group of large organisations.

4.4.1 Linux

The kernel and operating system Linux is one of the most famous open source
projects and goes back to 1991 when the university student Linus Torvalds from
Finland first released his code in the open to get opinions and suggestions. The
first objective with Linux was to create a more advanced replacement of the
small Minix system, and started out like a hobby (Wikipedia, 2004). More and
more developers and users got interested and the Linux project evolved rapidly.
Today it is estimated that more than 40 000 developers have participated in the
development. Linux is closely coupled to (but not a part of) the GNU project, a
project aimed at developing a free Unix-like system. When the Linux kernel was
released in –91, GNU had developed all components except a kernel, and Linus
and other Linux developers adapted the kernel to work with the GNU
components to create a fully functional operating system, and the Linux
operating system is still referred to, by some, as the GNU/Linux system. The
Linux kernel is released under the General Public License, GPL, discussed in
Section 4.5.1.
4.4.2 Eclipse

IBM started developing the Integrated Development Environment (IDE) Eclipse in 1999 and 40 developers initially staffed the project. In 2001 IBM released the code as open source and the Eclipse consortium was founded by a number of large organisations\(^2\) to provide an open platform for tool integration (Eclipse.org, 2004). Today 13 organisations are members of the Board of Directors, including Ericsson, Hewlett Packard, Intel and SAP to mention some. In 2004 IBM announced Eclipse to be an independent non-profit organisation.

Today, the Eclipse foundation has a CEO and five employees. Different councils give proposals in the areas of requirements, planning and architecture, and the Board of Directors decide which ones to carry on with. About 10 large independent projects are in progress within Eclipse at the moment, involving between 10 and 100 developers each. Most of the member organisations contribute with around five developers. IBM, as the original developer of Eclipse, has so far contributed a little bit more than the other members, but this is about to change. Often members simply contribute to the projects they are interested in, but a certain amount of cooperation is also needed to fit different parts together.

4.5 Computer Software Licences

In conventional software business, programmers are paid to develop software. Once the software is developed, the cost to reproduce it is very low. To finance development and to make a profit, a software licence is sold to the user. This usually means that the licence holder is authorised to use the program, without being the actual owner of it. The licence generally gives a non-exclusive right to install the program on a limited number of computers (Olofsson, 2003). The Business Software Alliance (BSA) was established in 1988 as a voice for software organisations to make sure that software licences agreements are followed, and to inhibit copyright breach (BSA 2004).

To end users, software is sold with a standard licence. The End User Licence Agreement, or EULA, is what the user usually sees before clicking the "I agree"-button when installing a program. The text informs the user about what is allowed and not allowed to do with the program, according to the manufacturer. It also releases the manufacturer from a majority of responsibilities regarding the program. Companies sometimes have the possibility to negotiate about their licences. Since many software companies originate from the United States, licences are often based on American law. When any condition in a licence agreement disagrees with Swedish law it might be adjusted by Swedish court.

Licences for Free Software and OSS have one thing in common with licences for proprietary software; they all use the author’s right to make any agreement whatsoever regarding the economic right of disposal of the program. While most software development organisations use this right to inhibit spreading of the program and to ensure that no one else can sell it, Free Software and OSS does somewhat the opposite. Instead of limiting the user’s right to the program these

licences increase the user’s right of disposal at the author’s expense. Open source can be used as a reversed patent since no one can obtain a patent for a program once it is released in public.

A legal issue concerning OSS is the risk of patent infringement, which means that someone claims to have registered a patent for some part of the code, included in the OSS. It is easier to get access to the code of an open source program than a proprietary program and therefore the open source code is likely to be examined by a larger number of people. If anyone finds a piece of code for which a patent is registered, there is risk for legal consequences. To be held guilty of patent infringement according to Swedish law, there should have been intent or serious negligence. If someone has used software that is a registered patent, without knowing, it is likely that the user must pay compensation for using the software if this is considered to be reasonable. The law covering this issue can be found in the Swedish law (9 kap. 58§ patentlagen). According to Golden (2004) there have been very few instances of patent infringement during the entire history of software and this can therefore be seen as a quite theoretical risk.

There are a large number of different licences for Free Software where some are OSI Certified and others are not, as seen in Table 3. Other types of alternative software distributions are freeware and shareware, but these are not treated in this Master Thesis. Even between different OSI Certified licences there are great differences. One might say that the OSD is the lowest common denominator for these licences. Some licences are very free in the sense that they allow the user to do practically anything with the program, such as the MIT License (Walleij, 2003). Others are more restrictive and impose certain requirements on the user, such as the GNU GPL, which require that any software derived from GPL software, should be licensed under GPL. To illuminate different aspects of these licences, the most commonly used are described and commented in the following sections. The source for facts about licences is a report written by Jessica Olofsson (2003), if no other source is mentioned. The descriptions of the licences in the following sections are just short summaries. To see the licences in full text, visit for example the OSI home page, www.opensource.org.

**4.5.1 GNU General Public License, GPL**

GPL is the most commonly used OSS licence, and is applied to approximately 70% of all OSS projects (Walleij, 2004) It has been formulated and revised by American law professors, and is considered to be the most well written licence regarding legal aspects. An important condition in this licence is its so-called viral properties. This means that all software containing, using or linking to software licensed by GPL must also be GPL. There are no restrictions against licensing the software according to other terms as well. For example the Swedish company MySQL AB distributes its programs both with a proprietary licence and with GPL, depending on the customer demands.

The GPL allows modifications of the software to create a new version based on the original software, and also to distribute this new version. The source code must be included in these distributions. GPL defines source code as all modules, interface files, definitions and script files for compilation and installation of the software. GPL gives you the right to sell GPL software, and also allows selling a guarantee for the software. It is possible to store a mix of proprietary software
and GPL software on the same hard disc drive or CD, but it is forbidden to combine non-GPL and GPL software parts in the same program.

It is allowed to modify a program internally, to be used within a company for example, but if the modified program is distributed the GPL takes effect. If any of the terms in GPL are violated, the user of the GPL software no longer has authorization to utilize the software. The software creator takes no responsibility for errors in the software, besides ordinary guarantees according to law. According to Olofsson, the GPL should be possible to apply within Swedish law.

4.5.2 GNU Lesser General Public License, LGPL

The LGPL was originally created to enable program libraries to be used by all sorts of software. One example is the libraries for the programming language C, which are licensed by LGPL. The terms in LGPL are similar to the GPL, but differ in a few areas. A distinction between “work based on the library” and “work using the library” is made. LGPL is the lighter version of GPL, and makes it possible for proprietary software development to use LGPL libraries without applying LGPL to the software.

Table 3. An Overview of software licences.

<table>
<thead>
<tr>
<th>Licences</th>
<th>GPL</th>
<th>LGPL</th>
<th>BSD</th>
<th>MIT</th>
<th>ASL</th>
<th>MPL</th>
<th>EULA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source code freely available</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Derived work must be free again</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linking with proprietary software is allowed</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software may be distributed without source code</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No software usage restrictions</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Source code modification allowed</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Software may be sold with a commercial purpose</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>New users must always have access to the source code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

4.5.3 Berkeley Software Distribution, BSD

The BSD License is a template, which allows anyone to create its own licence based on BSD. The conditions in the BSD License differ from both more restrictive OSI licences and from proprietary licences. The licence is simple and easy to understand, with few constraints. BSD is closely related to the MIT License and the Apache License, as described further on in the chapter. The most limiting part of the BSD License is the fact that the program is without
guarantee. If BSD software is malfunctioning or causing damage, the authors cannot under any condition be held responsible. Another condition is that the licence itself must be included when the program is redistributed. In case the program is revised it is no longer mandatory to include the original licence.

A reason for the liberal character of the BSD License is the fact that the development has been financed through the Berkley University, by American taxpayers. Therefore the result should be available to everyone. This is a fact that several commercial interests have profited from. BSD code has for example been used by both Microsoft and Apple, as a part of their products. In older BSD-licences it is compulsory to mention the authors of the program when marketing a product containing BSD licensed code. This requirement has been removed in more recent BSD Licenses, but there is still much software based on the old licences. Some advocates of Free Software believe that the BSD-licence is too free. According to them, developers will not be interested in contributing to projects if they know that companies can use their code without giving anything back to the community.

### 4.5.4 MIT License
Massachusetts Institute of Technology (MIT) has been using different licences through the years. The current one is also referred to as the X License. It is a very permitting licence. Just like the BSD License, it allows anyone to create a licence based on the MIT License. This licence also has much in common with BSD when it comes to the conditions. Anyone can use a program with this licence and it is also possible to change the program or to use it in other software. The only regulation is that the program should be provided with the licence and that the author cannot be held responsible for errors in the software or the implications of using the software.

### 4.5.5 Apache Software License, ASL
The ASL is based on the BSD License. The Apache web server and other projects in the Apache organisation are covered by this licence. It is a free licence and does not require much of the user. However, one requirement for redistributing Apache licensed software is that the text stated below must be included, either in the documentation or presented in an appropriate way within the program:

> “This product includes software developed by the Apache Software Foundation (http://www.apache.org)”

### 4.5.6 Mozilla Public License, MPL
When Netscape released the source code for the web browser Netscape, the Netscape Public License, NPL was created. However, the NPL was criticised for having conditions regarding exclusive rights to the source code for Netscape, and this was the reason for creating the MPL instead. OSI assisted Netscape in formulating the licence according to the Open Source Definition.

MPL is an extensive document, containing eight pages of text, which starts with 14 definitions of different terms. It is differentiated between the “original developer” and the “contributor” which only contributes with modifications and added parts. The licence agreement includes one part concerning software
4. Open Source Software

developed by the original developer, which grants the right to use the original software, and one part regarding the contributors’ modifications. MPL gives the licensee a right to use, modify and distribute the software. If the software is patented, the licensee also receives a patent licence, to create, use and sell the patented source code. All distributions with modifications must be under the MPL. Any modification must be documented with the date of the modification stated. Every source code file must also contain a document called “Exhibit A”, which includes the name of the original developer and the names of all contributors.
5 Commercial Use of OSS

As discussed in Chapter 4, the open source movement has involved a great number of developers all over the world. However, when it comes to applying open source in a business context a whole new set of questions arises. In this chapter some organisations and their experiences of using open source are presented and discussed. A number of gains and risks were revealed and are summarised in Table 4. Finally, some open source business models are discussed.

5.1 OSS Applications in Business

A number of organisations involved in OSS are presented in the following sections. Most of the information about the organisations originates from interviews with employees involved in open source. An exception is Beaumont hospital, where information was provided from articles. This case is included in the study since it is one of few complete studies covering an OSS introduction. Information about MontaVista was gathered from a seminar.

Table 4. Commercial organisations using OSS.

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Founded in year</th>
<th>Number of employees</th>
<th>Revenues 2003</th>
<th>Net Income 2003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scancoin</td>
<td>1966</td>
<td>400</td>
<td>644 MSEK</td>
<td>45.2 MSEK</td>
</tr>
<tr>
<td>Axis Communications</td>
<td>1984</td>
<td>350</td>
<td>623 MSEK</td>
<td>1.2 MSEK</td>
</tr>
<tr>
<td>Combitech Systems</td>
<td>1992</td>
<td>300</td>
<td>245 MSEK</td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>1911</td>
<td>319 273</td>
<td>$89.1 billion</td>
<td>$7.6 billion</td>
</tr>
<tr>
<td>Ericsson</td>
<td>1876</td>
<td>51 583</td>
<td>117 738 MSEK</td>
<td>-10 844 MSEK</td>
</tr>
<tr>
<td>Nohau</td>
<td>1981</td>
<td>30</td>
<td>60 MSEK</td>
<td></td>
</tr>
<tr>
<td>MontaVista</td>
<td>1999</td>
<td>180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teleca</td>
<td>1998</td>
<td>3 000</td>
<td>2 455 MSEK</td>
<td>-169 MSEK</td>
</tr>
<tr>
<td>Borland</td>
<td>1983</td>
<td>1 300</td>
<td>$295 M</td>
<td>-$40 M</td>
</tr>
<tr>
<td>Beaumont Hospital</td>
<td>1987</td>
<td>3 000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.1.1 Scancoin

Scancoin is a company offering cash processing solutions. The organisation was founded in 1966 to develop and produce coin sorters based on a new technical solution. Today the organisation has 400 employees of whom 150 work at the head office or the production plant in Malmö, southern Sweden. Software development at Scancoin is done by approximately eight developers organised as a unit within the R&D department, as seen in Figure 5.1.

Scancoin AB

Scancoin International

Scancoin Europe

Scancoin Industries

R&D

Logistics

Overhead

Software

Mechanics

Embedded

Figure 5.1. Software development at Scancoin.
Scancoin first started using software in coin sorters 15 years ago. Most machines used by customers today are based on the original software platform. A problem with the first software architecture is the fact that it had become unstructured over the years. A large mass of code lines made it difficult to search for bugs or make changes. To overcome these problems a new platform based on modules was suggested. This new platform was to be written in C++. The work with the new platform was never completed due to an overwhelming workload when writing device drivers. Instead of completing this new platform the old one was migrated from DOS to Windows. The C++ project was basically abandoned.

A third platform is now being developed using OSS. The ongoing project was initiated in August 2003 when the current project leader started working at Scancoin. The introduction of OSS was combined with new work procedures based on Extreme Programming, XP.

A one-year period of convincing and presenting the concept preceded the introduction of OSS at Scancoin. When the project was launched, the first step was to introduce the Concurrent Versions System, CVS, instead of the former proprietary configuration management system. CVS is the most popular open source configuration management tool, and it is widely used and evaluated by a large number of developers. Once CVS was up and running the project continued with building a software infrastructure based on OSS. This involves among other things a build system and issue tracking. The operating system, in the machine that is developed in the project, is still Windows but it is planned to introduce Linux when the organisation is ready.

5.1.2 Axis Communications

Axis Communications was founded in 1984, with the business concept of adding value to network solutions. Today, Axis operates worldwide with offices in 14 countries and headquarters in Lund. Axis' product portfolio encompasses network cameras and video servers, servers for printers, scanners and storage devices as well as wireless access points for mobile connection to local networks and the Internet. (Axis, 2004)

As early as 1997, Axis started to gain interest in Linux and open source, mainly because the servers from Sun, used at the time, were very expensive. The involvement in Linux started with developers using Linux systems for compiling. When the Linux project evolved, the possibility to run Linux on a processor without a Memory Management Unit (MMU) appeared. Axis decided to try to develop Linux for the in-house developed processor, ETRAX, and one developer was given a few months to complete the task. The project was successful and in the year 2000 the first Linux-based camera was released to the market. Since this release, all new products are developed on the Linux platform.

Within Axis, there are certain groups assigned to keep an eye on the Linux community. About ten people are responsible for updating the Linux kernel versions as well as adjusting device drivers for new versions, and another group is doing the same for applications. These groups also function as in-house support for Linux.

---

Two open source development projects have been initiated by Axis. In both projects, Axis first developed software in-house, and when the first version was ready, the code was released to an open source community. The first project was a Bluetooth stack, which today has been abandoned since Axis no longer focuses on the Bluetooth area. The second open source project was the Journal Flash File System (JFFS). Axis took care of the JFFS for the first two years, and then handed it over to Red Hat. These projects have contributed in making Axis well known in the world of open source.

5.1.3 Combitech Systems

Combitech Systems was founded in 1992 and is owned by Saab AB. Approximately 300 consultants work in twelve different locations in Sweden, and about 80% of their business is generated outside the Saab Group (Combitech, 2004). Combitech develops integrated hardware and software solutions for embedded real-time systems, and they started using OSS by curiosity regarding possible technical advantages.

The first open source-related project at Combitech was the construction of the Wireless Open Source Platform (WOSP), which started off as a Master Thesis. The WOSP is a system that together with a WAP-browser makes it possible to retrieve wireless information from an embedded system. It is also possible to control the embedded system from the browser. The project involved a number of different open source components that were of interest to Combitech. One of them was the open RTOS eCos, which was migrated to an ARM7-processor. Then an open Bluetooth-stack from Axis was migrated to eCos. The assignment was very successful, and led to several following projects such as the Volvo Personal Communicator (VPC) and StarFrec, a Flight Recorder for the communication buses MOST and CAN. The VPC and StarFrec both use eCos as a platform.

5.1.4 IBM

The company was founded in 1911 as the Computing-Tabulating-Recording Company (C-T-R) and changed name in 1917 to International Business Machines Co., Limited, (IBM). Nowadays IBM operates in more than 160 countries and is engaged in invention, development and manufacture of information technologies, including computer systems, software, storage systems and microelectronics (IBM, 2004).

IBM researchers started to use Linux in the mid 90’s, and considered it to be an additional alternative to IBM’s AIX operating system. IBM has so-called Impact Teams and Business Development Teams to investigate new topics and to incubate and create new offerings to support the sales teams to bring new products to the customer. The Linux Impact team started to drive Linux and Open Source, especially on servers in 2000. Today, IBM has a several thousands of clients using open source. IBM is involved in a number of open source communities, and contributes with development and bug fixing. About 300 IBM developers participate in community development. IBM has also launched the Eclipse open source project as described in Section 4.4.2. There is a central organisation within IBM that monitors activities and contributions of different open source communities. According to IBM, back-end Linux is mainstream and
used by thousands of customers. However, there are not as many open source projects on the client side as on the server side today.

When a client wants to start using OSS, a number of assessment services are offered by IBM, to investigate expectations, requirements, the skills of the employees, the infrastructure etc. IBM does not handle the OSS itself, and does not leave any warranty for this. Instead IBM focuses on services like support and installation, architecture and consolidation. Regarding development of OSS modules, IBM lets the customer decide what is to be open source, except when disclosure or nondisclosure of deliverables was part of the contract.

### 5.1.5 Ericsson

Ericsson was founded in 1876 as a repair workshop in Stockholm for telegraph instruments. Today, Ericsson is one of the world’s leading organisations within telecom industry (Ericsson, 2004).

Six years ago, Ericsson released the in-house developed programming language Erlang as open source, based on a decision not to get locked in with a technology only used by Ericsson and a wish for other companies to take part in the project. A couple of different products were based on Erlang at the time. Today, an open source community for Erlang exists, but no larger organisation has joined the project.

Ericsson started to gain interest in the Eclipse project in 2002 and joined as a strategic member in 2004. There were two main reasons to start using Eclipse. Within Ericsson, several hundred different development tools are used and it is difficult to exchange experiences and techniques among developers. There is also much work with re-education of developers each time they join a development project, since they need to learn new development environments and languages. When using the Eclipse IDE these issues are solved, since Eclipse has the same appearance for different tools and languages. This gives a possibility to integrate tools and projects much easier.

Today, about 10-15 Ericsson developers work completely in Eclipse environments with development of Java based user interfaces. Two years from now the plan is to have hundreds of Ericsson’s 20 000 developers working in Eclipse. To make clear to suppliers that using Eclipse is part of Ericsson’s strategy, a membership in Eclipse has seemed necessary. To be a strategic member, Ericsson pays SEK 2 million every year. This membership allows the company to join the board, and to make decisions about requirements, planning and architecture. Ericsson is involved in a number of other open source projects, and there is a group consisting of ten people working with open source and standardisation.

### 5.1.6 Nohau

Nohau was founded in 1981 and originated as a consulting firm. Today, Nohau is situated in five countries with 30 employees and work mainly as distributor of development tools for embedded systems (Nohau, 2004). Nohau focuses on listening to the customer’s requests and desires, and adapts solutions to fit the requested wishes. In recent years, there has been an increasing interest from customers regarding OSS and the possibility to save money by using open source products. This growing interest is the main reason why Nohau now act as a
distributor for MontaVista, an organisation providing an in-house developed Linux open source platform for embedded systems.

### 5.1.7 MontaVista

MontaVista is a global organisation based in California, US, founded in 1999 with the business idea of providing customers with the MontaVista Linux distribution for embedded systems (MontaVista, 2004). MontaVista offers customers what is called a subscription to the Linux distribution, where an annual fee is paid for upgrades and support. All source code is provided. Customers are mainly from the consumer electronics market and the mobile communication market, but there are also some customers from the automation industry. MontaVista uses Eclipse technology for the in-house developed IDE DevRocket. In October last year MontaVista started an open source project, with the goal of providing the Linux kernel with hard real-time abilities. IBM, Intel, Samsung, Sony, Siemens, and also some investment companies support the project. MontaVista has earlier developed a patch to make the Linux kernel preemptible\(^4\), and the 2.6 version of Linux includes this feature as standard. The goal in the project is to shorten latency times in the kernel to make them as short as in proprietary RTOS existing today. The project is expected to finish by summer of 2005.

In a recent seminar in Malmö, MontaVista presented its MontaVista Linux distribution, and Figure 5.2 shows how the 48 participants at the seminar were distributed between different industries.

![Industry interest in embedded Linux](image)

*Figure 5.2. Industry interest in embedded Linux.*

---

\(^4\) Preemption means the ability of the operating system to preempt or stop a currently scheduled task in favour of a higher priority task. (Wikipedia, 2004)
5.1.8 Teleca

Teleca is a global consulting organization with 3000 employees, which operates in four different areas: Mobile Devices, Products, Operators & Networks and Enterprises & Industry.

The Mobile Devices Smart Phone Division has in 2004 noticed an increasing interest from customers regarding Linux for smart phones. As more advanced mobile phones replace the ones used today, a different type of operating system is required. There are four main brands of operating systems on the market: Windows, Symbian, Palm and Linux. The main reason why some of Teleca’s customers are using or consider using Linux is the low initial price. They can simply not afford to develop mobile phones with an expensive operating system. Another motivation can be the fact that the customer is not big enough to get a source code licence. The most common way to acquire Linux for mobile phones is to buy it from a distributor, usually MontaVista. Another, less common alternative, is to download Linux and tailor it for specific needs in-house.

5.1.9 Borland

Borland has been in the software business since 1983, to support clients’ software development with different tools. The first Integrated Design Environment (IDE) was developed at Borland. Today focus is on all activities in the software development process and besides IDEs like JBuilder and Delphi, Borland offers customers software tools for requirement management, configuration management, build systems and design. Borland has been involved in, and affected by OSS in many ways, as a vendor, a developer, a competitor and a user of OSS.

Borland sells tools and environments for developing OSS, but since there are open source alternatives to Borland’s products the open source movement is also a competitor. The main open source competing product is Eclipse, which is gaining market shares from JBuilder, one of Borland’s most important products. Since Eclipse is a competitor, Borland has kept an eye on the Eclipse project and early became a board member. The organisation is still involved in the project, but mainly as a member and does not participate with developers.

Borland has also played the role of open source developer when the database Interbase was released as open source in July 2000. The measure originated from the decision that Borland should no longer focus on this area. Instead of abandoning the project completely, the alternative of an open source project was chosen. It seemed to be a good way to offer customers an alternative for support and development of the database. The IBPhoenix community created their database Firebird with the released code. However it became clear that some of Borland’s customers preferred getting support from Borland and eventually the company reintegrated Interbase as a product of their own. Nowadays Borland’s Interbase and IBPhoenix’s Firebird coexist. When developing software Borland sometimes uses OSS in less business critical parts of the systems. For example, the Apache web server has been incorporated in some products.
5.1.10 Beaumont Hospital

The Beaumont Hospital, formed from three of Ireland’s oldest hospitals, opened in 1987 and employs a staff of 3000. Beaumont serves as the training hospital for the Royal College of Surgeons in Ireland and the Dublin City University (Fitzgerald and Kenny, 2003). In recent years, the Beaumont IT budget decreased steadily, and ways to save money in this area were needed. OSS was seen as the solution to achieve large cost savings.

The implementation of Beaumont Hospital’s proposed Information System infrastructure is planned in two phases, as seen in Figure 5.3. Phase one concerns implementation of an entirely Linux-based desktop environment, including desktop applications, email and a content management system (CMS). The desktop application Star Office 6, a CMS called Zope, Dicom X-ray imaging, and the email package Skyrix are examples of introduced software. Phase one is partially finished. In phase two, the operational systems are to be replaced by open source alternatives currently investigated. Alternatives considered are among others, the Vista hospital system, the Compiere Finance System and an internally developed payroll system.

Figure 5.3 The Beaumont’s proposed infrastructure is planned to be implemented in two phases. Source: IEE Software 2004

Fitzgerald and Kenny (2004) talk about principle, pragmatism and practicality when explaining the Beaumont’s motivations for moving to OSS. The principle to get the best return for the taxpayer’s money was cited by the IT manager. Savings were estimated both in purchase prices and annual maintenance costs. A pragmatic approach resulted in six months of research regarding alternative open source applications, and after some experimentation with downloaded OSS programs the IT manager was convinced that the risk involved was relatively low. In terms of practicality, the functionality and graphical interfaces of the open source alternatives were similar to the proprietary ones, and the staff rapidly adapted to new work procedures.
5.2 Lessons Learned from OSS

In the following section an analysis of the collected material from the organisations presented above is conducted. By studying the interview from each company, a number of positive and negative aspects of open source were found. The results are presented in Table 5, and are further discussed in Sections 5.2.1–5.2.6. These aspects are grouped in six different areas: Development, Security & Quality, Support, Procurement, Licences & Legal Issues and Strategic & Organisational Aspects.

5.2.1 Development

When using open source, two of the studied organisations emphasised that they had managed to substantially shorten development time, by writing less code in-house. This may cut development costs and give a better time-to-market. On the other hand it may cause developers to worry about becoming redundant and there may also be resistance to learn a new language if this is required. This was for example one of the initial problems when introducing OSS at Scancoin. At Beaumont hospital developers feared to be deskilled by losing experience with commercial software.

At Scancoin developers became more motivated when they were introduced to community thinking and could get “ego boosts” when contributing to both in-house and community projects. Another gain was the increased competence developers achieved when working with open source. According to Combitech, there is a very efficient knowledge exchange in an open source community. A less obvious gain can be that many students are familiar with open source and using it may attract students to seek employment with the company. According to IBM, people with UNIX skills, in both the public sector and in industry, do not mind adapting to Linux or to support the open source philosophy.

The most evident gain with OSS is that the code can be examined, controlled and changed as opposed to proprietary software bought from a software vendor. Some of Teleca’s customers could not get access to another source code licence, for various reasons, and therefore turned to open source. For some companies, open source may be the only way to get access to source code. This is in general appealing to engineers, and sometimes OSS might be introduced as a technical challenge rather than as a good business idea. A mistake can be to involve all the best developers in configuring for example a Linux kernel, instead of developing products. It is also important to know that it takes time to integrate different OSS components. If a complete platform is to be integrated and adapted, the organisation must provide resources for this. When developing a new Linux platform at Axis, two years of development work were devoted to attain the same product as before on another platform. To manage all these difficulties it is important to have a strong and competent advocate leading an open source introduction. At both Scancoin and Axis management supported the idea to introduce OSS from the start.

5.2.2 Security & Quality

One of the most commonly mentioned advantages of OSS is that high quality of code can be achieved. Within IBM it is believed that bug fixing and improvement is handled fast and reliable in the community. Scancoin argues that code of high quality can be achieved due to a large number of skilled and
motivated developers in the community, and the fact that peer review is conducted frequently, also discussed in Section 4.2. Dedicated users, often developers themselves, result in a large amount of testing and feedback. An extensive developer base results in products being developed for many environments and different operating systems. Combitech points out that when you let someone else take care of development, it is possible to put more resources into testing. If bugs are found, they can be corrected immediately in-house, or by hiring someone else to do it. In some cases, the open source alternatives to proprietary software also offer improved functionality. For example Beaumont Hospital found that the open source desktop suite StarOffice had a built-in XML capability, which means one can structure parts of the documents to incorporate processing logic (Fitzgerald, Kenny, 2004).

It is important to know that the code received from the community must be tested and verified just as much as the code developed in-house, before incorporated into products. It is also advised by Combitech that OSS should not be used in safety critical systems, such as life-supporting equipment, since it might be too unstable for these kinds of rigid requirements.

5.2.3 Support

When introducing OSS, it is essential for an organisation to change its approach regarding software support activities. Instead of simply buying required support from the vendor providing the software, the organisation must take more responsibility for the support, and this is done by for example Axis and Scancoin. Scancoin mainly relies on community mailing lists to get the required support. Axis has an in-house Linux group that is responsible for support. Many of the organisations in the study agree that more software knowledge is needed when using OSS, compared to when buying proprietary software. Combitech suggests that someone must have a general view of the software development within the organisation when using OSS. The positive side of the increased responsibility is the freedom of choosing support from different providers to fit the needs and requirements of the organisation.

Borland mentions that professional support might not be available for the chosen OSS. This is something to consider when evaluating OSS products. Another important issue mentioned by Combitech, is that software problems sometimes need to be solved within the company; do not trust the open source community to solve everything. IBM does not recommend customers to use open source as a primary alternative since IBM has products of their own for which they offer support.
### Table 5. Companies’ experiences of OSS

<table>
<thead>
<tr>
<th>Development</th>
<th>Gains and Improvements</th>
<th>Challenges and Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Possibility to view and control the code</td>
<td>• It takes a strong and competent advocate to introduce OSS</td>
</tr>
<tr>
<td></td>
<td>• Possibility to modify the code</td>
<td>• Takes time to integrate different OSS components</td>
</tr>
<tr>
<td></td>
<td>• Increased competence of own developers</td>
<td>• Takes time to develop a new platform</td>
</tr>
<tr>
<td></td>
<td>• Knowledge exchange in the community</td>
<td>• Developers may worry about becoming redundant</td>
</tr>
<tr>
<td></td>
<td>• Developers get an “ego boost”</td>
<td>• Resistance to learn a new language</td>
</tr>
<tr>
<td></td>
<td>• Shorter development time</td>
<td>• OSS introduced as a technical challenge, for fun</td>
</tr>
<tr>
<td></td>
<td>• Cheaper development costs</td>
<td>• Staff may fear being deskilled</td>
</tr>
<tr>
<td></td>
<td>• Easier to attract students</td>
<td>• The best developers occupied by a Linux kernel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security &amp; Quality</th>
<th>Gains and Improvements</th>
<th>Challenges and Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Skilled and motivated developers in the community</td>
<td>• Code must be tested and verified even if it is not developed in-house*</td>
</tr>
<tr>
<td></td>
<td>• Bugs can be fixed immediately</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Good testing and feedback</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Peer review within the community</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Resources for testing instead of developing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Products get developed for many different environments and operating systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Improved functionality</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support</th>
<th>Gains and Improvements</th>
<th>Challenges and Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Freedom of choosing support from different providers</td>
<td>• Need more software knowledge than when buying proprietary products</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Do not trust the community to solve all problems</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Support is not for free</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Professional support is not always available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Need for a general view of software development*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procurement</th>
<th>Gains and Improvements</th>
<th>Challenges and Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Get access to interesting technology</td>
<td>• Spending time on downloading and evaluating OSS</td>
</tr>
<tr>
<td></td>
<td>• No need for going through purchase department</td>
<td>• Do not expect OSS to be for free</td>
</tr>
<tr>
<td></td>
<td>• Services and software associated with OSS has in general a lower price</td>
<td>• Lack of a specific software vendor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Important to determine the requested functionality*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Important to look at all costs, not just licences*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Strive to use standard components*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Licences &amp; Legal Issues</th>
<th>Gains and Improvements</th>
<th>Challenges and Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• No licence cost</td>
<td>• GPL is complicated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Give attention to GPL interfaces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• GPL may interfere with functionality</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Risk losing customers not accepting OSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Risk of patent infringement*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategic &amp; Organisational Aspects</th>
<th>Gains and Improvements</th>
<th>Challenges and Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Concentrate on core competence</td>
<td>• Top management support for OSS is crucial</td>
</tr>
<tr>
<td></td>
<td>• Organisations cannot run large projects like Eclipse alone – cooperation is needed</td>
<td>• Differ between commodities and core competences</td>
</tr>
<tr>
<td></td>
<td>• Relation between organisations in an OSS project more relaxed and non-political than otherwise</td>
<td>• OSS project may go in other direction than wanted</td>
</tr>
<tr>
<td></td>
<td>• An opportunity to sell more products by releasing one product for free</td>
<td>• Critical mass of participants for OSS project</td>
</tr>
<tr>
<td></td>
<td>• Sell add-ons to OSS</td>
<td>• A fear of failure</td>
</tr>
<tr>
<td></td>
<td>• A sales argument</td>
<td>• Difficult to find short-term gains</td>
</tr>
<tr>
<td></td>
<td>• Can create goodwill</td>
<td>• Difficult to evaluate if OSS is profitable</td>
</tr>
<tr>
<td></td>
<td>• OSS is useful when a third party needs to write an application</td>
<td>• An OSS project may be abandoned</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Need to incorporate components interesting to others, rather than to yourself</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Anyone may write an application</td>
</tr>
</tbody>
</table>

* Also true for proprietary software
5.2.4 Procurement

To make use of OSS is, for most organisations, a way to gain access to an interesting technology. When working with proprietary software it is customary to go through a purchase department to manage software acquisition. When using OSS this is not necessary, a circumstance that gives rise to both positive and negative effects. The positive effect is that overhead expenditures can be saved if there is no need to place orders, handle deliveries etc. However it is not advisable to let developers spend unlimited time on downloading and trying out different open source programs since these activities can be very time consuming. Therefore Combitech argues the importance of determining the needed functionality. If a clear definition of wanted functionality is provided, downloading can be limited to this area. To benefit from OSS it is important to use standard software components. This is pointed out by both Combitech and Scancoin. Most OSS is developed in areas where there is a general interest.

When an organisation tries to assess the consequences of introducing OSS it is important to consider all costs. Just because the software is not bought the traditional way it does not mean there are no costs to take into account. Most services and software associated with OSS, such as support and applications running on OSS, has a lower price than for proprietary software. However it is important to take into account that there might be more trouble with OSS. Nohau thinks that it is important not to expect OSS to be for free. In this Master Thesis a Total Cost of Ownership model is used to take costs from the whole product lifecycle into account, which is discussed in Section 6.1. Another factor to take into account is the relation towards an open source community, which is different from the ordinary seller-buyer relation and therefore demands a different approach. In Section 5.3 the relation towards an open source community is discussed in detail.

5.2.5 Licences & Legal Issues

The most obvious gain of introducing OSS is the fact that in most cases there are no licence fees. This is mentioned by all the organisations in the study. However, a licence is attached to the software and it is important to understand the conditions stated in the licence, as described in Section 4.5. One of the problems observed by several organisations is the difficulty to understand the GPL. It is essential to know how to treat interfaces between protected parts and GPL code. If this is not handled carefully there is a risk that code is contaminated, which means that GPL must be applied to proprietary code. Sometimes code that should have been placed in kernel space is located in user space to avoid being forced to use GPL for this code. This might deteriorate functionality.

Another difficulty concerning open source is the risk of patent infringement, as described in Section 4.5. Within IBM, a central office handles all legal issues, and all OSS projects must be committed by this office. According to Michel Pyschny, Leader Business Development Linux, Central Region at IBM, this office is aware of where the problems are on the market. A way to control legal issues is to know the code of the OSS that is used. Pyschny thinks that legal issues can be a difficulty when using and developing open source and customers might think of this as a threat. However, Pyschny does not see it as a showstopper for the open source movement. IBM is waiting for statements from the EU on software patents. MontaVista offers clients a warranty in case of
patent infringement. MontaVista takes the responsibility of any patent infringements that could be caused by the software in their distribution. Thus, if the customers are using MontaVista approved code, any kind of lawsuit will be directed to MontaVista. One of the studied organisations lost a client that didn’t want to buy products with OSS because the risk of legal conflicts with patent holders. One of Borland’s clients always asks for a list with all open source components included. This implies that the company’s policy is to have a structured approach to OSS issues and to keep track of OSS programs that are accepted within the organisation.

5.2.6 Strategic & Organisational Aspects

An important aspect to consider is what strategic direction the business should take, and how its organisational structure needs to be changed. Several of the organisations in the study believe there are both positive and negative long-term factors regarding OSS that are important to evaluate. Both Axis and Combitech argue the advantage of being able to concentrate on core competencies, instead of developing every component in-house. IBM relies on OSS and can therefore concentrate on business processes and consulting services. According to Ericsson, some development projects are too large for one company to run alone, and therefore open source development can be a solution. One example is Eclipse. Relations between organisations in these kinds of projects often seem more relaxed and non-political than in other situations where organisations interact.

It is important that management explicitly states what is to be considered as commodities and which the core competencies are. This way there will be no misunderstanding when deciding what to make open source and what to protect. According to IBM, many customers don’t know what OSS means to their strategies and studies with conflicting results confuse IT Managers. IBM customers also find it difficult to know how to build an IT infrastructure that fits OSS.

Introducing OSS may also create goodwill for the organisation, and could even be a sales argument. Nohau mentions that open source is very useful when a third party wants to write an application. This is easier done when the source code for the platform is available, and it is possible to make modifications. This could be a threat as well, since anyone may write an application to an open platform. As discussed in Section 5.4, there are several ways to gain revenue from OSS. Borland explains that they sell add-ons to Eclipse as a way to make money on OSS, and Axis talks about an opportunity to sell more products by releasing one product for free and being able to sell more of another product.

A large challenge for an organisation on the brink of introducing OSS is the fear of failure, something Axis points out. In many organisations there is a resistance towards OSS, and a mistrust of the associated programs and development techniques. It is sometimes difficult to evaluate whether OSS is profitable, and it is usually hard to find short-term gains. Therefore, top management support is important, something both Axis and Beaumont Hospital declare. IBM customers often introduce open source bottom up. This works due to Linux stability and reliability, but it would be ideal if the introduction of open source was backed up by strategic decisions from top down.
There are several risks associated with OSS projects. A project can always go in another direction than the one wished for, depending on the demands from different developers and users. Axis explains that even when running an OSS project of your own, this could happen. It is sometimes needed to incorporate components interesting to others, rather than to the owner of the project. Combitech Systems mentions that if starting an OSS project, a critical mass of participants need to be involved, otherwise there is a risk that the project will be abandoned.

5.3 Relations to the Community

Besides the gains and challenges mentioned in the previous sections, a number of different ways to relate to an open source community were also revealed in the study. Some important aspects to consider when interacting with the community were found, and these are presented in this section.

The most common way to relate to a community for the organisations in the study is to use OSS components in a product sold to a third party. This is done by almost all of the companies in the study. Axis and Combitech points out that if something is corrected or modified in an OSS product, it is important to return the modifications to the community; otherwise this may result in migration problems when later versions of the components are introduced. It is also important to contribute to the community in order not to be seen as a “free rider”, which may result in difficulties to get help from other developers in the project and to get your parts integrated into the software. Axis has not noticed any resistance against organisations making money on open source as long as they contribute to the community. This is the reason why Axis believes it is ok to display the company name when contacting mailing lists.

When using OSS, it is important to have some kind of organisational structure for handling the relation towards the community. Axis argues that it is not enough to just have one person being responsible for this. Mailing lists can be very useful in several different ways. Scancoin and Axis mention that it is possible to evaluate an open source project by looking at events at the mailing lists, to see how many developers that participate, and the characteristics of messages. For example, are people writing general questions like “How do I get started”, or are they asking how they can move on with a certain issue? Scancoin explains that it is also possible to get software support and help with questions such as how conditions within licences should be interpreted.

Many of the companies have also released an in-house developed product as OSS, for example Axis, Borland and Combitech Systems. In these cases it is important that the software released is already useful to some extent, otherwise community members will not be interested in continuing the development. For the same reason, the product must be of general interest to others. There is no point in releasing something very company-specific, since no one else will be interested. Axis describes that when managing an open source project it is to some extent possible to steer it in a certain direction, but other stakeholders’ opinions must also be considered. Besides running a project, it is also possible to become an influential member in another project to have an impact on developed functionality, which is what Borland and Ericsson have done when joining the Eclipse project as a board member.
5.4 Business Models

There are many ways to define a business model. Hohensohn and Hang (2003) describe the concept as converting a business idea into a business model by orienting it and positioning it in a market to gain revenues. For a better understanding of how an organisation can find a business model based on open source, a number of different roles are explained below. Originally there were two main roles, the developer of OSS and the user of OSS. However an increasing interest in OSS from non-IT specialists have created a need for new roles. The different roles identified are (Hohensohn and Hang, 2003):

- Developers: Private developers, professional developers and academic developers.
- Distributors: A distributor bundles and packages the software and offers releases in a product-style.
- System integrators: An integrator provides services along the value chain. The integrator offers consulting and customisation of software for customers, and also integrates software into existing customer structures.
- Software and hardware companies: Sell software and hardware.
- Users: Home users, software developers, companies and institutions.

In many situations one person or company may have several of the roles above. The roles show that there are many different ways for stakeholders to interact with an open source community. Commercial organisations are interested in open source only if they can find a role that generates revenue, therefore an appropriate business model must be found.

Are there business models that can apply to the organisations discussed in Section 5.1? Hohensohn and Hang (2003) describe a number of different business models based on OSS, and a selection of these is illustrated in the last sections of this chapter. Product-related business models and service-related business models are differentiated, as seen in Figure 5.4. Some of the organisations in the study may fit into several models.

![Figure 5.4 OSS Business Models](image-url)
5. Commercial Use of OSS

5.4.1 Product-Related Business Models

The product areas can be divided into operating systems, applications and appliances. Appliances are here defined as bundles consisting of a combination of hardware and software.

*Operating System Distributors* add modules to the kernel, fix versions, provide documentation and package the software. They make money on creating a product of the software. MontaVista with the MontaVista Linux distribution fits right into this model.

*OSS Application Providers* develop new applications, create releases and distribute different types of OSS or applications running on OSS. Products include applications, development tools and administrative tools. Borland sells tools for open source development and Teleca has developed applications for mobile phones with the Linux operating system. MontaVista’s Linux distribution includes tools and packages for application development.

*OSS Appliance Manufacturers* develop appliances. This is common when software itself is not creating revenues, and is provided only as a supplement to hardware. By utilizing open source products, it is possible to acquire software beyond what is possible with internal resources and knowledge. This software can also, when managed right, be achieved at a low cost. This is the model that Scancoin, Axis and Combitech are using for their businesses.

5.4.2 Service-Related Business Models

Services regarding OSS are very similar to proprietary software services.

*OSS Distribution Sellers* sell distributions together with installation services and support. Distributors of OSS usually charge a low price for their software compared to proprietary alternatives. The way to make money with this business model is instead to sell after-sale services or to sell easy to install versions. MontaVista provides support and other after-sale services to the MontaVista Linux, and Combitech offers support services for delivered products.

*OSS Project Investors* support projects to get a larger product portfolio. Borland, IBM and Ericsson are supporting the Eclipse project to obtain a certain power to inflict on the functionality developed for Eclipse. MontaVista sponsors a number of Linux open source projects.

*System Integrators* offer IT-services, and mainly focus on big and complex IT-projects in existing IT-infrastructure. A complete range of IT-services is offered. Both Nohau, Combitech and Teleca act as consultancies offering different kinds of IT-services.
6 Financial Aspects

As described in the previous chapter, there are many examples of commercial use of open source, and different business models have been suggested. One missing piece in these models is often how to estimate economical gains in explicit terms. The authors have chosen to utilise Total Cost of Ownership, TCO, which involves looking at costs during the whole product life cycle, from procurement until the product is phased out. A customised version of a TCO model is applied to selected software components in Tetra Pak Operator Panel, TPOP.

6.1 A Cost Model – Total Cost of Ownership

The concept of Total Cost of Ownership (TCO) was suggested by Ellram and Siferd (1993) as an integrating concept to evaluate purchasing alternatives. TCO involves all costs related to an acquisition, not just the purchase price. Other cost drivers affecting TCO can be activities related to use and maintenance. Ellram and Siferd argue that there are many advantages with using TCO. The model is logical and easy to understand. Gathered information can be used for analysis and negotiation during a purchase process. An investigation (Bever and Collofello, 2002) evaluating different techniques for addressing software affordability, concludes that TCO is useful in software evaluation when examining costs from a broad perspective.

The authors have used a study conducted in 2002 (Ferrin, 2002), of 146 companies active in various industries to determine what cost drivers to look at. Ferrin points out that many leading-edge companies use the TCO approach. An important conclusion from the study was that no generic model could be derived. Different companies used different cost drivers, suited for their specific business area and products. It was even suggested that multiple models might be needed within the same company to accurately estimate costs for different products. However, some of the cost drivers revealed in the study could be seen as core drivers applicable to all models in each company. 73 respondents answered an open-ended question about which cost drivers they use, and 237 different cost drivers were found, which were then reduced to 135 due to duplication elimination. The cost drivers were categorised into 13 categories, as seen in Figure 6.1.

- Operations Cost
- Quality
- Logistics
- Technological Advantage
- Supplier Reliability and Capability
- Maintenance
- Inventory Cost
- Transaction Cost
- Life Cycle
- Initial Price
- Customer-Related
- Opportunity Cost
- Miscellaneous

Figure 6.1. TCO cost driver categories (Ferrin, 2002)

Software differs from physical products in many ways and all cost drivers in the categories mentioned above may not be applicable in a software cost analysis. The categories “Inventory” and “Operations Cost” are not relevant for intangible products like software, and have therefore been left out. The category
“Opportunity Cost” is also excluded since the authors believe that it would be very difficult to determine within the scope of this Master Thesis. Based on interviews concerning Tetra Pak software management and several studies regarding cost analysis of OSS, some of the cost drivers from the study are considered to be more relevant than others in a software context. It is also important to have a manageable number of cost drivers and balance gain of details against cost of effort. The suggested cost drivers are presented in Table 6. A complete list of identified cost drivers from Ferrin’s study can be found in Appendix C.

### Table 6. Selected cost drivers, suitable for software.

<table>
<thead>
<tr>
<th>Initial Price</th>
<th>Logistics</th>
<th>Quality</th>
<th>Technological Advantage</th>
<th>Transaction Cost</th>
<th>Life Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Cost</td>
<td>Freight</td>
<td>Customer Downtime</td>
<td>Suitability for Intended Use</td>
<td>Administration of Post-Purchase Agreements</td>
<td>Long-Term Usage</td>
</tr>
<tr>
<td>Initial Purchase Price</td>
<td>Leadtime</td>
<td>Inspection</td>
<td>Flexibility for New Use</td>
<td>Supplier Conversion Cost (Cost to Change Supplier)</td>
<td>Redesign Cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rework</td>
<td>Changing Technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplier Reliability and Capability</td>
<td></td>
<td>Maintenance</td>
<td>Supplier Ability to Change Technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trust</td>
<td>Transaction</td>
<td>Training</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplier Capabilities</td>
<td>Downtime</td>
<td>Downtime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplier Support</td>
<td>Labor</td>
<td>Long-Term Maintenance Costs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment Terms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplier R&amp;D Capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Familiarity with Supplier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer-Related</td>
<td>Maintenance</td>
<td>Warranty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Satisfaction</td>
<td></td>
<td>Safety</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer Perceptions</td>
<td></td>
<td>Support Costs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td></td>
<td>Installation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warranty</td>
<td></td>
<td>Ease of Operation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safety</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support Costs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 6.2 Cost Analysis of Software in Tetra Pak Operating Panel

The TCO model presented in the previous section will here be applied to software in TPOP, described in Chapter 2. To fully make use of a TCO model it is necessary to have more detailed information about costs and this can only be achieved once a projected plan is established, which is not in the scope of this Master Thesis. Costs will be briefly analysed for the current software InTouch and Windows, as well as scenarios where Windows is replaced with Linux in Section 6.2.2 and InTouch is replaced with software developed in an open source project initiated by Tetra Pak in Section 6.2.3. A reason for choosing to analyse these software components in a Tetra Pak machine is to give a rather straightforward example of how to use the TCO model. The authors discuss cost drivers considered relevant within the ten different categories from Table 6. Cost drivers are displayed in the margin, next to each category. This analysis is not to be seen as a complete investigation, but gives examples of how the model may be used to evaluate software costs.

When presenting the TCO analysis each category of cost drivers is discussed individually. In Section 6.2.1 the cost drivers presented above are applied to
Windows and InTouch in TPOP. Information was gathered from interviews with four Tetra Pak employees within the subsidiaries Carton Ambient and Tetra Top within Carton Chilled. Some of the interviewees were asked to estimate the largest software cost drivers. Answers are presented in Figure 6.2. The cost driver analysis of the scenario with a Linux alternative in Section 6.2.2 and an open source project scenario in Section 6.2.3 are derived from some external sources as well as the authors’ estimations with studied material in mind. Table 8 contains a short summary of the cost analysis.

![Figure 6.2. Largest software cost drivers according to some Tetra Pak employees.](image)

### 6.2.1 InTouch & Windows

In TPOP Tetra Pak uses a Windows operating system from Microsoft, and the SCADA software InTouch from Wonderware. There is no central organisation at Tetra Pak handling OEM licences in the same way as the department Tetra Pak Information Management (TPIM) handles desktop software.

**Initial Price:** A TPOP including hardware and software costs approximately SEK 50 000. Installation of both InTouch and Windows is done before delivery and is included in the price. Tetra Pak gives specifications about how the software should be installed. Approximately 500 OEM licences are purchased each year. In addition to this Tetra Pak gets 60 free developer licences included in the price of the 500 from Wonderware. These can be downloaded from a Tetra Pak Carton Ambient website by employees. Basic support is included in the price of SEK 7000 per licence. An OEM-licence for Windows is cheaper than a full-
scale license, and costs approximately SEK 2000. About 500 licences are bought per year. Innoscan delivers to Tetra Pak Carton Ambient and B&R delivers to Tetra Pak Carton Chilled. Table 7 summarises costs for software licences.

Table 7. Annual software licence costs.

<table>
<thead>
<tr>
<th>Software</th>
<th>Number of licences</th>
<th>Price (SEK)</th>
<th>Total cost (SEK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>InTouch</td>
<td>500 + 60</td>
<td>7 000</td>
<td>3 500 000</td>
</tr>
<tr>
<td>Windows</td>
<td>500</td>
<td>2 000</td>
<td>1 000 000</td>
</tr>
</tbody>
</table>

**Logistics:** Since both InTouch and Windows are preinstalled, freight costs are included in the price. For the same reason, there is no extra lead time for software.

**Quality:** Downtime is most of the time not caused by software itself, but by the human factor. The largest cost for quality assurance and inspection occurs when introducing new software or a new version since most effort regarding testing is done at this stage. There was one year of testing when InTouch was to be used for the first time. When error reports from the field are received, the programs are verified again.

**Technological Advantage:** Tetra Pak chooses products from the supplier product portfolio. It is not possible to get non-standard versions but this is usually not a problem. InTouch has so far been well suitable for Tetra Pak needs, but some modifications are made. One example of this is TPOP Basic, created by one of the developers. This is software configured for easy installation on the field. When creating TPOP Basic there is work with configuration of the operating system, software testing, bug fixing of drivers and field-testing, estimated to take one man-year. The last ten years two main versions of TPOP Basic have been developed.

Tetra Pak would probably not pay Wonderware to develop special functionality. An interesting case at the moment is TP Modena in Italy that wishes for some extra functionality from Wonderware. If this request is fulfilled, it will probably be in the next version, and would then not be of much use for Tetra Pak since versions are not frequently updated in the machines. Regarding Supplier Conversion Costs, a change of supplier would be very expensive. Large gains must be proven, for example a possibility to integrate all components into the same system.

**Supplier Reliability and Capability:** A Tetra Pak supplier must have high capacity and be globally represented. Both Microsoft and Wonderware fulfil these requirements and have been supplying software for TPOP for 10 years. Stable business relations have been established. InTouch has a support organisation providing service via telephone or e-mail, and this is included in the licence agreement. There is also a Wonderware support website available from Tetra Pak’s intranet. If more advanced support is needed, there are additional costs. The support function has been criticised; often it takes a long time to get answers from Wonderware. Microsoft offers support via the usual channels, but most of the time very little support is needed.
**Maintenance:** Approximately 150 developers work with maintaining TPOP software at Tetra Pak, but this is generally not their main task. When upgrading to a new version, this is done on all machines at the same time. Two persons work full-time with maintenance of TPOP software such as development and support. They are also responsible for sending updates to the customers. Staff using InTouch is mostly trained internally at Tetra Pak. Training can also be purchased from Wonderware. One week’s course costs SEK 10 000 and this provides the basics such as knowledge to create simple applications.

**Transaction Cost (transactions concerning purchasing):** A list of all licence numbers in use is administered by each machine system, since Wonderware requests this. Wonderware demands hardware keys for certain countries such as China and Russia to protect software from being copied illegally. To make administration easier, Tetra Pak has decided to use hardware keys for all markets. Two to three persons administer hardware keys or “dongles” for InTouch as part of their job. If a hardware key breaks when the guarantee time of one year has expanded, the broken one must be sent back, and a replacement key can be bought for SEK 2 000. A broken key may result in downtime for the customer, but a temporary key can be downloaded from the Wonderware website rather quickly. Unfortunately, there are sometimes installation problems with the keys, such as that a certain amount of electrical current is needed to get them to work, and this is not always provided from the computer.

Two persons work partly with handling all global agreements and negotiations with software suppliers. Another person scans the market and about 40% of her time is dedicated to these kinds of tasks. It is difficult to start using a new supplier, since a goal at Tetra Pak is to decrease the total number of suppliers for machine development from 1100 to 450. Negotiations with new suppliers take time. For Wonderware the latest negotiations went on for about a year. It is very expensive to change supplier since new software needs to be implemented and this affects the whole organisation. Changing software is much more complex than changing hardware. New applications are needed, as well as training of the 150 persons that work with InTouch today in some way. It will probably not be an easy migration. There is no operational purchasing department for OEM software at Tetra Pak, since software is preinstalled on the TPOP. The Tetra Pak Technical Service division purchases spare hardware keys.

**Life Cycle:** At Tetra Pak software versions are used for a very long time, something the suppliers want to avoid, since supporting old versions for a few customers is costly. This may result in problems and increased costs for Tetra Pak when support is no longer offered from the supplier, which is an existing problem for Windows 3.11. Additionally, Microsoft has announced that 2006 is the deadline to purchase Windows 3.11. Changing version is expensive, since all machines of a certain model in the field need to be updated. Brian Gammage from Gartner group says that it is neither easy, nor cheap to upgrade from, for example Windows 95 to XP (Danielsson, 2004). At the moment, when Tetra Pak has stable and tested software, one person at Carton Ambient and one at Carton Chilled works about four weeks each year with requests from people regarding investigation of errors and requested new functionality. They also analyse possibilities and consequences of changing to a new version.
Customer-Related: Tetra Pak guarantees a certain efficiency level in the machines, but does not give a specified warranty for software. The customer may sign service contracts, and then service regarding software is included. The contracts may last between a year and the lifetime of the machine. At Tetra Top there are around 20 service technicians working with supporting customers with their machines, including software. Carton Ambient has a group of 40 service technicians, but has additional technicians within the market companies working on the field. Customers request continuity and security and therefore it is important that the Graphical User Interface is similar in all machines. Customers see TPOP as a black box but sometimes they might request extra functionality without specifying how to implement this with software or hardware.

Miscellaneous: When it comes to warranties, there are principally none from Microsoft regarding errors in the code, but there are service packs to correct errors. Wonderware also offers this service. Most of the time Tetra Pak cannot use them since it is company policy to avoid large updates once a machine is in production. If software problems arise, Tetra Pak has the burden of proof against the software supplier to prove that it was the program that caused the error. This is often very difficult if it concerns a large supplier like Microsoft. Another difficulty for Tetra Pak is to guarantee the safety of software. TPOP software cannot interfere with the physical safety system in the machine. The most important is that the machine is secured to not harm people in case of an accident. There is always a risk assessment before releasing a new machine or a new function to clients. One person, the Machine Safety Advisor, works full-time with machine safety. This includes education of employees regarding this area.

Tetra Pak offers support to customers if they have signed a service agreement. Technical Service has what they call SuperUsers for each machine system, and they can offer help. If they can’t solve a problem, the question is forwarded to one of the TPOP developers mentioned in the Maintenance part of this section. All customers get training in how to use a machine; they are for example taught how to install the software on TPOP in case of a reboot. The customer is not allowed to make modifications in software. To install a full update of all TPOP software on a Tetra Top machine is estimated to take about two hours for a service technician, and Tetra Pak pays the customer for this time.

6.2.2 Linux

Scenario: The Windows operating system in TPOP is replaced by a purchased Linux distribution from for example Red Hat or Novell.

Initial Price: No licence costs, but an initial purchase price. Only one copy is needed, which is then free to distribute. As long as the same version is used, there is no need to purchase more copies. Prices for Linux distributions:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat Enterprise Linux 3</td>
<td>SEK 2800 (Computer Sweden, 2004)</td>
</tr>
<tr>
<td>Novell SuSe Linux Professional 9.2</td>
<td>SEK 884 (Novell, 2004)</td>
</tr>
</tbody>
</table>

Logistics: It must be investigated if it is possible to get Linux pre-installed on TPOP by Innoscan or B&R. The interviewees do not think this would be a problem.
Quality: As for new versions of Windows, there will initially be extensive testing and inspection of the selected Linux distribution. Changing from Windows to Linux might generate some additional costs.

Technological Advantage: Since there are many different Linux distributors, one of them will probably be able to satisfy Tetra Pak’s needs. If new technology or functionality is requested, it may be easier to receive from a Linux supplier than from Microsoft, since there is more competition between different Linux distributors. However, this may not be interesting for Tetra Pak concerning the operating system. It is also possible to adapt the operating system if requirements change. The Technology Manager from Boss Media (Ricknäs, 2004) explains that Linux is a clean operating system, which means that it is possible to configure only the needed services and everything else can be turned off.

Supplier Reliability and Capability: Regarding R&D capabilities, Linux is continuously developed by a large community, and by several commercial organisations. If support is needed, the Linux community may help. The Linux distributor also provides support. Tetra Pak is familiar with current suppliers and would have to build new knowledge if using Linux.

Maintenance: Few people are involved in maintaining the operating system for TPOP. Maintenance includes developing and maintaining interfaces and connections to applications. Training costs for employees performing these tasks are currently small. If changing operating system, it will be necessary to build Linux competence, which generates initial costs.

Transaction Cost (transactions concerning purchasing): With Linux, it is not necessary to handle licences to be prepared for inspections from BSA etc. Instead, there is a need for understanding the terms of GPL. If a distributor is involved there will still be negotiations and agreements. It may be easier to change supplier, since there are many Linux distributors to choose from. They are supposed to follow standards from the Free Standards Group therefore different Linux distributions should be quite similar.

Life Cycle: There is no risk of compulsory upgrading to new versions since there are no expiring licenses when using OSS. Tetra Pak would not need to change from an older version of Linux because of a request from a supplier.

Customer-Related: Tetra Pak’s service agreements with customers will probably not change. In general, customers do not have any preferences regarding the operating system.

Miscellaneous: There is probably no difference regarding warranties for code errors between Linux suppliers and Microsoft, but it might be easier and faster to get help with bug fixing from a Linux distributor. Red Hat (Åsblom, 2004) and MontaVista offer a warranty that the code is free from patent infringement. If there are higher security demands on the operating system when connecting the machines to Internet, Linux may be a better alternative when it comes to virus attacks, as discussed in Section 4.3.
6.2.3 Open Source Development Project Initiated by Tetra Pak

Scenario: Tetra Pak develops SCADA software to replace InTouch and releases the code as OSS when the software has the functionality needed to interest other stakeholders. The code is kept in a repository, for example SourceForge, to make it available for those who want to participate in the project.

**Initial Price:** When acquiring software by profiting from open source development there will be no costs for OEM-licence royalties. However, a project must be launched and this means a large initial cost. This includes for example costs for developing the first version to be released, and marketing of the software to build a community and gain interest from others in the industry. It may also be costly to develop additional functions if the community does not contribute enough.

**Logistics:** There will not be any freight costs, since the program can be downloaded from the repository.

**Quality:** If Tetra Pak can profit from testing performed by other community developers, initial testing at Tetra Pak when a new version is introduced might be performed faster.

**Technological Advantage:** Internal software development costs at Tetra Pak may be reduced if other developers’ contributions are used. This might also help shorten development time. Software will be better adapted to Tetra Pak’s needs, and by making a broader application base it may be possible to reduce time for configuration and adapting applications to each machine system. In an open source project it is possible to influence the software architecture to include flexibility for future requirements.

**Supplier Reliability and Capability:** An open source project is more unpredictable than a traditional software supplier like Wonderware. The project may go in another direction than wanted, or may be abandoned. However, software can more easily be adapted to new requirements occurring when developing a new type of machine. Since Tetra Pak has access to the code it is always possible to make adjustments in-house or to hire consultants to do it. Support can be handled in-house, or by using the community for questions.

**Maintenance:** It is important that the software is continuously maintained. Maintenance of the software can be handled in the community, but it is essential that someone is responsible for this and sees to it that all necessary tasks are performed, such as providing documentation. Different kinds of documentation for architecture, design and end users must be made available.

**Transaction Cost (transactions concerning purchasing):** It will no longer be necessary to keep track of licence numbers when using software from an open source project, and the problem with keeping track of hardware keys is also eliminated. However, Tetra Pak must have knowledge of OSS licence terms, and what consequences they may lead to.
Life Cycle: Key developers may abandon the project, which makes it difficult to forecast how the project evolves. The project also may take another direction than wanted. Therefore it is important for Tetra Pak to be engaged in the project and keep informed of other stakeholders’ interests. When the code is available, it is possible to use the same version of software for a long time and there is no need to negotiate with suppliers to get licences for old software. When a change is required it may be possible to pay developers to develop a certain feature and then get this integrated into the project.

Customer-Related
Tetra Pak’s service agreements with customers will probably not change. In general, customers do not have any preferences regarding software, except for the graphical user interface. If launching a project it would be desirable to keep as much of the graphical structure as possible.

Miscellaneous: Tetra Pak must handle all testing regarding safety control. It may be possible to influence safety features in the project to avoid solutions likely to be exposed to virus attacks. When launching a project, Tetra Pak has the possibility to make specifications so that software is easy to install and learn.
### Table 8. Summary of the cost analysis

<table>
<thead>
<tr>
<th>Cost Driver</th>
<th>Windows and InTouch</th>
<th>Linux</th>
<th>OSS Project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial Price</strong></td>
<td>InTouch SEK 3 500 0000</td>
<td>Red Hat Linux SEK 2 800</td>
<td>Large one-time initial costs for an open source project, but no purchase cost.</td>
</tr>
<tr>
<td></td>
<td>Windows SEK 1 000 000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Logistics</strong></td>
<td>Freight cost included in the figures above.</td>
<td>Can Linux be preinstalled on TPOP by the hardware supplier?</td>
<td>No freight costs.</td>
</tr>
<tr>
<td><strong>Quality</strong></td>
<td>Large costs for quality assurance and inspections when introducing new software.</td>
<td>Additional initial costs when changing to Linux.</td>
<td>Can Tetra Pak profit from community testing?</td>
</tr>
<tr>
<td><strong>Technological Advantage</strong></td>
<td>InTouch is suitable for Tetra Pak needs after in-house modifications. Difficult to influence supplier development.</td>
<td>Competition between Linux suppliers gives negotiation opportunities for Tetra Pak.</td>
<td>Software may be better adapted to Tetra Pak needs.</td>
</tr>
<tr>
<td><strong>Supplier Reliability &amp; Capability</strong></td>
<td>Microsoft and Wonderware have high capacity and are globally represented. Both organisations offer support.</td>
<td>Linux is continuously developed by a large community. Linux distributors often offer support.</td>
<td>Reliability and capability depends on number of competent participants in the project.</td>
</tr>
<tr>
<td><strong>Maintenance</strong></td>
<td>Approximately two persons work full time with maintaining and developing TPOP software.</td>
<td>Few people are involved in maintaining the operating system for TPOP.</td>
<td>Essential that maintenance is ensured in the project.</td>
</tr>
<tr>
<td><strong>Transaction Cost</strong></td>
<td>Much time spent administrating hardware keys.</td>
<td>No need for licence tracking.</td>
<td>Tetra Pak must have knowledge about OSS licenses.</td>
</tr>
<tr>
<td>(transaction concerning purchasing)</td>
<td>Software suppliers do not want Tetra Pak to keep outdated versions.</td>
<td>No expiring licenses or versions.</td>
<td>Difficult to forecast how the project evolves.</td>
</tr>
<tr>
<td><strong>Life Cycle</strong></td>
<td>Customers request continuity and security, but see TPOP as a black box.</td>
<td>Customers do not yet have any preferences regarding operating system.</td>
<td>Desirable to keep the same graphical structure as before.</td>
</tr>
<tr>
<td><strong>Customer-Related</strong></td>
<td>There are no warrantees from software suppliers.</td>
<td>No warranties. Linux is less exposed to virus attacks than Windows.</td>
<td>Maybe easier to influence protection against viruses in an open source project.</td>
</tr>
</tbody>
</table>
6.2.4 Cost Comparison

In Table 10, a comparison between Linux and Windows is performed concerning the ten categories mentioned above. At each category the effect on costs is presented using the symbols described in Table 9. The same symbols are used in the comparison between InTouch and an open source project initiated by Tetra Pak, presented in Table 11.

Table 9. Effects on costs.

<table>
<thead>
<tr>
<th>Cost effect</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decreasing costs, positive effect</td>
<td>+</td>
</tr>
<tr>
<td>Increasing costs, negative effect</td>
<td>-</td>
</tr>
<tr>
<td>No significant change</td>
<td>0</td>
</tr>
<tr>
<td>Unknown cost</td>
<td>?</td>
</tr>
<tr>
<td>Both positive and negative effects</td>
<td>+/-</td>
</tr>
</tbody>
</table>

Table 10. Linux replaces Windows.

<table>
<thead>
<tr>
<th>Cost Driver</th>
<th>Effect</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Price</td>
<td>+</td>
<td>Only one copy needs to be purchased.</td>
</tr>
<tr>
<td>Logistics</td>
<td>0</td>
<td>Need to find a supplier that preinstalls Linux.</td>
</tr>
<tr>
<td>Quality</td>
<td>0/-</td>
<td>Initially larger testing costs.</td>
</tr>
<tr>
<td>Technological Advantage</td>
<td>+</td>
<td>Linux is more flexible.</td>
</tr>
<tr>
<td>Supplier Reliability &amp; Capability</td>
<td>-</td>
<td>Tetra Pak is familiar with Microsoft.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>0/-</td>
<td>Need for some Linux competence.</td>
</tr>
<tr>
<td>Transaction Cost</td>
<td>+</td>
<td>No licence administration.</td>
</tr>
<tr>
<td>Life Cycle</td>
<td>+</td>
<td>No expiring licenses or versions.</td>
</tr>
<tr>
<td>Customer-Related</td>
<td>0</td>
<td>No customer opinions regarding operating system.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>0</td>
<td>No significant changes.</td>
</tr>
</tbody>
</table>
Table 11. Open source project replaces InTouch.

<table>
<thead>
<tr>
<th>Cost Driver</th>
<th>Effect</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Price</td>
<td>-</td>
<td>Large one-time initial costs for an open source project, but no purchase cost. Cost of project depends on contribution from community.</td>
</tr>
<tr>
<td>Logistics</td>
<td>0</td>
<td>No freight costs.</td>
</tr>
<tr>
<td>Quality</td>
<td>+</td>
<td>Assuming that Tetra Pak can profit from community testing.</td>
</tr>
<tr>
<td>Technological Advantage</td>
<td>+</td>
<td>Assuming that Tetra Pak can profit from community development.</td>
</tr>
<tr>
<td>Supplier Reliability &amp; Capability</td>
<td>+/-</td>
<td>An open source project is more flexible, but also more unpredictable than a software vendor like Wonderware.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>?</td>
<td>Difficult to estimate.</td>
</tr>
<tr>
<td>Transaction Cost</td>
<td>+</td>
<td>No need to keep track of licences and hardware keys, but Tetra Pak must have knowledge about OSS licenses.</td>
</tr>
<tr>
<td>Life Cycle</td>
<td>+</td>
<td>No expiring licenses or versions.</td>
</tr>
<tr>
<td>Customer-Related</td>
<td>0</td>
<td>Customer has no opinions regarding software.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>0</td>
<td>Maybe easier to influence development of protection against viruses in an open source project.</td>
</tr>
</tbody>
</table>

6.2.5 Conclusions

The above examples show that with the presented TCO model it is possible to get a broad picture of software costs in reasonable short time. Our analysis shows how to make use of the TCO model, and how the results will appear. However, to perform a more accurate evaluation and comparison of software components, more time and resources to collect information are needed. It is not certain that all requested information is possible to attain, since Tetra Pak’s organisational structure is fragmented. A more detailed description of alternatives is also necessary to fully make use of a TCO model.

The software costs argued by Tetra Pak employees in Figure 6.2 are interesting to consider in the TCO analysis. Important costs can be related to development and configuration of applications as well as to administration of licences and hardware keys. In the TCO model these costs correspond to the categories Technological Advantage, Maintenance and Transaction Cost. The Linux scenario would slightly increase costs in the Maintenance category. For the OSS project it was not possible to draw any conclusions regarding Maintenance costs due to lack of information. Both scenarios included in the analysis would result in decreasing costs in the two categories Technological Advantage and Transaction Cost. To sum up, two of the three cost categories considered important by Tetra Pak employees would be decreased in the two scenarios.
7 Analysis and Discussion

In this chapter the authors try to answer the Main Questions at Issue presented in Chapter 1. General findings, concluded from literature and from interviews with the organisations presented in Chapter 5, are summarised, followed by a discussion of how Tetra Pak can relate to risks and gains when considering OSS issues. The validity of the Master Thesis work is discussed in Section 7.5. Finally the authors offer some thoughts about the future of OSS in general.

7.1 General Findings about Commercial Use of OSS

The organisations in Chapter 5 use OSS in different ways to support their business. Some of these have OSS product-related business models, for example MontaVista, which develops and distributes embedded Linux. Other organisations such as Combitech and Teleca have models related to OSS services. Some of the organisations use OSS in embedded systems, for example Scancoin and Axis. This gives the answer to Main Question at Issue number 1, concerning business models for OSS.

OSS is often introduced bottom-up, since developers see many gains with using open source in their daily work. These gains include the possibility to control the code and to get assistance from other community developers. However, a bottom-up approach may not be the best way to start. For an OSS introduction to succeed it is essential to have management support and a strong advocate to lead the project. In a larger, bureaucratic organisation management approval is required before initiating OSS activities. In a small, more informal organisation it might be possible for one single initiator to introduce OSS and show fast results. In both cases it is necessary to define a role concept in order to assign tasks and responsibilities. To truly profit from OSS there must be a clear strategy and the reasons for migration should be known. It must be defined which parts are core competencies and which ones are not. This definition can be applied as a guideline when deciding where OSS can be used and if in-house code can be released. It is also important to know the conditions of the OSS licence in use. Some licences are stricter than others. GPL states that all software containing, using or linking to software licensed by GPL must also be GPL. BSD is more liberal and has therefore been used in various commercial applications. Finally, the relation to the community must be clarified, such as discussions about upgrading, and how to make use of mailing lists.

There might also be long-term strategic advantages. It is for example possible to launch large development projects if several organisations cooperate, resulting in moderate costs for each organisation. To make OSS a long-term strategy it is important to monitor community activities and be prepared with alternative solutions for problems. Such problems could be an undesired roadmap for the development project. To get respect from the community the company needs to contribute to the project in some way and not just exploit the work of others. Question number 2 about gains and risks associated with OSS is partly answered by this discussion, but also in Sections 7.2 and 7.3.
7.2 Risk Assessment for Tetra Pak

Throughout this Master Thesis work a number of risk factors associated with OSS have been revealed. By interviewing Tetra Pak employees from different departments the authors have identified some risk factors considered relevant for Tetra Pak. To get an overview of the analysis, risk factors are grouped under the same topics as in Chapter 5: Development, Security & Quality, Support, Procurement, Licenses & Legal Issues, and Strategic & Organisational Aspects.

7.2.1 Development

If OSS is to be introduced at Tetra Pak this will to a large extent influence software development. To succeed with an OSS introduction it is crucial to have a competent and strong project manager, since there will be several obstacles to overcome. Tetra Pak does not have much knowledge regarding OSS and therefore it will surely take time to integrate different OSS components with existing software. The result will be high initial costs and might create a general resistance towards OSS among Tetra Pak employees. Another risk is that OSS development is depending on community contribution, which cannot easily be controlled by Tetra Pak.

7.2.2 Security & Quality

It’s difficult to evaluate whether OSS is more or less secure than proprietary software. There are many arguments from OSS defenders as well as opponents but no large independent study has been done to assess OSS security and quality. Therefore Tetra Pak must test and verify all software that is to be integrated, proprietary as well as OSS.

7.2.3 Support

Today, software engineering is not given high priority within Tetra Pak, since it is not considered to be a core competence. There is no comprehensive organisation for handling OEM software, which results in a fragmented support approach for Tetra Pak employees. If Tetra Pak decides to use open source, this type of organisational structure would cause problems when support is not provided the same way as it is for proprietary software, discussed in Section 5.2.3. The organisations studied in this Master Thesis use different solutions for support. Tetra Pak needs a strategy for software support that specifies how support should be provided in different situations.

7.2.4 Procurement

When acquiring OSS there is often no specific software vendor in the traditional sense. This means a different relation to the supplier where Tetra Pak must put more effort into managing the purchasing process for OEM software. To control OSS acquisition Tetra Pak would have to make sure that required functionality is clearly defined. It is also important to determine procedures for downloading and evaluating OSS. This way it is possible to avoid confusion among co-workers. Another important issue to consider is that standard software components must be used if OSS is to be profitable, otherwise it will be difficult to find useful software.
7.2.5 Licences & Legal Issues

Large international organisations like Tetra Pak are more exposed to lawsuits than smaller, less well-known companies. For this reason Tetra Pak must be aware of risks associated with patent infringement. However, there have been very few incidents of this kind in software history. If Tetra Pak considers using Linux, as described in the scenario in Section 6.2.2, it is necessary to know the conditions of GPL and to give attention to GPL interfaces. Since Tetra Pak are concerned with protecting intellectual property, the viral characteristics of GPL may be limiting. Question number 4, about legal implications, is answered here.

7.2.6 Strategic & Organisational Aspects

Today OSS is an unknown concept within Tetra Pak. This can result in a fear of failure since people working with software do not know what to expect of an OSS introduction. It can be difficult to demonstrate short-term gains and this increases doubt of succeeding with OSS. Altogether this may give rise to a negative attitude that complicates OSS introduction. This threat can be eliminated if top management within Tetra Pak R&D clearly states the objectives, and supports change right from the start. At this stage it is essential that it is explicit what Tetra Pak core competences are, as well as what can be defined as commodities. The later can be open source since there is less need for protection.

If Tetra Pak was to initialise an in-house open source development project, new risks arise. The first issue is to find a critical mass of participants, to get the project started. The idea must be of general interest to other organisations. If the project cannot attract developers from other companies, Tetra Pak must handle most of the development, and this way there is no point of making the software open. Even if external developers join the project there will be large initial costs. There is no guarantee that these investments will pay off since an open source project is unpredictable and may go in another direction than wanted.

7.3 Possible Gains for Tetra Pak

There are not only risks with OSS; there is also much to gain. In this section the authors present OSS gains of value for Tetra Pak. Just as in the previous section, the analysis is based on facts and opinions from the studied organisations and Tetra Pak employees.

7.3.1 Development

The very meaning of OSS is that it is possible to control the code, in other words it is possible to view, change and freely distribute it. The authors believe this would be a gain for Tetra Pak since it is possible to fix bugs when needed, instead of waiting for the supplier’s updates. Another gain, noticed by many of the organisations presented in Chapter 5, is that developers can get an “ego boost” by contributing to an open source project. This is due to attention and recognition from other developers in the community. There is also open communication in the community, which leads to efficient knowledge exchange, as discussed in Section 4.2.
7.3.2 Security & Quality

As mentioned in Section 7.2.2 there are diverging opinions on whether OSS in general is more or less secure than proprietary software. Those who advocate OSS believe that peer review between community developers contributes to high quality. They also argue that higher quality can be achieved when bugs can be corrected by anyone and not only by the software supplier. If Tetra Pak gets involved in an open source project, higher quality software could be obtained if programs are customised for Tetra Pak needs. It may also be possible to have an impact on what security solution to choose in the project.

7.3.3 Support

Within Tetra Pak there has been complaints on how support is handled by software suppliers. For proprietary software only the originator can provide the kind of support that requires source code access. This is not the case for OSS since several providers can compete on the market. OSS might give an opportunity to negotiate more satisfying support conditions.

7.3.4 Procurement

Sometimes developed products require technology not provided by the current software supplier, and a supplier change is required. Such a procedure is costly and results in much work for Tetra Pak. Development environments, programming languages, protocols and supplier specific standards all need to be replaced. By acquiring software with open standards such as OSS it would be easier to change supplier. Tetra Pak would not be forced to throw out the old platform or to invest in a new one. Vendor lock-in can be avoided and large cost savings achieved if Tetra Pak changes technology.

7.3.5 Licences & Legal Issues

When using OSS there is no licence cost. It is enough to buy one copy, which then can be duplicated and distributed all over the organisation. As discussed in Section 6.2.1, there are many problems associated with hardware keys. With OSS, Tetra Pak could avoid administration costs regarding hardware keys as well as licence tracking.

7.3.6 Strategic and Organisational Aspects

As discussed several times in the thesis, Tetra Pak has trouble with keeping old software in the machines since it is difficult to get support. The supplier would prefer Tetra Pak to upgrade every time a new version is released, and in some cases old licences are no longer sold. If OSS is used, there are no expiring licences resulting in compulsory upgrading due to supplier demands. Tetra Pak would be able to keep old versions of software in the machines as long as needed.

If a valuable software solution developed in-house is considered to be outside the range of Tetra Pak core business, an option can be to release it as an open source project. This way, the solution can be developed for the benefit of Tetra Pak as well as for other stakeholders, and cannot be registered as a patent by another organisation. There might be several gains with starting an open source project. Some software issues are too big for one single organisation to solve, but if several organisations cooperate in an open source project much can be achieved.
with small investments. One example of this is the Eclipse project, where many organisations have contributed to the successful Eclipse IDE. Participating organisations have experienced that relations between organisations are more relaxed in an OSS context than under other circumstances.

7.4 Is Tetra Pak Prepared for OSS?

An organisation using OSS needs to have a structured approach to software management. There is no traditional software vendor taking care of certain issues such as suggesting a roadmap for software strategy. Tetra Pak lacks tradition of using standard procedures for software development, since this is not considered a central part of Tetra Pak technology as discussed in Section 2.4. People working with software at Tetra Pak do not have a network for communication between different departments. Bearing this in mind, changes are required to make an OSS introduction feasible. A potential improvement of the information and communication structure could be achieved by developing procedures and standards for coherent software documentation.

One of Tetra Pak’s core values is “Innovation and Creativity”, which express that the organisation shall gain the competitive edge by following technological developments. Today there is a growing interest for OSS among organisations dealing with software in various ways, and Tetra Pak must keep updated on how the situation evolves. To be able to thoroughly evaluate OSS it is necessary to get involved in an OSS project in some way. A first step might be to introduce an OSS component in a non-critical system. A change of software platform is required from time to time for Tetra Pak, and there are always large costs involved. OSS may be an interesting alternative the next time this occurs. The authors have summarized some advice, presented in Table 12, to consider before introducing OSS. These partly answer question number 5, how Tetra Pak can cooperate with OSS communities. This first step then needs to be evaluated and if judged successful, a larger project involving OSS can be initiated. A good way to assess the pilot project could be to determine how different TCO cost drivers are affected, using methodology similar to the one used in Chapter 6. This discussion gives the answer to question number 6 dealing with prerequisites needed to introduce OSS.
Table 12. Some advise before starting.

- Look for sweet spots with high potential return from using OSS.
- Ensure management support for OSS.
- Start with non-critical systems.
- Determine requested functionality.
- Consider releasing internal software as open source if it is not a strategic differentiator.
- Build up expertise and relationships with the OSS movement.
- Developers must understand the open source licences they are using.
- Introduce tools used in the open source community when engaging in a project.
- Create a safe list of OSS for the organisation to use or consider. Keep it current.
- Ensure configuration control.

7.5 Validity

During literature research, the authors discovered that many sources are biased; either very positive or very negative towards OSS. There are few objective studies performed comparing open source to proprietary software. One example of this is the topic of security and quality, where diverging assessments results in confusion. The authors have tried to be aware of the phenomenon and have rated for example published scientific articles above information from single interest organisations.

When sampling organisations for the study in Chapter 5, snowball sampling technique was applied as described in Chapter 3. In most cases, only one person from each organisation was interviewed. This resulted in a quite small population but the authors believe the selected organisations still make a good representation of commercial open source activity in Sweden.

Tetra Pak is a very large organisation divided into many departments. It is difficult to get a unified picture of how things work, and there are diverging opinions on how work should be organised. Given this, it has during the Master Thesis work been complicated to attain objective facts regarding software management at Tetra Pak. “Challenges in Software Management” presented in Section 2.4, as well as the cost analysis in Chapter 6, are mainly based on a few interviews and may be incomplete. In search of an unbiased assessment, the authors have chosen interviewees from different departments and with different professions.
7.6 Future of OSS…

Can OSS be part of a long-term strategy? It is obvious that there is a growing interest in OSS among companies. There are already products containing OSS on the telecom market as well as in consumer electronics. The automation industry, which is more conservative, is yet to discover OSS. The authors believe that when enough products are based on OSS, the concept will probably be accepted by large corporations and eventually become mainstream. More organisations providing services for OSS will form. Just as for proprietary software, it will be possible to outsource support, training and integration of OSS. This answers question number 3, if OSS can be part of a long-term strategy.

When OSS grows more popular within companies there might be a contra movement in open source organisations. FSF and other similar associations will strive to keep the underground characteristics of OSS. This is even more likely to happen if commercial organisations take advantage of community development without contributing. It is also probable that software suppliers will be increasingly affected by OSS and will have to change business strategy. Source code for commodities will no longer bring in money. Customers will become more aware of what they are actually paying for, and will not blindly buy software licences without evaluating the terms carefully.
8 Conclusions

OSS is here to stay. Interviews performed with different organisations in this Master Thesis clearly show that there is noticeable interest in OSS from different directions. Software developers see more efficient work procedures, and managers see a possibility to save money. Therefore, OSS is something to bear in mind when assessing alternative strategies for software management. Large software organisations such as IBM and Borland have been forced to take a position on how to deal with OSS.

OSS implies a different approach compared to traditional software use and development. Access to source code gives rise to new possibilities, such as freedom to make in-house modifications and further development of acquired software. However, to truly succeed with an OSS migration, there are some important factors to consider. A clear strategy for OSS introduction is required and the goals must be defined. It takes a strong and competent advocate to introduce OSS, but also support from management. Otherwise, it will be difficult to manage new ways to think and work regarding licences, community relations and the lack of traditional software vendors. One important question is to know how the organisation should cooperate with other developers in the community. In addition to this there is uncertainty when it comes to security, quality and cost estimations for OSS.

Not only pure software organisations will be affected by the evolution of OSS, organisations using OEM licences will also need to take a position. Tetra Pak must monitor the OSS movement to see how it evolves. If OSS continues to increase in popularity and gain acceptance, Tetra Pak needs to have a strategy to meet new conditions. One way to evaluate consequences of OSS use is to introduce a pilot project to replace a non-critical proprietary software component with OSS. Tetra Pak changes software platform for machines every few year, and there are always large costs involved. OSS may be an interesting alternative to consider next time a change is required. Tetra Pak does not consider software knowledge to be a core competence, and this makes it hard to be in the front line when it comes to new procedures and tools such as OSS. Before going any further in this matter, Tetra Pak employees must evaluate gains and improvements found in this Master Thesis. The authors are not familiar enough with Tetra Pak to determine the impact of different gains and risks. Only insiders are able to decide if a certain factor is more or less important in a Tetra Pak context.

During this Master Thesis work, the authors have found a number of OSS-related areas where more work is required. One example of this is the matter of security and quality. Is OSS more or less secure than proprietary software regarding viruses? Opinions diverge. Another area where more research is required is cost analysis of OSS. In a commercial context, economical factors are highly relevant and cost savings are always appealing. So far, there are very few objective studies made that compare costs between proprietary software and OSS.
References


Business Software Alliance, BSA www.bsa.org, visited 2004-12-22


Ericsson www.ericsson.com, visited 2004-12-20


Free Software Foundation, FSF, 
[www.fsf.org/fsf/fsf.html](http://www.fsf.org/fsf/fsf.html), visited 2004-10-20

Free Standards Group, 
[www.freestandards.org](http://www.freestandards.org), visited 2004-12-14


IBM, 
[www.ibm.com](http://www.ibm.com), visited 2004-12-06

Krishnamurthy S. “Cave or Community?: An Empirical Examination of 100 Mature Open Source Projects”, First Monday, Internet publication 2002. 
[www.firstmonday.org/idea.html](http://www.firstmonday.org/idea.html), visited 2004-10-26

MontaVista, 
[www.mvista.com](http://www.mvista.com), visited 2004-11-28


Nohau, 
[www.nohau.com](http://www.nohau.com), visited 2004-11-15


Open Source Initiative, OSI, 
[www.opensource.org](http://www.opensource.org), visited 2004-11-08


[www.firstmonday.dk/issues/issue3_10/raymond/index.html](http://www.firstmonday.dk/issues/issue3_10/raymond/index.html), visited 2004-12-10


Walleij, L., ”Att använda GNU/Linux”, 2004.


Wikipedia, the Free Encyclopedia.

A. Appendix – Open Source Dictionary

**BSD:** Berkley Software Distribution is a Unix-based operating system, which has forked into both proprietary software and OSS. The most common OSS distributions are FreeBSD, NetBSD and OpenBSD.

**Community:** A gathering of developers and other stakeholders for a specific OSS.

**Copyleft:** By the principles of copyright, preserve something to be free. This is the opposite of copyright.

**Dongle/Hardware key:** A small hardware device that connects to a computer and acts as an authentication key for a particular piece of software.

**EULA, End User Licence Agreement:** What you can and cannot do with a program according to the manufacturer. This is the text you usually see before clicking the ”I agree”-button when installing a program.

**Freeware:** A computer program anyone may use and distribute. Usually not distributed with source code.

**FLOSS:** Fre/Libre Open Source Software.

**FOSS:** Free and Open Source Software.

**Free Standards Group:** Organisation that works to grow the use and acceptance of OSS.

**FSF:** Free Software Foundation. Founded by Richard M. Stallman to develop and distribute Free Software.

**FUD:** Fear, Uncertainty and Doubt. According to many supporters of OSS this is the strategy Microsoft is using when acting against OSS.

**GNU:** Gnu’s Not Unix, a recursive acronym and a project within FSF with the purpose of making a free Unix-like operating system.

**GPL:** GNU General Public License, the most commonly used OSS licence.

**IDE:** Integrated Development Environment, containing a source code editor, a compiler, interpreter, a debugger and sometimes other tools to facilitate software development.

**LGPL:** GNU Lesser General Public License is less restrictive than GPL regarding Copyleft, and allows integration between OSS and proprietary software.

**Library:** A collection of software functions and/or data organised in a way that makes it easy to link to an application and thereby make an executable program.
**Link:** Computer programs may contain linking to object files. These files are executed when the program is executed. Linking can be done in two different ways, static or dynamic. Static linking means that the program contains the object file. When linking dynamically to an object file it is assumed the object file is already on a specified place in the computer. This has nothing to do with hyperlinks on HTML pages.

**Mailing list:** A special usage of email that allows for widespread of information to a large group of Internet users. In the open source projects the mailing list takes the form of a discussion list, where a subscriber can use the list to send messages to all the other subscribers, who may answer in the same way.

**MIT:** Massachusetts Institute of Technology, a respected American university, considered being in the frontline of Computer Science since early days.

**MPL:** Mozilla Public License, the licence under which Netscape released the source code to the web browser Mozilla. It is very similar to the LGPL.

**OEM:** An Original Equipment Manufacturer is a company that builds components that are used in systems sold by a value-added reseller, or VAR.

**Operating System:** A program running on a computer to make it possible to run other programs.

**OSI:** Open Source Initiative. Bruce Perens and Eric S Raymond founded OSI in 1997 as a more liberal alternative to FSF. OSI has a more pragmatic way to look at OSS.

**OSS:** Open Source Software.

**PLC:** A Programmable Logical Controller is a small computer used for automation of real-world processes.

**Public domain:** Programs, or other things such as music, without a named author and thereby free to use for anyone. It does not need to be distributed with source code.

**RTOS:** Real Time Operating System, a program running on a computer to make it possible to run other programs, dividing memory time between several running tasks, where different priorities can be defined.

**SCADA:** Supervisor Control and Data Acquisition provides information about the machine to for example a machine operator.

**Shared Source:** A new policy from Microsoft, with the purpose to give certain groups a possibility to get access to and review the source code of a Microsoft product, after signing an agreement regulating the terms.

**Shareware:** Programs you may use for free during a limited period before paying the licence fee. It may also be a reduced version of a program, where you get access to more functionality by paying the fee.
**Source code:** A representation of a program readable to humans, which is then made understandable to the machine with the help of a compiler.

**Source Forge:** Repository for OSS on the Internet.

**TCO:** Total Cost of Ownership, a cost model considering cost drivers during the whole product life cycle.

**TCP/IP:** "Transmission Control Protocol/Internet Protocol". The protocol (languages) all computers use to communicate over Internet.

**TPOP:** Tetra Pak Operating Panel, the panel from which an operator can control a packaging machine.

**XML:** Extensible Markup Language is a flexible and easy-to-understand text format based on SGML (Standard Generalized Markup Language, a format for structured text), created to produce advanced electronic publications. XML is also important when storing and exchanging data on the web.
B. Appendix – Where to Start with OSS…

…if you need a broad, easy to read, introduction to OSS in handy paperback:

…for a general introduction of different open source licences:
http://www.opensource.org/

… if you need an introduction in patent and copyright for open source:
www.rosenlaw.com/lj9.htm

… to learn how different OSS licences relates to Swedish law:

…if you want to check out current open source projects you can find the majority here:
www.sourceforge.net

…to get familiar with a large successful OSS project with many participants from software industry:
www.eclipse.se

…to learn about a different kind of OSS project that have resulted in an extensive and informative encyclopaedia, built up by thousands of contributors:
www.wikipedia.org

The European Commission has defined a set of migration guidelines, quite useful if considering using OSS:
www.netproject.com

…if you want to see how Microsoft evaluates open source:
www.microsoft.se/getthefacts

…to learn about OSS history and development from one of the founders of OSI:
www.firstmonday.dk/issues/issue3_3/raymond/index.html
C. Appendix – Cost Drivers

<table>
<thead>
<tr>
<th>Operations Cost</th>
<th>Quality</th>
<th>Logistics</th>
<th>Technological Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manufacturing</td>
<td>• Durability</td>
<td>• Freight</td>
<td>• Design Obsolescence</td>
</tr>
<tr>
<td>• Machine Efficiency</td>
<td>• Replacement</td>
<td>• Packaging</td>
<td>• Suitability for Intended Use</td>
</tr>
<tr>
<td>• Production to Schedule</td>
<td>• Field Failure</td>
<td>• Customer Service</td>
<td>• Flexibility for New Use</td>
</tr>
<tr>
<td>• Labor Savings</td>
<td>• Customer Downtime</td>
<td>• Availability</td>
<td>• Technology</td>
</tr>
<tr>
<td>• Assembly Cost</td>
<td>• Inspection</td>
<td>• Handling</td>
<td>• Changing Technology</td>
</tr>
<tr>
<td>• Operating Supplies</td>
<td>• Cost of Quality</td>
<td>• Instability in Freight Rates</td>
<td>• Long-Term Advantage</td>
</tr>
<tr>
<td>• Long-Term Operating Costs</td>
<td>• Calibration Cost</td>
<td>• Outbound Cost</td>
<td>• Supplier Ability to Change Technology</td>
</tr>
<tr>
<td>• Capacity Utilization</td>
<td>• Rework</td>
<td>• Tariffs</td>
<td></td>
</tr>
<tr>
<td>• Increase in Production Output</td>
<td>• Scrap</td>
<td>• Leadtime</td>
<td></td>
</tr>
<tr>
<td>• Equipment Speed</td>
<td>• Customer Returns</td>
<td>• On-Time Delivery</td>
<td></td>
</tr>
<tr>
<td>• Cost in Use</td>
<td>• Rejection Cost</td>
<td>• Supplier-Managed Inventory</td>
<td></td>
</tr>
<tr>
<td>• Line Speed</td>
<td>• Quality Improvement</td>
<td>• Time to Schedule</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Unplanned Downtime</td>
<td>• Warehousing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Out-of-Service Costs</td>
<td>• Duties</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintenance</td>
<td>Inventory Cost</td>
<td>Life Cycle</td>
<td>Supplier Reliability and Capability</td>
</tr>
<tr>
<td>• Supplies</td>
<td>• Safety Stock</td>
<td>• Long-Term Usage</td>
<td>• Partnering Costs</td>
</tr>
<tr>
<td>• Training</td>
<td>• Design/Procurement for</td>
<td>• Projected Life Cycle</td>
<td>• Team Costs</td>
</tr>
<tr>
<td>• Downtime</td>
<td>Inventory Reduction</td>
<td>• Life of Product</td>
<td>• Trust</td>
</tr>
<tr>
<td>• Costs</td>
<td>• Storage</td>
<td>• Life Cycle Stability</td>
<td>• Supplier Capabilities</td>
</tr>
<tr>
<td>• Labor</td>
<td>• Perishability</td>
<td>• Cost Savings over Life of</td>
<td>• Payment Terms</td>
</tr>
<tr>
<td>• Repair Costs</td>
<td>• Turnover</td>
<td>Product</td>
<td>• Supplier R&amp;D Capability</td>
</tr>
<tr>
<td>• Parts</td>
<td></td>
<td></td>
<td>• Supplier Ability to Grow</td>
</tr>
<tr>
<td>• Spare Parts</td>
<td>Transactional Cost</td>
<td></td>
<td>• Supplier Support</td>
</tr>
<tr>
<td></td>
<td>• Administration of Post-Purchase Agreements</td>
<td></td>
<td>• Service by Supplier</td>
</tr>
<tr>
<td></td>
<td>• Ease of Transaction</td>
<td></td>
<td>• Stocking at Supplier</td>
</tr>
<tr>
<td></td>
<td>• Supplier Conversion Cost</td>
<td></td>
<td>(Quantity Availability)</td>
</tr>
<tr>
<td></td>
<td>(Cost to Change Supplier)</td>
<td></td>
<td>• Familiarity with Supplier</td>
</tr>
<tr>
<td></td>
<td>• Small Orders</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Procurement</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Transactional Activity</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Long-Term Savings</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Life Cycle</td>
<td>Opportunity Cost</td>
<td>Initial Price</td>
</tr>
<tr>
<td>• Taxes</td>
<td>• Installation</td>
<td>• Cost of Money</td>
<td>• Unit Cost</td>
</tr>
<tr>
<td>• Value Chain</td>
<td>• Ease of Operation</td>
<td>• Overhead</td>
<td>• Initial Purchase Price</td>
</tr>
<tr>
<td>• Warranty</td>
<td>• Noise Level</td>
<td></td>
<td>• Long-Term Price Stability</td>
</tr>
<tr>
<td>• Product Design</td>
<td>• Technical Support</td>
<td></td>
<td>• Initial Capital Expenditure</td>
</tr>
<tr>
<td>• Availability from a Supplier</td>
<td>• Validation/Registration Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Disposal Costs</td>
<td>• Overall Competition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Liability and Indemnification</td>
<td>• Service Costs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Obsolescence Cost</td>
<td>• Disposal Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Currency Exchange Rates</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Direct Labor</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table VIII: Categorization of Identified Total Cost of Ownership Cost Drivers

85