



Graduate School of
Business Administration Zürich



PRIFYSGOL
CYMRU
UNIVERSITY
OF WALES

How OpenPKG can support IT business.

Written by:

Richard Maier
Hopfenstraße 9
84091 Attenhofen

Contents

1	<u>EXECUTIVE SUMMARY</u>	1
2	<u>GETTING STARTED</u>	2
2.1	WHAT ARE UNIX AND UNIX-COMPATIBLE SYSTEMS?	2
2.2	WHAT IS OPEN SOURCE?	4
2.3	WHAT IS FREE SOFTWARE?	5
2.4	WHAT IS FREWARE?	6
2.5	WHAT IS OPENPKG?	7
3	<u>UNIX ISSUES</u>	9
3.1	AVAILABILITY OF SOFTWARE	9
3.2	DIFFERENCES BETWEEN UNIX TYPES	12
3.3	MIGRATIONS	13
3.4	CONSOLIDATION	14
3.5	DEPENDENCY	16
3.6	SECURITY	17
4	<u>OPEN SOURCE ISSUES</u>	18
4.1	SUPPORT	18
4.2	BUSINESS CONTINUITY	19
4.3	PLATFORM	21
4.4	SECURITY	22
5	<u>THE SOLUTION – OPENPKG</u>	23
5.1	OPENPKG FOUNDATION	23
5.1.1	SOLVING ISSUES BETWEEN THE DIFFERENT UNIX TYPES	23
5.1.2	SOLVING PACKAGING	24
5.1.3	SOLVING MIGRATION	25
5.1.4	SOLVING CONSOLIDATION	25
5.1.5	SOLVING DEPENDENCIES	27
5.1.6	SOLVING PLATFORM	27
5.1.7	BUSINESS CONTINUITY SOLUTION	28
5.1.8	SECURITY	30

5.2 OPENPKG GMBH	31
5.2.1 CONSULTING	31
5.2.2 DEVELOPMENT	31
5.2.3 DEPLOYMENT	31
5.2.4 TRAINING	31
5.2.5 SUPPORT	32
5.2.6 MAINTENANCE	32
5.2.7 OPERATIONS	32
5.2.8 PROJECT MANAGEMENT	32
6 CASE STUDIES	33
<hr/>	
6.1 PORTLAND STATE UNIVERSITY, USA	33
6.1.1 IT ENVIRONMENT INFORMATION	33
6.1.2 ISSUES BEFORE USING OPENPKG	35
6.1.3 ISSUES SOLVED BY USING OPENPKG	36
6.1.4 ADDITIONAL BENEFITS OF USING OPENPKG	37
6.1.5 ISSUES AFTER USING OPENPKG	38
6.2 INTERNATIONAL TELECOMMUNICATIONS COMPANY	38
6.2.1 IT ENVIRONMENT INFORMATION	39
6.2.2 ISSUES BEFORE USING OPENPKG	40
6.2.3 ISSUES SOLVED BY USING OPENPKG	41
6.2.4 ADDITIONAL BENEFITS OF USING OPENPKG	42
6.2.5 ISSUES AFTER USING OPENPKG	43
6.3 FRAUNHOFER INSTITUT INFORMATION- UND DATENVERARBEITUNG	44
6.3.1 IT ENVIRONMENT INFORMATION	44
6.3.2 ISSUES SOLVED BY USING OPENPKG	44
6.3.3 ADDITIONAL BENEFITS OF USING OPENPKG	44
6.3.4 ISSUES AFTER USING OPENPKG	45
6.4 SUMMARY OF CASE STUDIES	45
7 EXAMPLE COMPANY - IMPLEMENTING AND USING OPENPKG	47
<hr/>	
7.1 CURRENT SITUATION	47
7.2 VISION	51
7.3 MISSION	51
7.4 SHARED VALUES	51

7.5	BALANCED SCORE CARDS	52
7.6	STRATEGY MAP	54
7.7	IT GOVERNANCE	55
7.8	IMPLEMENTATION PLAN	57
7.8.1	PROJECT PHASES	57
7.8.2	COSTS	58
7.8.3	IMPLEMENTATION	60
7.9	ISSUES SOLVED AFTER OPENPKG IMPLEMENTATION	60
7.9.1	SYSTEM UPDATES	60
7.9.2	STANDARDIZATION	61
7.9.3	AVAILABILITY	61
7.9.4	SECURITY	62
7.9.5	COMPLEXITY OF THE ENVIRONMENT	62
7.9.6	PACKAGING	62
7.9.7	DOCUMENTATION	62
7.9.8	NEWCOMER ISSUE	62
7.9.9	MIGRATION & CONSOLIDATION & PLATFORM INDEPENDENCE	63
7.9.10	SUPPORT	63
7.9.11	DEVELOPMENT	63
7.9.12	CONSULTING & TRAINING	63
7.10	SUMMARY	63
8	<u>FINAL CONCLUSION</u>	<u>64</u>
9	<u>APPENDICES</u>	<u>67</u>
9.1	TASK LIST IN-HOUSE PROJECT	67
9.2	TASK LIST WITH EXTERNAL PARTNER PROJECT	69
10	<u>GLOSSARY</u>	<u>73</u>
11	<u>FIGURES</u>	<u>76</u>
12	<u>TABLES</u>	<u>77</u>
13	<u>REFERENCES</u>	<u>78</u>
14	<u>ENDNOTES</u>	<u>80</u>

1 Executive summary

Your IT uses UNIX systems and a lot of Open Source software but is not efficient enough? You would like to change the hardware supplier of your Unix systems, but you do not know how to migrate the Open Source software to the new systems? You would like to consolidate your services and need some tools to accomplish this task?

This thesis presents and analyzes the solution: OpenPKG. In all the cases given above, OpenPKG provides tools, techniques and mechanisms to perform the required tasks in a very efficient way.

The first step is to get a common understanding of the topic, which is detailed in chapter two. The next two steps focus on the issues created by the use of UNIX and Open Source software. After you have read through it, you might like to ask yourself how these issues can be solved because some of them may apply to you. The answer to this question is: Use OpenPKG. OpenPKG as a solution is described right after the discussion of the issues.

The next question might be: Yes, that is possible, but how can it be done? And how does it work in real life? To answer these questions, this thesis contains case studies with descriptions of actual problems and how they were solved using OpenPKG. Some problems might not even have been experienced at all because OpenPKG was used.

Now you might ask: How can I implement OpenPKG to better support my business? To demonstrate this, I created an example company. This includes some financial data and, in addition, shows how OpenPKG supports IT governance and the corporate strategy.

2 Getting started

In this Chapter I will give a short introduction to the history and invention of UNIX and UNIX-compatible systems and where they are mainly used. I will also explain Open Source software and Free Software and provide an introduction to OpenPKG to have a common understanding. In the UNIX section a short introduction to UNIX will be given to illustrate how many different kinds of UNIX operating systems are available, both commercially and freely. The important LINUX distributions will be mentioned as well. I will describe where UNIX and UNIX-compatible systems are used.

In the Open Source section I will describe the idea behind Open Source and the way it works. I will explain the revision numbers, upgrade/update, and required tasks to get it compiled and installed.

In the freeware section I will point out the difference between Open Source and Freeware. I will also mention the ideas behind freeware and the reasons why it exists.

The OpenPKG section will clarify how OpenPKG.org, OpenPKG.net and OpenPKG.com are related to each other. It will also explain the purpose of each part of OpenPKG. Additionally, a few statistics about the size of the community and OpenPKG releases will be given.

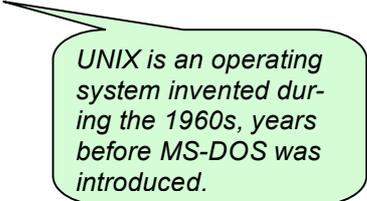
2.1 What are UNIX and UNIX-compatible systems?

The invention of UNIX started in the 1960s at the Massachusetts Institute of Technology (MIT). AT&T Bell Labs and General Electric worked on an operating system (OS) called Multics, which was designed to run on a mainframe. This was (in computer times) ages before MS-DOS was invented in 1981, which was later developed into Windows.¹

The MIT's production release of UNIX had poor performance, and AT&T Bell Labs pulled out of the development team and started up on their own. Ken Thompson, one of the developers at Bell Labs, continued to develop for the mainframe and developed a game for it, but it was slow and costly. With the help of Dennis Ritchie, he rewrote the game in a different language, 'C'. The experience of rewriting and his work on the Multics project led Thompson to start a new operating system.²

A team of developers including Rudd Canaday and led by Thompson and Ritchie started developing a new multi-tasking operating system and file system. The project was called Unics (**U**niplexed **I**nformation and **C**omputing **S**ystem) and could already support two simultaneous users. Later on the name was changed to UNIX and, thus, a legend was born. This was the first time the operating system UNIX was called UNIX and was installed in 1969 on a PDP-7 in a single-user version.³

In the 1970s UNIX was further developed and written in a programming language called C. At that time UNIX was only used internally at Bell Labs. 1977 was the first time UNIX



UNIX is an operating system invented during the 1960s, years before MS-DOS was introduced.

was ported to a non-PDP machine. This was the starting point to port UNIX to a large number of systems quite quickly.⁴

Figure 1 illustrates how UNIX was developed further and at what time the different kinds of UNIX split from the original invention. From 1977 until 1981, UNIX was mainly used at universities. Due to the fact that a lot of students used it at the university, UNIX got broader recognition, especially when students started working.

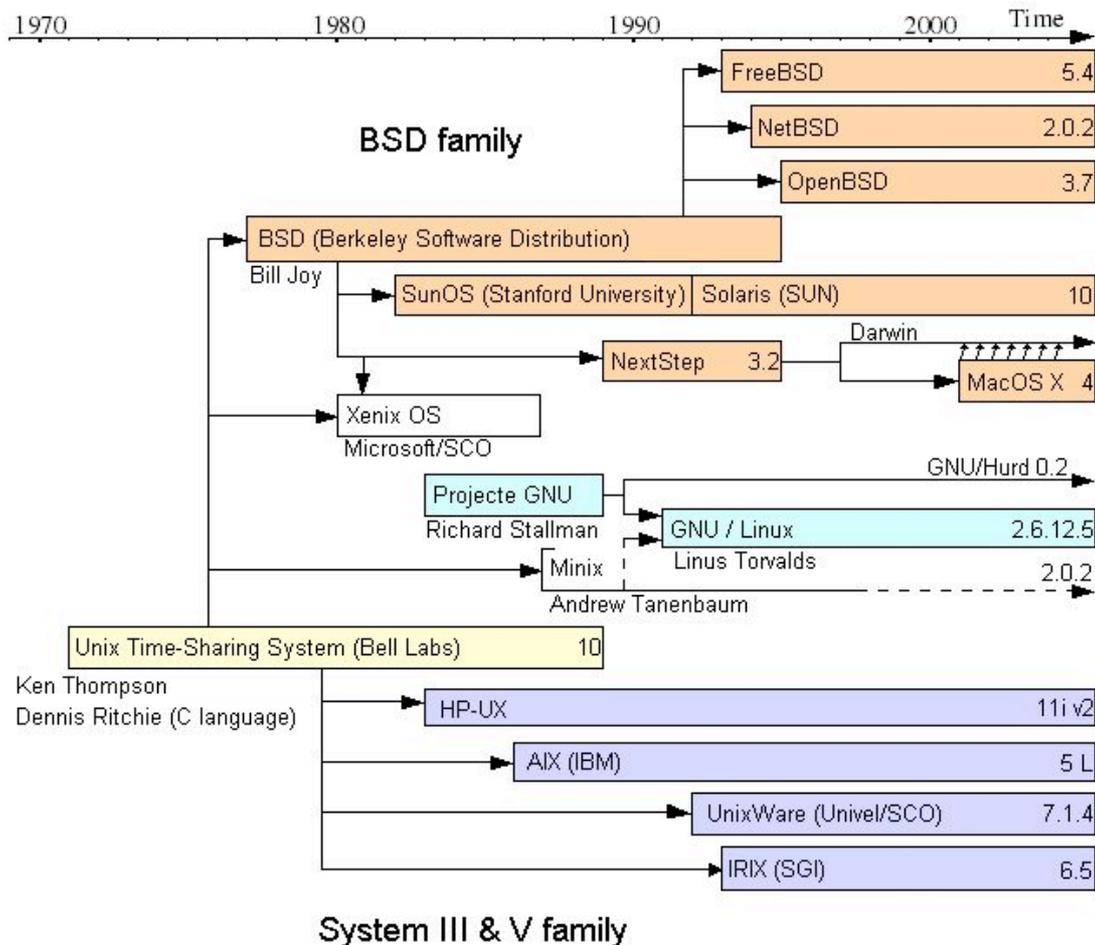


Figure 1: Filiations of UNIX and UNIX-like systems⁵

Figure 1 additionally depicts the many splits of Unix since the beginning. It also shows the entry of Linux, which was invented by Linus Torwalds and represents one UNIX-compatible system that is very popular at the moment.

Today there are two different kinds of UNIX and UNIX-compatible systems, namely commercial and non-commercial. UNIX is mainly used in a datacenter environment, where reliability and availability of services are very important for the company business. There are many different kinds of UNIX and UNIX-compatible systems available. I would like to concentrate on those kinds for which OpenPKG is available. The following table provides an overview of the different kinds of UNIX supported by OpenPKG at the time the

Since the invention of UNIX many different versions have been developed and are now available commercially and non-commercially.

document was written and on the operating systems for those OpenPKG received positive feedback from the users.

Commercial	Non-commercial
Sun Solaris (SUN Microsystems)	FreeBDS
RedHat Enterprise Linux (RedHat)	NetBSD
Novell SUSE Linux (Novell SUSE)	Debian-Linux
AIX (IBM)*	Fedora-Linux
HP-UX (Hewlett Packard)*	Gentoo-Linux
IRIX (SGI)*	Mandriva-Linux
MAC OS X (Apple)*	

* OpenPKG received positive feedback from the OpenPKG user for the specified OS

Table 1: Supported Operating Systems

2.2 What is Open Source?

Open Source software is created and governed by a license that enables all users to receive a human readable version of the software, which is called source code. The basic idea behind Open Source is that software evolves when programmers can read, modify and redistribute the source code. People around the globe can improve, adapt, and fix bugs, and this happens at speed. From the Open Source perspective, conventional software development seems rather slow.⁶

The Open Source community has learned that fast and evolutionary software development produces better software than traditional ways of development. This is because more people have a look at the source code, more ideas are introduced, and each person brings his or her own unique interests and competencies into the development process.⁷

Open Source software not only allows access to the source code, it also permits everyone in the world to improve and change it by fulfilling the terms of the original software.

A group of individuals presented the term Open Source to re-label free software as Open Source software in 1998. Eric S. Raymond was a member of that team and he coined the term Open Source.⁸

Open Source software must comply with the following distribution terms.

- **Free Redistribution:** The software must be redistributable without having to pay any royalty fee and shall not restrict any party from giving away the software.
- **Source Code:** The software must include the source code and must allow its distribution. If a program gets distributed in a compiled form and no source code is delivered with it, it must be well published where the source code can be obtained without charge. Hiding and intermediate forms of source code are not allowed.

- Derived Works: Modifications and derived works must be allowed by the license and it must be allowed to redistribute the software under the terms of the original software.
- Integrity of The Author's Source Code: To distribute source code in a modified form may be restricted by the license, but the license has to allow the distribution of "patch files" with the source code for the purpose of modifying at build time. It explicitly has to permit the distribution of the software built from a modified source code.
- No Discrimination Against Persons or Groups
- No Discrimination Against Fields of Endeavor: The license of the software must allow anyone to make use of the program in a specific field or endeavor.
- Distribution of License: The rights attached to the software must apply to all software that is distributed without requiring an additional license.
- License Must Not Be Specific to a Product: The rights attached to the software apply to the software as a whole and all specific parts taken from that software. This means that if one part of a piece of software is taken and used elsewhere, the rights from the original software apply to it.
- License Must Be Technology Neutral
- License Must Not Restrict Other Software: It is not allowed to place restrictions on other software by the distributed one.⁹

The fact that Open Source is free in its redistribution does not mean that it is free of charge completely. The software might be available as Open Source and might be free in its redistribution, but you might have to pay to use it. It can also be free to use under certain circumstances, but not for others. Most of the time it is free, but it is wise to have a look at the license agreement to see what is stated there and what applies to your specific application.

2.3 What is Free Software?

Free Software is defined by the Free Software Foundation (FSF). Free Software is software that can be used, copied, studied, modified and redistributed without restriction. The central point of Free Software is the freedom from restrictions. The usual way to distribute software, i.e. as Free Software, is to distribute it with a Free Software license and the source code has to be made available.¹⁰ The opposite of Free Software is proprietary software, which is protected and sold for profit.

Richard Stallman, the founder of the Free Software movement, introduced a Free Software definition: "Free Software is a matter of liberty not price. To understand the concept, you should think of 'free' as in 'free speech', not as in 'free beer'". This means that computer users have the freedom to cooperate with whomever they like and are able to control the software they are using.¹¹

Free Software is free of speech (libre software), but may not be free in terms of cost.

Debian, a distributor of software, rewrote the definition of "Free Software" from FSF, to which the term Open Source is now attached. This ended in a situation where today almost every Open Source program is Free Software as well, but there are some exceptions. The big difference between Open Source and Free Software is that the main

focus of Open Source is on the availability of the source code, whereas the basic idea of Free Software is freedom, as already stated.¹²

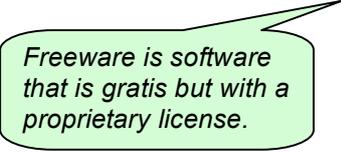
From a technical viewpoint Open Source and Free Software are more or less the same. Richard Stallman also coined the term FOSS, which means Free and Open Source Software and implies that both licenses apply to the software detailed in this chapter.¹³

Now you know the reason why there is Open Source and Free Software, the ideologies have a different understanding of software and its distribution.

2.4 What is Freeware?

Freeware is software that is made available free of charge but in general is proprietary. The users may not have the freedom to use, copy, study, modify or redistribute it.

Therefore, the software is gratis but is not free in sense of 'free in speech'.



Freeware is software that is gratis but with a proprietary license.

Freeware has its roots in the times of shareware. In those days shareware was made available free of charge, but you had to pay for it when you were using it. There is still a considerable amount of shareware available, and you still have to pay for it. If you choose not to pay, you may be running the software illegally. The author of the software might well decide not to sue you because first, he has to find out who is using it illegally and second, to sue the user costs time and money. However, the use of the software is still illegal. Freeware increased because it was software that was made available for free and it was cost free to use.

Freeware, first of all, has nothing to do with Open Source or Free Software at all. The fact that Open Source is often called Freeware is because it is often available for free and often is cost free to use. Unlike Freeware, cost is not the driver for Open Source or Free Software because the requirements of both can be fulfilled and it can still cost money. However, at the present time this is not very often the case.

Andrew Fluegelman coined the term Freeware and distributed his software under that term. The umbrella term Freeware includes many variations of the Freeware model, which are:

- **Loss leader:** Mostly used by commercial vendors to attract customers to a product or service that costs a fee.
- **Adware:** It requires the user to view advertisements to use the software. Spyware is often adware.
- **Donationware:** The author of the software asks each person using it to make a donation to the author or a third party.
- **Abandonware:** This is commercial software that has not been sold for a long time or the copyright holder is defunct. Most of the time it requires a payment and forbids its redistribution.
- **Postcardware:** This is mainly freeware, but the author asks the user to send him a postcard where feedback is provided and to thank him for the software.
- **Baitware:** This is defective or very limited freeware software to attract the users to use the commercial product.¹⁴

Now you know the reasons why there is Open Source, Freeware and Free Software, the ideologies have a different understanding of software and its distribution and costs.

2.5 What is OpenPKG?

When having a look at the Internet you might be confused because you will see three different web sites of OpenPKG. Before we start with OpenPKG, I will explain the different versions and clarify why they are there.

- [OpenPKG.org](#) is the project that provides rules and the results. Ralf S. Engelschall and Thomas Lotterer started it in 2000. To start such a project a place is required to make other people aware of it. This is important because such a project cannot be done by one person as it requires contributors – who founded the OpenPKG Foundation e.V in 2005.¹⁵
- [OpenPKG.net](#) is the OpenPKG Foundation e.V. and it was formed by the very committed contributors in order to bundle the social network and computing resources. They work on OpenPKG on an unpaid basis, purely because they believe it is a great idea and because it helps them improve their business. The foundation is required to have sponsors and to ensure the provision of hardware support for the infrastructure. Otherwise sponsors would have to give their donations to individuals, who then need to go through the German tax department, thereby reducing the money available for the infrastructure.¹⁶
- [OpenPKG.com](#) is the commercial part of OpenPKG (OpenPKG GmbH) and was founded in December 2005. It was founded to provide commercial support and add-on services to OpenPKG business customers. The services offered by OpenPKG GmbH will be described in section 5.2 OpenPKG GmbH.¹⁷

[OpenPKG.org](#) –
the Project
[OpenPKG.net](#) –
the foundation
[OpenPKG.com](#) –
the commercial

As you can see, the foundation and the corporation do support each other. The foundation makes people aware of additional support if they require some kind of assistance. The company supports the foundation by improving the chances of getting new sponsors and contributors.

The idea behind OpenPKG is to create an environment with cross-platform and multi-instance functionality, to package Open Source Software for UNIX systems and deploy them very easily on different systems. It acts on top of the operating system (UNIX and UNIX-compatible) and underneath the application (Open Source or Free Software). It is available for UNIX operating systems, as I already mentioned in chapter 2.1. OpenPKG currently consists of about 950 freely available Open Source packages.¹⁸

OpenPKG is a facility that provides cross-platform, multi-instance, packaging and deployment functionality and has two branches and uses the FOSS licensing model.

There are two different branches of OpenPKG software, a current version and a release version. In the current version the latest software packages are available, and at a specific point in time the snapshot will become a release. The second type is the release version. It gets released at the end of a development cycle – to get user feedback before releasing it and to provide a stable version for production environments –

which is usually four to six months. While working on the next release, security updates and other important patches get applied to the former two releases.

The licensing model of OpenPKG is a FOSS licensing model. The Open Source software development model consists of very fast Internet driven infrastructure, a central and publicly available source code repository, the worldwide distributed OpenPKG software and developers who do their best to make OpenPKG successful.

The following pictures illustrate the difference between installing Open Source Software straight onto a system and how it is handled with OpenPKG.

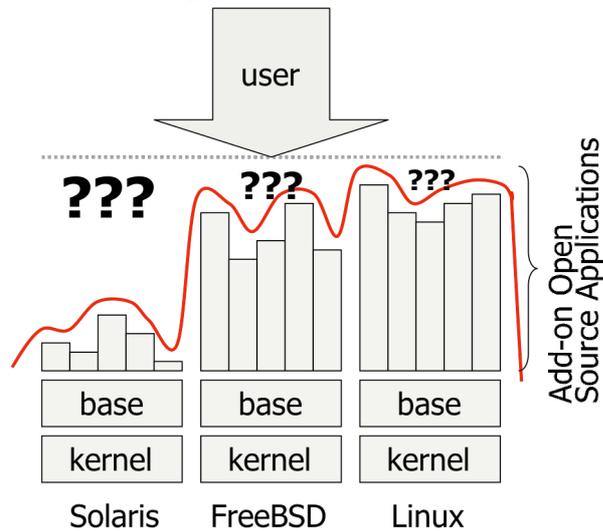


Figure 2: Open Source Software installation without OpenPKG¹⁹

In Figure 3 OpenPKG is aware of all of the installed packages and files related to the package that are under OpenPKG control.

As you can see from Figure 2 the installed applications are far less organized and you do not know exactly what is installed on that system or where it is installed. I will discuss these issues in the following chapters.

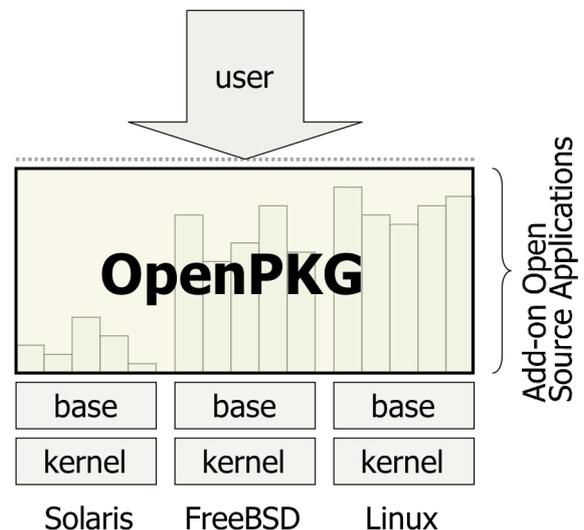


Figure 3: Open Source installation with OpenPKG²⁰

3 UNIX issues

In this chapter I will discuss a number of UNIX issues beginning with the drawbacks, which exist because of the design of the different kinds of UNIX and UNIX-compatible systems. Secondly, I will discuss the problems regarding the availability of existing software for UNIX and UNIX-compatible systems. Thirdly, I will explain the differences between the UNIX systems and how these issues need to be resolved for the use of Open Source software. Fourthly, I will point out the problems that might occur during migration or consolidation and what kind of dependencies exist and need to be solved. Finally, there are also some security issues that can surface during the use of the software.

In each section I will begin by describing the problems and how they might be solved in a common way. I will also try to point out what kinds of resources (money, manpower, time, hardware, etc.) are required to solve the drawbacks. Additionally, in each subsection I will point out how management decisions can hinder or support these situations.

3.1 Availability of Software

There is quite a lot of software available for UNIX and UNIX-compatible systems. In my opinion this kind of software can be split into three major branches, commercial software, Free and Open Source software, and home-grown software.

By having a look at the Internet, you will find out that there is a large amount of Open Source software available that might be capable of solving any problems you may have. Not making use of the whole bandwidth of available software might slow down your business, meaning that you may have to wait too long until a different solution to the problem is found and implemented. By allowing the IT department to use commercial, Free and Open Source, and home-grown software internally, the company can enjoy several benefits from it. To help you in making this decision, I will point out the advantages and disadvantages, which are based on my own experience.

There are three major branches of software: commercial, free and Open Source, and home-grown software, and not making use of the whole bandwidth might slow down your business.

Commercial Software

Advantages

- Commercial support is available.
- The development of the product will continue.
- At a small company software problems might get fixed very quickly.
- It supports the business.
- Software comes in a packaged format and only needs to be installed and configured.

Table 2: Commercial Software - Advantages

Disadvantages

- Software needs to be purchased.
- You might have to pay royalties on a monthly or yearly basis.
- Most of the time it cannot be built to fulfill company requirements.
- The production company can only fix software problems.
- At a large company software problem solving might take a very long time.
- No access to the source code.
- If it is a small company, there may be some uncertainty as to how long they can continue in business.
- Only a few people can work on the source code, which may end up leading to slow problem fixes and a minimum of improvements to the software.
- It might only be available for one set of hardware and operating system.
- If you don't have the option of an alternative piece of software, you have to live with the problems of that software.

Table 3: Commercial Software - Disadvantages

Free and Open Source Software

Advantages

- Most of the time there is no need to purchase it.
- On the whole no royalties need to be paid.
- You can fix software problems yourself.
- Solving software problems will be done quickly because of the large community of users.
- Access to the source code.
- Further development can be done yourself as long as the programmer no longer volunteers, or if you would like to contribute to the project yourself you can as well.
- Many people can work on the source code because it is available to everyone in the world.
- Flaws and bugs get detected quickly. The reason being that more people are using it because they don't have to pay for it.
- Flaws, bugs and security issues get fixed very quickly because of the large community of users.
- Your further development and enhancements of the software might get incorporated into the next version if you send them to the project owner.

Table 4: Free and Open Source Software - Advantages

Disadvantages

- To improve the likelihood that the changes you made to the software get incorporated into the next version, the project owner needs to be informed.
- Changes only get incorporated into the next version if the project owner thinks that the change is useful.
- If the change does not get incorporated, you have to adjust the changes you made and are required to do so for all the upcoming releases again and again.
- The software most of the time comes as source code and needs to be compiled before it can be installed and configured.
- While compiling the software there may be a problem that needs to be solved before the compilation can be finished. This may take some time depending on the experience of the engineer.

Table 5: Free and Open Source Software - Disadvantages

Home-grown Software**Advantages**

- No need to purchase it.
- No royalties need to be paid.
- Access to the source code.
- You can do further development.
- It could be one of your competitive advantages.
- You are taking care of what will get implemented.

Table 6: Home-Grown Software – Advantages

Disadvantages

- Only you can fix your own software problems.
- Solving software problems depends on the availability of resources within the company.
- Only people in your enterprise are able to work on it.
- Flaws and bugs only get detected by the people using the software, meaning it may take a while before a bug is detected, which can cause problems for the business.
- Flaws, bugs and security issues get fixed at a rate dependent on the available resources.

Table 7: Home-Grown Software - Disadvantages

The software used by a company depends on the strategy of the company and on the IT governance decisions. The decision about the kind of software used belongs to two

parts of the IT governance decision-making process. One part is the IT architecture decision and the other part is the business application needs. Depending on how strict the architecture decisions are made – that is, where the standards are defined – it will end in an easier or more complicated implementation to accomplish business application needs.²¹

The decision what kind of software to use in the company is part of the IT-Governance, and the decision-maker at that level has to decide if the company can afford not to use the whole bandwidth of available software.

If a company declares that Open Source is not in any regard company standard – because you can only get a minimum of support, people working on the project are volunteers, and there is always the uncertainty of whether the project will continue or not – then the engineers cannot use the software. This brings me to the point that one of the most important elements of a successful IT investment is to ensure that the enterprise is spending its money in the right place to reflect strategic priorities. The investment

process has to fulfill the requirements of a specific business unit, as well as the needs of the whole enterprise. If the company additionally has the strategic goal that they would like to reduce the amount of money spent on software, then there might be a conflict with the previous decision – as pointed out in this paragraph. Both decisions together may prevent people from reaching the goal.²²

By conducting a value benefit analysis each advantage and disadvantage can be ranked in terms of its value to the support of the business. You can also add those advantages and disadvantages that you think are missing in the above list. At the end you will see what branch of software has the most, medium and least effect on your business.²³ One should keep in mind that you have ranked the branch that only helps you decide whether to use Open Source software inside the company or not. I have picked the point of using Open Source because there are many companies where the use of Free Open Source software is not allowed. If you decide that you do not wish to use any kind of Open Source software, then you might have to reinvent the wheel if the piece of software that solves your problem only exists as Open Source.

In a day-to-day business the environment requires further development to remain competitive. The use of Open Source can help to get new support tools developed and implemented quicker than other companies. The fact that solutions get implemented quicker might speed up your business because your reaction to the market can be faster than others. The key word is speed; some companies might have to speed-up if they wish to remain competitive.²⁴ At that point the use of Open Source becomes a competitive advantage.

3.2 Differences between UNIX types

Many people think that UNIX equals UNIX, but that is not true. Most of the time there is a big difference and this is because different developers develop the software. Different application requirements and different hardware is another point. This all means that Open Source software requires adjustment for each platform in order to be able to work like it was designed.

There are several differences that should already be considered during the decision-making process. Some important ones are good support, portability, availability of applications and engineers to support it.

For most of the non-commercial operating systems the majority of Open Source software is available. The reason for this is that most of the non-commercial operating systems are Open Source as well. Porting Open Source applications between these operating systems is a bit easier than for commercial ones, because the major part of Open Source software gets developed on non-commercial operating systems (Open Source), where you have access to the source code.

To run Open Source software on commercial operating systems is a bit more complicated. Before you start with the installation, you have to resolve all dependencies of the software. This will be discussed in more detail in section 3.5. Then the software needs to be compiled. After this is successfully completed the software can be installed. In most cases self-compiled software at the end of the compilation process will not be packaged and the installation will be done by copying the software and related parts to a specific location. This is a minor problem when installing the software but a major one for de-installation because you might not know exactly where the files are located. Commercial software very often gets delivered in a packaged format; otherwise you will face the same problems during a de-installation.

One of the biggest differences in systems is configuration files which are stored in different locations. Another difference is that some commands for system configuration are not available, whereas others are. Another point of commercial UNIX is that you do not have access to the source code most of the time. This requires you to wait until an upgrade or fix for a problem becomes available. For commercial operating systems you can get support, and most of the time this support is good as well. The price for the commercial operating systems typically depends on the hardware (number of processors) on which the software is running.

There are many factors that require analysis, and you have to rank each factor accordingly. Within the ranking you should also consider what kind of UNIX you are already using, how experienced your engineers are in the different kinds of UNIX, how much money you can afford, what kind of application you would like to run on it and if it is available, installation costs and, most importantly, costs of operation.

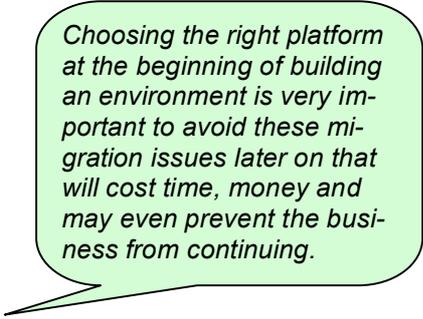
During the live cycle of an IT application only 20% of the costs occur at the initial development phase. The other 80% can be attributed to the integrations and operating stages.²⁵ This means that money is being wasted if the system does not provide a value to the company above its costs.

3.3 Migrations

Due to the described differences between the different UNIX systems, a decision-maker might think about business continuity. You could argue that this is no problem because we are using Open Source operating systems, and for most of those operating systems, the majority of Open Source software is available already in packaged and source code format. If one distribution is no longer offered, we just use a different one. Although it is true that you can just use a different distribution, during the migration you might have to adjust all configuration files because the installation path might have changed. Some of the distributions install software under `/usr/local`, some under `/opt/local` or straight in `/usr`, and there are many other options. In the case that you have to change the distribution depending on the amount of servers, some money is involved.

For commercial operating systems it is even worse. For many of those the required Open Source software is only available as source code and needs to be compiled before you can use it. This can get very complicated, because you might require an engineer who is able to make the required changes to the source code. The path might change as well, because during the initial installation the engineer might have not thought about exchanging the operating system. Now the formerly used installation path is used by the operating system for other software that should not be mixed with other applications. Some commercial operating systems do not have a compiler included. In such cases you might have to buy one or find a different solution. For some other commercial operating systems, packaged Open Source software is available but with limited features. This may obviously be a problem if you require those features. Quite often software does not require a specific installation path, which is exactly the issue. If software gets installed in an unstructured way, you do not know where it is located on a specific system, and this is common without a packaging mechanism. The next point is that applications can be mixed up, and it is not defined anywhere which file belongs to which kind of software.

Such situations can happen whenever you have to migrate your applications to a different or newer platform (hardware and operating system). As can be seen, many complications can occur during a migration independent of whether the operating system is Open Source or not. With commercial applications it can happen that the application is not available for a different operating system at all. The companies using such software have to wait until the application supplier has ported it to a different operating system. From the issues pointed out, you can see that it is very important to choose the right platform at the beginning to avoid later migrations.



Choosing the right platform at the beginning of building an environment is very important to avoid these migration issues later on that will cost time, money and may even prevent the business from continuing.

3.4 Consolidation

Consolidation is when you centralize services or when you run more services on one unique server. Consolidation of services can happen for several reasons and I will now discuss a few of them.

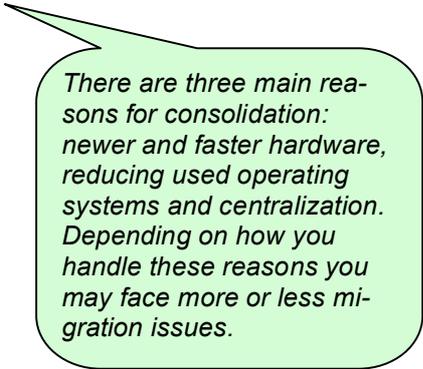
One reason for consolidation is the availability of newer and faster hardware. Some causes that support this reason are:

- Saving power and air conditioning costs in the datacenter due to the reduction in installed equipment.
- Saving support contract costs for the servers as less servers equals less costs.
- Increasing efficiency as only a minimum number of servers are required to be maintained for the business of the company.
- One server can handle more than one service because with one service the server might only be utilized by 15%.

In these cases you are required to do a migration to the new equipment. As the word migration already indicates, you might face the issues pointed out in section 3.3.

The next reason is that a company might decide to reduce the operating systems in use within the company. This can happen because:

- One used operating system is no longer available (bankrupts, no volunteers any more).
- The operating system supplier changed its licensing strategy and you do not wish to pay for it.
- Increased prices that you cannot afford.
- You would like to reduce system architecture complexity for the engineers, meaning there is no longer a need to have knowledge in many operating systems and this can drive down costs of new employees.
- You would like to increase efficiency, which can be achieved by reducing the amount of administered servers.
- You would like to have a single point of contact for your support contracts, which is almost impossible or very expensive if you are using different operating systems.



There are three main reasons for consolidation: newer and faster hardware, reducing used operating systems and centralization. Depending on how you handle these reasons you may face more or less migration issues.

In all of these cases you have to migrate the applications that are based on an operating system that you would like to reduce. Again, you might face some of the migration issues pointed out in section 3.3.

Another reason for consolidation is if a company would like to consolidate most of its servers into a centralized datacenter because:

- It should be administered centrally where the servers are located as well.
- The servers should be located in a secured area.
- There are free resources available in your datacenter and in other areas you require them for different things.

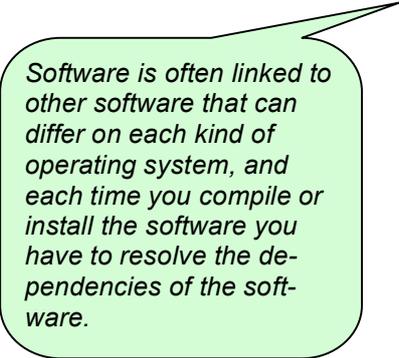
If you are just moving boxes from one place to another, you might have a minimum of tasks and problems, but you might have a long downtime for the services if both places are far away from each other. If you would like to prepare a server in your datacenter where almost everything is configured like the one you would like to consolidate, you might have a large number of tasks to fulfill (installation, configuration, testing) before you can start with the consolidation. Even taking care of all of this, you still cannot be one hundred percent sure that everything will work the same as before. It is more or less the same as for a migration.

Running different services on a server is not a problem. You might just face the problem that different departments responsible for different kinds of services cannot administer their services without conflicting responsibilities. You might also not want to give each administrator “root” access, rather to administer the service for which they are responsible, and you can by using the program “sudo”. With this program you can grant “root” access to those commands for which they require it. All other access requirements can be done with file system rights. Just assume the amount of “sudo” entries that are required and how much time is required to implement such changes beforehand to ensure that everything works fine after the consolidation. You also have to think about maintaining the “sudo” configurations.

These are real problems that can happen in a UNIX environment. Often the management only decides to consolidate because this will allow some reductions in headcount without seeing the consequences that will actually result. In my opinion, it is very important when you are consolidating that you have a more efficient environment afterwards. This includes a well-staffed team with a good selection of engineers and an optimum amount of hardware and software.

3.5 Dependency

Most of the programs used with UNIX have some dependencies on other programs or libraries. To ensure that the program is running well and as it was designed, all pre-requirements for the program have to be fulfilled. This should already be done before installing the program, because it could be that the installation process consists of some



Software is often linked to other software that can differ on each kind of operating system, and each time you compile or install the software you have to resolve the dependencies of the software.

mechanisms to create links to other programs. If you have to compile the program before you can install it, then it is even more important to have the correct libraries and programs installed. Otherwise the compilation process fails or links to the wrong software and the program does not work well.

For example, in a software development company it is very important to have a framework you can trust and that fulfills your requirements. If you do not have such a framework, you might have more bugs in your software. This could happen because one kind of UNIX requires you to link to

software ABC and another to link it to XYZ and a third to somewhere else. Therefore, if you would like to do software development for more than one platform, it is very important to have a framework that is the basis for your development and is equal on each kind of UNIX. Not having such a framework will increase development costs. The reasons for this are that you will require some quality management for each kind of UNIX, you will need engineers trained in each kind of UNIX you are producing software for, and you will require a development environment for each operating system that you are developing.

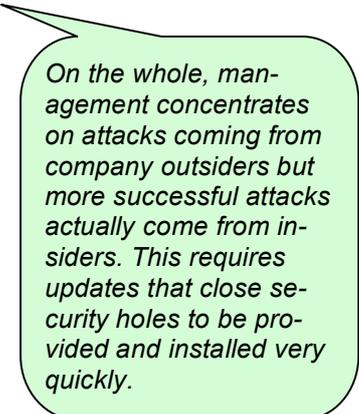
You will face these issues more often if you are using Open Source software. The reason for this is that it gets updated quite often. If it is not necessary for you to be up-to-date all the time, then this is a minor issue. For the commercial operating systems you have to install updates – provided by the operating system supplier – because if you have a support contract with the supplier and you have a problem with its software, on the whole, they ask you to install the latest version otherwise the level of support would be limited.

There are possibilities how management can support their engineers. One such possibility is to ensure the business gets supported in the best possible way, the management should already be thinking about the consequences of their decisions. When thinking about the consequences they should think about the entire system and not just about cause and effect. Another way to support the engineers is to provide them with the tools they require to fulfill the necessary tasks. This enables the engineer to fulfill the assignment in a shorter period of time, in the best possible way, and with a sense of motivation as well.

3.6 Security

The security for a system is very important. There are two kinds of attackers, insiders and outsiders. The real problem is that inside attackers cause major damage to the organization. Insiders are in a much better position than outsiders because they are already inside the company, have an account, and have access to several applications etc. In the majority of cases, the management underestimate the risk posed by the insider. More successful attacks come from the inside than from the outside and they pose greater risks as well.²⁶ In other words, you do not only have to secure servers placed on the Internet, you also have to have your internal servers even more secured than the external ones. This applies for all kinds of servers, independent of whether the software is Open Source or not.

Due to the fact that you have to keep your IT-environment up-to-date for security reasons, you require security updates very quickly after the public becomes aware of the vulnerability. For commercial operating systems and applications, you have to wait until the supplier has developed an update that closes the security gap. During that time you only have the possibilities of shutting down the affected parts to ensure data safety, to continue without security incidents and hope nothing goes wrong, or to implement a workaround.



On the whole, management concentrates on attacks coming from company outsiders but more successful attacks actually come from insiders. This requires updates that close security holes to be provided and installed very quickly.

Some of the vulnerabilities will never get fixed in the release you are using because the supplier thinks that this is only a minor problem that will be fixed in the next release anyway. If you would like to get rid of the problem, you have to upgrade to the latest release of the software, either now or at the time it becomes available. This can cause problems with other applications that may require an upgrade as well.

With Open source it is a bit different. In the case a security weakness becomes known, under normal circumstances the update will be made available very quickly. At the very least a patch fixing this issue will be made available very quickly. If there is no package available, including the patch to update the system, the IT engineers of a company can apply the patch to the source code of the software, compile it, and update it afterwards. In the very seldom case that a weakness does not get solved, the engineers still have the possibility to fix the problem by having a look at the source code and making the required changes. Due to the fact that everyone around the globe can have a look at the source code, issues with Open Source software get detected very fast. This also applies to all kinds of source code problems that can occur.

The costs of security depend on the security level that should be or is applied to the IT environment. Costs are one of the biggest concerns of management. If the management is considering whether to spend money on security or not, they should really think about what can happen to the company data and what harm will be caused to the company if a security breach should occur.

4 Open Source issues

In this chapter I will point out the issues concerning Open Source software and I will mainly focus on the Open source applications because the Open Source operating systems have been discussed sufficiently – for the purposes of this document - in the preceding sections. I will point out supporting issues, business continuity issues, platform issues, and security issues to do with Open Source. I will explain how these issues are handled in a daily business. In addition, I will mention the risks that management might face and should consider in regards to the IT-strategy.

4.1 Support

As already mentioned a few times in the previous chapter, support for Open Source operating systems is an issue. This applies to Open Source applications as well. If you require support for Open Source applications, it is very difficult to get it because there are so many uncertainties that a supporting company can face.

It is very difficult to get support for each piece of software used in a heterogeneous environment from one supplier. Reducing the heterogeneous environment makes it easier to get it supported.

A unique Open Source application can be installed on several operating systems. It can also be compiled with several required features. Most of the time those features depend on other software that is linked to the application during the compiling phase. There can be 20 or more features for one application, which can either be enabled or disabled. This means there will be about 400 different combinations because some of them already depend on each other. There

are also different distributors that offer the software as a package. Assuming there are about 20 different distributions available and that you can use the source code and compile it yourself, this equates to about 8000 different possibilities for just one application.

To support Open Source applications the supporting company requires very experienced engineers. Otherwise they can only support one application on a unique operating system. The same applies to the company if they would like to support all their applications by themselves. In a situation where the company does support itself, they may have a problem in a case where the engineers are not able to solve the application problem. The only solution would then be to get an external engineer who is a real expert in solving that particular problem. This can cause delays in finding the solution to the problem because internal processes for requesting the engineer have to be fulfilled, he might be booked out, or some other preventive factor.

Using different Open Source applications to offer the same kind of service (e.g. http) is very expensive to operate and requires an immense amount of resources. To control costs and resources and to use them in a wise manner, you should use a minimum of different operating systems and applications. You should also decide what kind of Open Source application is to be used for what kind of service. In the end there should be an

environment that fulfills the requirements of the company, supports the business, and allows further development through a minimization of costs and a minimization of resources required for the operations and development. It should also speed up the time to market and support the customer needs as well as possible in order to get the company in a better market position than its competitors.

A good solution would be an environment that allows the company to use different Open Source applications on different operating systems that are easy to install, configure and maintain and which get supported as well.

4.2 Business continuity

Business continuity in relation to Open Source is another important point that needs to be discussed. Some of the advantages and disadvantages of Open Source software, which have already been mentioned in section 3.1, also apply here. I think in the case of business continuity the issues can be seen in two ways, either as a risk or as an opportunity.

Business continuity is very important for a company and this should already be considered in the decision of how the company would like to use Open Source software.

NR	Issue	Seen as Risk	Seen as Opportunity
BC-1	Most of the time there is no need to purchase it.	Nobody pays the developers for further development, there is the risk of further development and there is uncertainty as to whether the software will cost money later on.	The business can continue without incurring high costs.
BC-2	Most of the time no royalties need to be paid.	This can be viewed the same as the risk in BC-1.	This can be viewed the same as the opportunity in BC-1.
BC-3	You yourself can fix software problems.	To fix software problems the company requires engineers who are able to fix these problems. However, they are often very expensive, and who will fix the problem if the engineer is not available?	In the case of a failure the company is able to fix the software problem themselves or they can hire an external resource to do it for them.
BC-4	Solving software problems will be done quickly.	The volunteering project team is no longer willing to work on the software to improve it, is lazy, or on vacation.	There is a big community meaning that under normal circumstances, there will be a volunteering project team working on it.
BC-5	Access to the source code.	Having access to the source code is a problem because a "bad guy" can add some miscellaneous code, which can cause security issues and other problems.	Having access to the source code allows you to make changes, and you can see if there is any miscellaneous code in it.
BC-6	Further development can be done by you if the programmer no longer volunteers. Additionally, if you would like to contribute to the project you can as well.	If the team of volunteers is no longer willing to volunteer, you have to further develop the software yourself if you intend your business to grow. Further development of the software can only be influenced minimally.	If the team of volunteers is no longer willing to volunteer, you can take over further development of the software. If you would like to help in the further development of the software, you can do so as well.

NR	Issue	Seen as Risk	Seen as Opportunity
BC-7	Many people can work on the source code because it is available to everyone in the world.	Due to the fact that so many people are able to work on the source code, it could be that release cycles are too fast. There is the saying: "Too many cooks spoil the broth."	Due to the fact that many people can work on the source code, it can be further developed very quickly.
BC-8	Failures and bugs get detected quickly.		This is good because often they get fixed before you even realize that there is a problem with the software, and all you have to do is update it.
BC-9	Failures, bugs and security issues get fixed very quickly thanks to the big community of users.	This is a serious problem because there are people who like to attack systems. In a situation where you do not have the resources to update the software or the update has some strange side effects, an attacker can use this knowledge against a system to break in.	Security holes also get fixed very quickly in order to secure the system before an attacker gets in.
BC-10	Further development and enhancements of the software might get incorporated into the next version if you send them to the project owner.	Further development and enhancements of the software done by the company have to be sent to the project owner and he then decides whether ideas get implemented or not.	Further developments and enhancements to the software done by the company can get incorporated into the next version of the software.
BC-11	Changes only get incorporated into the next version if the project owner thinks that the changes are useful.	If the change does not get implemented, you have to adjust the changes you made and this is a repeated requirement for all upcoming re-releases until you find a better solution.	You can adjust the changes to the newer version of the software. If you dislike the further development of the software, you can start your own breakaway from the software and develop it further in whichever way you like. You only have to make sure that you fulfill the license terms.
BC-12	The software mostly comes in the form of source code and needs to be compiled before it can be installed and configured.	While compiling the software there might be problems that need to be solved before the compilation can be finished. This might take some time, depending on the experience of the engineer.	This is good because it makes you aware of the dependencies of the software. Therefore, you have no need to be afraid afterwards if you have forgotten something.

Table 8: Business continuity - Issues, Risks, and Opportunities

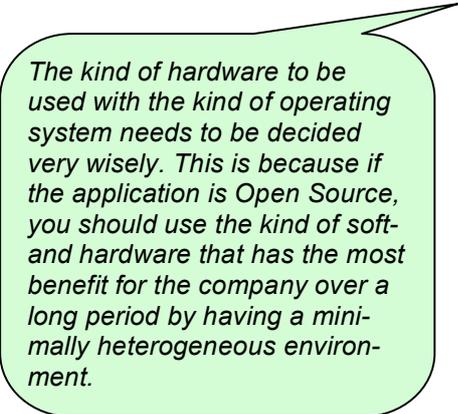
I think these are the most important issues within the area of business continuity concerning Open Source. Each point now needs to be ranked to see if Open Source is more of a risk or an opportunity to your business. I see it as more of an opportunity because you have so many options regarding what you can do with the software and how it can help you to support the different departments in the best way.

4.3 Platform

In this section I would like to discuss the Open Source issues with the platform. On the whole, the platform you are using dictates what kind of problems you are facing. As already mentioned, for most Open Source operating systems quite a lot of prepackaged software is available, and you do not have to compile the executables yourself as the distributor already did it for you.

The biggest issue is that different operating systems often have the configuration files in different places. This happens because different people think differently about how software should be installed, configured and maintained. Another point is that the configuration differs very often because each operating system has a different environment – different people think differently about a problem. This often reflects in a different configuration. The next issue I would like to mention is that during compilation, some kinds of software collect platform specific parameters and use them statically after the compilation. One example is the apache web server, which statically includes the physically available memory in the binary.

It is a big problem if you have very heterogeneous IT Architecture, because your engineers require in-depth knowledge of each operating system if they are to support it efficiently and configure it properly. Documentation is also a more complex task for such systems.



The kind of hardware to be used with the kind of operating system needs to be decided very wisely. This is because if the application is Open Source, you should use the kind of soft- and hardware that has the most benefit for the company over a long period by having a minimally heterogeneous environment.

This gets even more complicated if you are using no Open Source operating systems and install Open Source applications on it. The reason for this is that quite often Open Source software is not available for such operating systems in a prepackaged format. This means you to need to build the executables yourself, which requires a lot of work before you can actually start with building the required applications, as pointed out in sections 3.2 and 3.5. Configuration files and

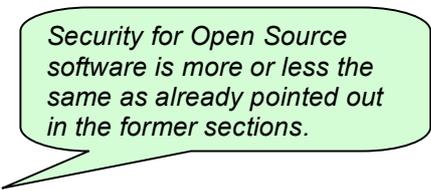
paths might again be different because it is a different operating system, and the engineer implementing the software thinks differently about its installation location. Although this might only be a minor problem during the installation and configuration process, when maintaining it you may suddenly realize that such a way of operating the software is inefficient. You will face some of the problems while you are consolidating or migrating your IT environment, as discussed in sections 3.3 and 3.4 respectively. It also applies to a situation where you would like to set up a new service running on a different server. This is easy if you use the same operating system as the one where such a service is already up and running and for which prepackaged software is available. If you are using a different operating system or there is no prepackaged software available, then you may have to redo the same work again. This needs to be done because you might not be able to just transfer the required software to the new server due to the fact that there may be dependencies that cannot be fulfilled.

Therefore, as can be seen, there are a few problems associated with having no clear strategy and standardization process within your company. This also might be a problem if there is high IT staff turnover (high fluctuation rate) because different people like or dislike different things. By having no standards, the company might end up in a very

heterogeneous environment which is inflexible and in which no one will know what will happen if something fails.

4.4 Security

I have already spoken about the issue of security in section 3.6 by focusing on UNIX. All points that have been mentioned concerning Open Source operating systems apply to Open Source applications as well as the points about security mentioned in section 4.2.



Security for Open Source software is more or less the same as already pointed out in the former sections.

For a company it is very important to have a clear security strategy and a plan of what tasks and procedures are required to ensure system security. This plan should also include tasks and procedures about what needs to be done in the case of an incident and in cases where it is more or less impossible to update the systems with the latest security patches.

5 The Solution – OpenPKG

We now have a clear understanding of the vast majority of Unix and Open Source related issues that may appear in a datacenter environment. Using OpenPKG will solve some of them and buying additional services by OpenPKG GmbH will solve a few more. By using OpenPKG a few additional benefits will become available to your company as well in comparison to just using Open Source software. The solutions will be described step by step. In the first section of this chapter I will discuss issues solved by using OpenPKG. In the second part of the chapter I will point out what kind of additional services from OpenPKG GmbH are available and how it can support your business.

At the end of this Chapter you will be able to decide whether or not OpenPKG is a solution for you and what part of OpenPKG is required to improve your business.

5.1 OpenPKG foundation

5.1.1 Solving issues between the different UNIX types

This is one of the biggest issues in a heterogeneous UNIX environment. To resolve this issue, the OpenPKG foundation performed several tasks to enable you to use Open Source software with different UNIX types.

Cross-platform functionality is enabled by the brainpower of members of the OpenPKG foundation, provided tools and package management.

OpenPKG provides the same environment and user interface on the supported OS's, which enables standardization and automation.

The first task is to create an environment that only requires a specific directory – chosen by the engineer and based on his or her needs – where the software is installed. Generating a binary package requires a compilation process beforehand. All parameters required for the compilation are stored in the generated package and in the package database after it has been installed.

People participating in the OpenPKG foundation take care of all packages and their Cross-Platform functionality. The OpenPKG members prepare the packages in such a way that they can be compiled on the different operating systems just as well. You can either use pre-

compiled packages, or you can use the source packages provided by the foundation. The package management, provided tools, and brainpower – provided by OpenPKG – enable the Cross-Platform functionality. To manage the packages, RPM (RedHat Package Manager) is used and is included in the “OpenPKG package”.

The biggest benefit for the business is having one environment that works the same on all different kinds of UNIX. This only requires operating system knowledge and environment knowledge. The user interface is the same on all systems, as are the parameters for the application. Even if the operating system is different, it allows standardization on the different operating systems from a software/application perspective. This allows the engineer to work as efficiently as possible.

Solving the issue between the different kinds of UNIX is one part of solving the issue of the availability of software, which I discussed in section 3.1. Building packages for the software is the other part of solving the availability issue of software, which I will describe in the following section.

5.1.2 Solving packaging

Sometimes it is required that software is compiled on the platform on which it will be used. This is required because some platform-related parameters (e.g. physical memory, which can be different on different systems) are statically compiled into the software.

There are two types of packages, namely the source package and the binary package. The source package includes the source code and the configuration files for building a binary package. The binary package includes all binary files related to the software, and configuration files required by the software.

To solve the “platform related parameter issue”, OpenPKG uses the source package and enables the user to create his or her own binary package on a certain platform. If the configuration within the source package is generated properly, the binary package can be built on each supported platform. This is very useful and enables cross-platform functionality. The binary package gets built on each platform with the same command, which automatically executes all platform related tasks for building it. During the compilation of the binaries, platform specific parameters get automatically configured into the binary package by the standard procedure of the software, and the output is an automatically generated binary package ready for installation.

Binary packages can be built on every platform with the same command.

Platform related tasks get automatically executed.

Platform specific parameters get automatically configured during compilation.

The installation path for the OpenPKG instance can be chosen before compilation.

Installation path for all other OpenPKG packages gets applied automatically.

Configuration files can be easily transferred on equal systems.

Another problem that gets solved by using packages is the problem with the installation path of the software. By using OpenPKG, the engineer can choose the installation path before starting the compilation of the “OpenPKG package”. The chosen installation path is the path of the entire OpenPKG instance. All additional installed OpenPKG packages related to the instance get installed under the same directory structure and the path information gets stored in the package during compilation. If you are using the same path for the instance on two or more systems, you only have to install the software and copy the configuration files to the new system – assuming you would like to have the same configuration on it. If you would like to install the software in a different hierarchy, you can easily copy over the configuration files and exchange the former paths used in the configuration file by executing a command. If the path, for example, is dependent on the platform, OpenPKG developers take care of it and include the required changes into the source package, which will be considered during the compilation process automatically.

configuration files to the new system – assuming you would like to have the same configuration on it. If you would like to install the software in a different hierarchy, you can easily copy over the configuration files and exchange the former paths used in the configuration file by executing a command. If the path, for example, is dependent on the platform, OpenPKG developers take care of it and include the required changes into the source package, which will be considered during the compilation process automatically.

A very important point that needs to be mentioned is Sarbanes-Oxley Act (SOX) related. With SOX you have to track and document all changes made to a system.²⁷ If you

are using packages, documentation gets easier because each file is documented in the package and, after installation, in the package database. This information can be gathered from there and put into your documentation. This might reflect in less additional tasks like snapshots to document the system. You still have to document the changes you make to the configuration files, but you have to do this anyway.

Companies that have to follow SOX can document changes to their systems more easily.

5.1.3 Solving migration

If you have to migrate a system for whatever reason, you might come across some migration issues. By using OpenPKG, the migration of a system can be performed more easily. You can more or less use the same process as is used during a consolidation – i.e. reinstall service on the same system by using OpenPKG, install service on the new system, copy configuration files to the new server, perform some local adjustments, and migrate the service.

By using OpenPKG, migration to a different system is very easy. It is fast and takes a minimum of resources.

OpenPKG provides the flexibility by choosing an operating system and suppliers can be changed more easily.

If all Open Source software used on your systems is installed via OpenPKG, the migration to a different system is very easy and does not take a lot of time or resources. This gives the company the flexibility to decide what kind of operating system needs to be used in the company whenever they wish. If, for example, one operating system supplier goes bankrupt, the company can easily change their environment and are not affected much by the bankruptcy of a supplier.

If there is a technology shift in that a different supplier provides better equipment that supports your needs better than the one you are using, then you can change it easier if you are using OpenPKG. The reason for such an easy migration path is the already mentioned cross-platform functionality of OpenPKG. Migration is often an issue if a company would like to rationalize, improve efficiency, and cut down maintenance costs. Some aspects that are related to consolidation might be faced during a migration as well.

5.1.4 Solving consolidation

If you have to consolidate services for whatever reason, you have to ensure that the services provide the same functionality after the consolidation. Just moving servers to a different location is easy, but it does require a longer downtime of the service, and cannot be solved by using OpenPKG. However, OpenPKG can solve the issues via newer and faster hardware, and a reduction in the number of systems used.

If the company decides to buy newer and faster hardware to consolidate services, OpenPKG can be used to support the consolidation process.

The possibility to specify the installation path for an OpenPKG instance and to have more of these instances installed on one system is very useful for the consolidation process.

Figure 4 shows a costly setup where each kind of service is installed on a different server. Each system in this example is only utilized by about 20%, which means that the server is idle for about 80% of the time.

Figure 5 depicts an example of a consolidated setup where several instances are installed. A directory called services gets generated and the instances get installed in the services directory with a self-explanatory name for the service. If, for example, an engineer would like to see what kind of services are running on a system, he need only take a look at the documentation or into the services directory. The engineer might not require documentation to start finding a failure on the system.

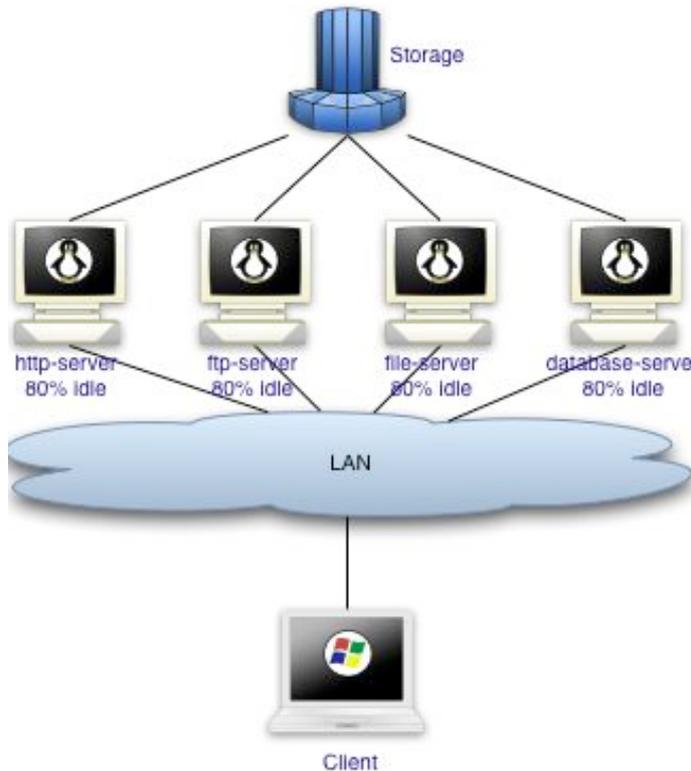


Figure 4: Environment before consolidation

With OpenPKG it is very easy to consolidate services. The easiest way to consolidate a system is to prepare the same service that you would like to consolidate on the same system again, but within the final directory – in my example `/services/{SERVICE}`. After this has been finished, the old service has to be shut down and the new service has to be started. Now you can prepare the system the services will get consolidated on, and when this is finished you only have to copy the configuration files for the service. Before you can start the service, some minor adjustments are required, such as: mount points, IP-addresses, net groups, etc. After this is done you can switch the service. This needs to be done for each service, step by step to enable easy problem resolution. Such a consolidation frees up hardware that can be used for other services or as replacements for broken hardware.

If, for example, an engineer would like to see what kind of services are running on a system, he need only take a look at the documentation or into the services directory. The engineer might not require documentation to start finding a failure on the system.

A different installation path can be specified for each instance.

More than one instance can be installed and used on a system.

Administrative and access rights can easily be given to the service department.

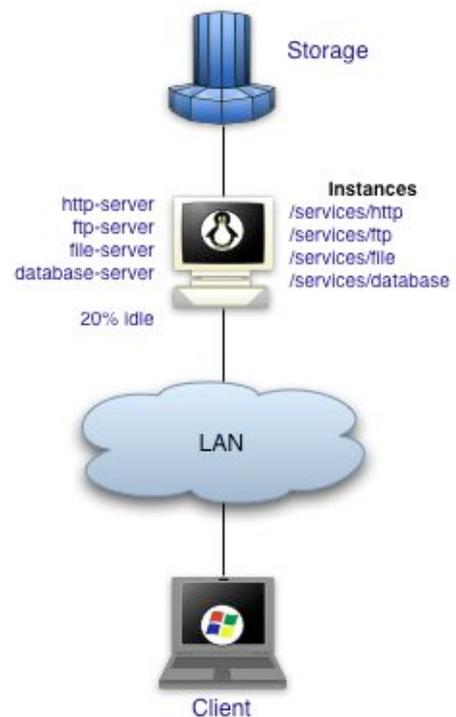


Figure 5: Environment after consolidation

During or after a consolidation the problem with administration rights may come up. This can happen if, for example, one department is responsible for the service and the other for the operating system. In such a situation, the initial setup has to be done together, and later on the access rights can be given to the service department. By adding some sudo parameters the service department can even restart their services whenever they like. By going through OpenPKG the service department can update and upgrade their service whenever there is a need for it.

5.1.5 Solving dependencies

Every kind of package has dependencies. These dependencies need to be solved before compiling or installing the package. The software developer knows on which other software packages his software is dependent and checks it in his makefiles. OpenPKG developers use these checks to get information about the package mechanism in order to provide user information if the installed software is sufficient or not. This is a kind of dependency list, which is prepared for each package and stored in it. It is divided into two parts, build requirements and installation requirements. This is necessary because there could be a situation where a package is required only during compilation or only after compilation. Each time such a situation happens the user gets informed what is required to resolve the issue.

OpenPKG provides information about whether the installed software is sufficient or not by performing checks during packaging and installation.

Whenever there is a problem, information is provided to the user on how to solve it.

To get informed by a software package about what is required to compile or install it is very useful and saves the engineer and the company both time and money. By getting this dependencies message the engineer knows

straightaway what kind of package is required to resolve the dependencies and to continue with the installation. This can be different on different kinds of operating systems.

5.1.6 Solving Platform

It is often the case that there are a few different Platforms within the company. The point I will mention here has some connection to point 5.1.3 and 5.1.4, because in some cases it applies to these sections as well.

Setting up a new Service on a new platform can cause problems during compilation. Normally if a new system gets installed, the latest patches are applied to it. Under certain circumstances, it could happen that the software does not compile correctly. One solution is to install the former version of a specific patch. The other one is to install a pre-compiled package. This could be compiled on one of the company's servers – with the same operating system – or by getting the correct package from OpenPKG and installing it on the new system. It is difficult and sometimes very time and resource consuming to know all of the required patches for a software program to ensure cross-platform functionality. This is done during the creation of the source package by the OpenPKG team.

Packages can be precompiled on one server and used on another one that has the same OS installed.

Applying the right patch, dependent on the OS is configured inside the package by OpenPKG developers and automatically gets applied.

If there is a need for setting up a redundant system the required parameters for compilation can be gathered from the package database.

The reason why I would like to mention the point about additional software patches here and not in the section about Solving issues between the different UNIX types is that there might be a situation where you have to apply a patch dependent on the release of the operating system (e.g. for Solaris 9 patch A is required, and for Solaris 10 patch B is required) to the source code of a software program. The developers of OpenPKG, who further develop all OpenPKG packages, solve this issue. They ensure that the software they provide works in the way it was intended and as easily as possible. When preparing such a source package, the engineer checks what kind of patches need to be applied to the source code dependent on the operating system. This gets configured for each operating system inside the package. When executing the package for compilation the package checks on which operating system it gets compiled and applies the correct patches automatically. This feature provided by OpenPKG saves the engineers of a company a large amount of time. The company no longer requires engineers with very in-depth software development knowledge, which might save money.

Another point that applies here is that the engineer already knows what parameters are required for software compilation. The reason for this is that he can get it from the already installed software package on one of the company's servers. This saves time for both the engineer and the company.

5.1.7 Business continuity solution

Business Continuity is another important point that gets considered by OpenPKG. Based on each issue pointed out in section 4.2, I will explain how OpenPKG handles these issues. In addition, I will explain how you can help to ensure your business continuity by supporting OpenPKG.

You can ensure your business continuity by contributing to the project, which can be done in many different ways. All contributions are useful to keep the project running.

How you can support the OpenPKG project is pointed out in the table.

The main idea of Open Source is give and take. If there is just one side giving, the project will collapse, which might have an impact on your business.

One important point is how Open Source projects ensure business continuity. The biggest issue when it comes to whether Open Source projects should continue is often money because they require equipment on which to test developed software. Due to the fact that OpenPKG is a project that provides cross-platform functionality, for each kind of operating system the relevant hardware is required. This hardware either needs to be bought, leased or sponsored. Another factor is that an entity has to pay for the Internet traffic they generate. All this can be very costly and may cause a wonderful project to go offline. The basic idea why Open Source software is available is: each person using that software should contribute to the project. This can be done in many different ways, e.g. sponsoring (man power, hardware etc.), money, feedback, etc. Just think what kind of benefit you get by using the software for free, and then think what you might have to pay for it if you have to buy it when

the project stops running. After you have thought about this you should think about how you can support the project to ensure its business continuity.

The following table shows how OpenPKG supports your business continuity and provides a few examples of what you can do to support OpenPKG, from which you will benefit as well.

NR	Issue	Solution provided by OpenPKG to ensure your business continuity	Your possible support to ensure OpenPKG continues
BCS-1	Most of the time there is no need to purchase it.	All OpenPKG software is available for free - you only have to register.	You can become an active or a passive member, sponsoring, spend money, etc.
BCS-2	Most of the time no royalties need to be paid.	As mentioned, it is free.	See above.
BCS-3	You can fix software problems yourself.	OpenPKG is available as Open Source, which enables you to fix your problems yourself.	You can share your fixes with the OpenPKG community and get them implemented
BCS-4	Solving software problems will be done quickly.	If there is a problem with OpenPKG, it will be fixed very quickly.	If you find a problem in OpenPKG, you can provide your fix to the community
BCS-5	Access to the source code.	OpenPKG is Open Source, hence you have access to the source code.	You can change the code in response to your requirements and you can see if there is any miscellaneous code.
BCS-6	Further development can be done yourself if the programmer no longer volunteers, or if you would like to contribute to the project you can as well.	All OpenPKG software is provided as Open Source software.	You are able to further develop the software yourself or you can take some actions beforehand, as discussed in BCS-1.
BCS-7	Many people can work on the source code because it is available to everybody in the world.	OpenPKG is Open Source, hence it applies to it as well.	Your engineers can contribute to OpenPKG because they have access to the source code. The release cycle already gets reduced.
BCS-8	Failures and bugs get detected quickly.	OpenPKG gets used by many entities that report problems immediately upon detecting them. This might happen before you detect them yourself.	You can contribute to the project by fixing errors, or by reporting your problems with the software to further enhance it.
BCS-9	Failures, bugs and security issues get fixed very quickly thanks to the big community of users.	OpenPKG is very fast at implementing security and bug fixes as well as failures. It also prepares packages for you that enable you to close these flows faster.	You can contribute to OpenPKG by taking some actions mentioned in BCS-1
BCS-10	Further development and enhancements to the software might get incorporated into the next version if you send them to the project owner.	OpenPKG takes enhancements into consideration and provides feedback on them. These enhancements get implemented more often if they are already provided in code instead of a wish list.	Prepare the enhancement and send it to OpenPKG. If you have more good ideas. become an active member.

NR	Issue	Solution provided by OpenPKG to ensure your business continuity	Your possible support to ensure OpenPKG continues
BCS-11	Changes only get incorporated into the next version if the project owner thinks that the change is useful.	See BCS-10	If you are a member you are part of the project team and you have a vote on it.
BCS-12	The software mostly comes as source code and needs to be compiled before it can be installed and configured.	OpenPKG comes as source package and as a package for your platform. You only have to build the package yourself if you would like to activate some special features, but this is very easy.	If you contribute the hardware architecture and software to OpenPKG the packages will support your environment as well. This helps you and OpenPKG.

Table 9: Business continuity solution

The table shows a number of ways how OpenPKG can help your business to continue using software free of charge. It also shows how you can support OpenPKG so that it is able to continue assisting your business. As I mentioned previously, the main idea behind Open Source is give and take. If one side only takes, and the other side always gives, then the system will collapse. Such a collapse will have an impact on IT environments using this software. The decision of what you take and what you give is up to you. There are so many possibilities depending on the industry you are in in terms of what you can contribute to the community – it does not have to be money all the time.

5.1.8 Security

The security of a system is a very important point but, as you know, software sometimes has security flaws. These security flaws get detected either by accident or by a very highly skilled person scanning the code for such problems. Scanning the code often takes a very long time – longer than the coding itself – and this is the reason why software is already installed and used when security flaws get detected. At the same time that it gets detected, the developers of the code are informed. Working together with the group of developers who wrote the software, they start to solve the problem. It

can take some time until a solution is implemented. As soon as the software with the solution is made available, OpenPKG is informed and starts to implement it into the available package. After the implementation is done and tested, the new packages with the security flaw fixed, become available. Now the user of OpenPKG can start to install the newly available package to solve his problem.

The delay between the announcement of the source codes containing the bug fix and its availability as an OpenPKG package is very small. This is possible because of the many automated standard procedures within the OpenPKG foundation.

You might argue that there is a delay between the new source code and the availability of the package. However, you need to ask yourself: Are you really faster at applying the

OpenPKG is very fast in implementing security fixes, because they get informed when a new patch, which closes a security hole, is available. The patched software gets issued after it is successfully tested.

By having many systems to update using packages is much faster than compiling the software yourself.

required patches, compiling and installing the software on your servers, and does the software provide the same functionality as before? Later on I will highlight a case where a university was seven times faster at closing security holes as a direct result of using OpenPKG.

5.2 OpenPKG GmbH

5.2.1 Consulting

To start using OpenPKG some companies require a consultant onsite to analyze the requirements and to design an appropriate solution together with the company experts. For Open Source software this kind of service is seldom available. In the case of OpenPKG GmbH, consultants not only focus on OpenPKG itself, but they also provide consultancy for Open Source software packaged for OpenPKG.²⁸

Consulting for OpenPKG software can be bought at OpenPKG GmbH.

5.2.2 Development

Package adaption based on company requirements can be bought at OpenPKG GmbH.

OpenPKG GmbH software developers adapt existing packages to fulfil customer requirements and create new customized packages and the related tools. This can be useful if a company requires specific changes to a package and does not wish to share it with others. If a company requires some additional tools to manage

OpenPKG, these tools can be developed as well.²⁹

5.2.3 Deployment

OpenPKG Package deployment can be bought at OpenPKG GmbH.

If a company would like to deploy OpenPKG and has no free resources to perform this task, the OpenPKG GmbH deployment team can do it. The team implements designed solutions into the company's production environment. The required and supported operating system will be installed and configured based on customer standards.³⁰

5.2.4 Training

Before or after implementing OpenPKG the company may require training for their IT staff on this technology to have a common understanding. Depending on the company requirements, trainings can be held for different experience levels - from user to developer level. The content of the training can be selected from a pool of available knowledge modules. The training is a balanced mixture of presentation and hands-on experience.³¹

Trainings based on OpenPKG technology can be bought at OpenPKG GmbH.

5.2.5 Support

Should a complex problem require troubleshooting, a company can come to OpenPKG GmbH for support. The engineers will find a solution for the software problem. This can be done either remotely – although the company has to allow remote access here – or onsite depending on customer policies. Once the decision to use Open Source software has been made, getting support is often crucial.³²

Support and sophisticated troubleshooting can be bought at OpenPKG GmbH.

5.2.6 Maintenance

OpenPKG GmbH developers maintain modified OpenPKG packages. These modifications get applied to new vendor releases and are made available in the OpenPKG format only to the company. This includes back-ported security updates, which get applied to the software even if the software is outdated.³³

Maintenance for packages especially tailored to your company can be bought at OpenPKG GmbH.

This is very useful if companies have specific requirements for a piece or bundle of software. For such a task the company no longer requires experienced developers because it can be outsourced and the company can concentrate on their core business.

5.2.7 Operations

The smooth operation of OpenPKG setups is a tribute to the OpenPKG GmbH System Administrators' proactive approach. Customer setups get monitored and problems get rectified before they can affect production services. It includes the very important task of quick security fix deployment and other requirements of operation.³⁴

Operation of OpenPKG setups can be bought at OpenPKG GmbH.

5.2.8 Project management

OpenPKG GmbH project managers manage OpenPKG rollouts in terms of the quality and budget to finish the project and the related goals in time. For the rollout project they interface between management, engineering, and users and fulfil each of their requirements.³⁵

The project management for OpenPKG rollouts can be bought at OpenPKG GmbH.

6 Case Studies

In this section some case studies of different organizations are presented to show how they have solved their problems by using OpenPKG and how it supports their business. To do this I have chosen a university from the United States, an international telecommunications company with its headquarters in the United Kingdom, and a large scientific research center in Germany, who use OpenPKG as a basis for their development. Each organization experienced different kinds of problems, and you will see how they acted and solved them.

6.1 Portland State University, USA

David Fetter from Portland State University (www.pdx.edu) in the United States of America provided the following information. David Fetter is an employee of the Portland State University and is responsible for a flexible, effective, efficient and highly available IT environment. After the foundation OpenPKG was established, he joined it immediately because he thought: "This is a great idea that I would like to support". In the first election for the head of the foundation he joined, he volunteered for the seat as Director of Software and Maintenance, a position, which he still holds today. You can visit the link <http://www.openpkg.net/department/sm.html> if you would like to know more about David Fetter.

The University requires a flexible, efficient, effective and high available IT environment.

In this section I will provide some Information about the IT environment, the issues that existed before using OpenPKG and the issues that have since been solved, as well as those that still exist and that have been created by using OpenPKG. The main driver for changing the IT environment at Portland State University was because they planned to reduce overall maintenance costs for keeping core software up to date. They looked at several possible solutions and even started to develop their own software maintenance scheme. They discovered very quickly that they were trying to do more or less the same as OpenPKG and this is what they actually needed – this happened before the OpenPKG foundation was formed.

6.1.1 IT environment information

The IT environment at Portland State University consists of roughly 100 servers, which are administered by seven people. The operating systems installed on these Servers are SUN Solaris and Linux. Two versions of Solaris are installed, Solaris 8 (about 5 systems) and Solaris 9 (about 45 systems). For Linux servers the University is using RedHat Enterprise Linux 3 (about 45 systems) and SuSE 10.0 (about 5 Systems). OpenPKG is used on all of these servers with the exception of one or two. Systems get upgraded to newer releases once a year. In the meantime

The University has about 100 servers consisting of Solaris and Linux systems, which provide many different services. The servers are administered by 7 engineers.

only updates are performed.

All Servers are installed over the network via network installation mechanisms. For Solaris systems the mechanism is called jumpstart and for the Linux it is called kickstart. The initial configuration of a new server is done on the server that offers the appropriate network installation service. After the network installation for a new server is initiated and everything is configured correctly, there is nothing to do until the installation is finished. The time required for an installation depends on the hardware and the operating system.

Portland Sate University offers several different services to their users. To provide these services they use the following software:

Software	Service	Software	Service
Sudo	To allow a user to perform certain tasks under administrative rights	sendmail	Offers Mail service for sending and receiving mails
Bind	To hold information about their own DNS domain and to look up domain names for clients requesting it	Imapd (cyrus)	User mailbox, mails received for the user get stored and can be collected via e-mail client
NTP	Providing time service over the network for clients and server	samba	To offer Unix file systems to Windows users
openldap	User directory with information about the login, mail, etc.	apache	Used to offer Web services
Rsync	To synchronize two systems to have the same data on them	lprng (custom made openpkg rpm)	To allow users to print, this application is adjusted for their needs
postgresql	A database which holds specific information	pureftpd	To allow users ftp access to their files
subversion	A system which tracks changes performed by an administrator	dhcpcd	To ensure users get IP-addresses assigned by connecting their PC
SNMP	To enable systems for monitoring	ircd	Software to enable users to chat.
Nagios	Monitoring system which provides information about the status of the environment	cfengine	A policy language and agent for building expert systems to administrate and configure large computer networks
Mysql	A database which holds specific information	amd	To automatically mount file systems from other systems
openssh	Secure connection to logon to a Unix system to perform certain tasks	scp only	To copy data from one system to another via a secure connection

Table 10: Services provided - Portland State University

6.1.2 Issues before using OpenPKG

Before Portland State University started using OpenPKG they installed Open Source software directly onto their systems. On some of their servers they could use the pre-packaged software provided by the distributor (RedHat and SuSE). On Solaris systems they had to either install software packages from <http://www.sunfreeware.com/> if the package was available, or to compile and install Open Source software directly without using any kind of packaging mechanism. In doing so they faced some problems including: dependencies on other applications that needed to be built and installed first; if the operating system did not have the correct patches installed, the software did not compile correctly; some additional patches to the software were required to get it compiled on the operating system, etc. This is as discussed in section 3.5.

Only limited software was available in a packaged format. Dependencies on other applications sometimes caused problems and additional patches were required. Sometimes it was required to compile the software on the system where it was used. Migrating systems required many human resources. Recompilation of software on new hardware was required. Configuration files adjustments had to be performed. The NFS server was used heavily. Closing security flows required the entire department.

There have been problems regarding the platform as well. Some kinds of software need to be compiled on the platform where they are used. If this is not done, the application may not work properly and will cause some strange errors. Some other problems are that paths to configuration files and use in those files depend on the platform. This was described in section 4.3 Platform.

During migration and consolidation some other issues surfaced. In the process of migrating systems, quite a lot of human resources were required to perform the necessary tasks within a certain time. The problems faced during that time were configuration file adjustments, compiling problems, and the fact that software was not available in a packaged format. Systems were required to be prepared like the original one in advance without prior knowledge of exactly whether it would work or not – as pointed out in section 3.3.

One reason for consolidating systems was the availability of newer and faster hardware, which was required and duly caused some problems as well. Software needed to be recompiled on the new hardware. Configuration files required adjustments for the new server and new directory structure – as previously discussed in section 3.4.

The university also had the problem of using different kinds of UNIX and were faced with the issue that if they would like to consolidate or migrate systems, quite a lot of manpower would be required to fulfill this task. They also faced the issues pointed out in section 3.2.

Additionally to the above mentioned difficulties, they had to cope with the problem of a heavily used NFS server. This happened because by reducing the maintenance of the installed software, the software was installed on a shared file system and could be used on different servers. All these servers required the same operating system to run the software, and whenever the software got executed, excessive usage of the NFS server and network occurred.

The next problem faced was the matter of a security flow. In such a situation, the entire department was needed – consisting of seven people – to perform the required updates

on the affected servers. They had to divide the affected servers by the number of engineers and each engineer got one piece of the entire environment to upgrade. As a result, they had to stop any further development of their systems in order to close the security hole.

There are clearly many issues and some of them can be solved by using OpenPKG, which I would like to point out in the next section.

6.1.3 Issues solved by using OpenPKG

As you have seen in the previous sections the university uses a large amount of Open Source software and they have four different kinds of Unix in use. Before using OpenPKG they faced a number of problems that actually changed while using OpenPKG, which I would like to explain now.

The first problem that got solved by using OpenPKG was dependency on other applications, as was described in section 5.1.5. It enabled the engineers to know what kind of software has what kind of dependencies. If a pre-required package is missing, the administrator gets prompted to install it first.

Another problem, which got solved by using OpenPKG, was the problem of having an outdated or wrong patch level for the operating system, meaning the software did not compile correctly. This issue got solved in the way it was discussed in section 5.1.6. This allows the engineer to proceed with the installation of the software without applying the latest operating system patches. Installing operating system patches can still be done on a regular basis.

Depending on the hardware and on the operating system, some additional software patches may be required to have compileable software and to ensure it is working properly. This got solved by using OpenPKG, as is described in section 5.1.6.

The problem that got solved was that some kinds of software need to be compiled on the platform where they are in use due to certain platform specific parameters. Using source packages, from which you will get binary packages after compilation, easily solves this issue, as was detailed in section 5.1.2.

An additional problem that got solved by using OpenPKG is the problem of paths to configuration files and the paths in configuration files. This issue was already discussed in section 5.1.2. This feature enables the university to choose the path it likes for installing OpenPKG software.

Due to the fact that OpenPKG supplies cross-platform functionality software in a packaged format for different operating systems – as mentioned in sections 5.1.1, 5.1.2, 5.1.5, and 5.1.6 – it solves the issue of requiring a high amount of human resources to perform a certain task (e.g. to update to a newer version of the software). The university



OpenPKG solved dependencies on other applications. Outdated OS patches are no longer an issue. OpenPKG handles the requirement for additional patches. Platform specific parameters get automatically detected by package generation. Paths in configuration files and to the software get automatically adjusted. Packages can be installed on different OSs because of their cross-platform-functionality. It involves less work if software is required more than once. Network traffic gets reduced. Solving security flows only takes one person instead of seven.

is now able to perform the same task with one person which previously required seven people.

Using the provided solution of OpenPKG solved the issue of compiling problems of Open Source software. The OpenPKG developers take care of it during the generation of source OpenPKG packages, as discussed in section 5.1.2 and 5.1.5. This saved the university both time and money.

Another problem faced by the university, namely the small amount of available Open Source software for SUN Solaris systems, got solved by using OpenPKG and especially by using the technique described in the sections 5.1.2, 5.1.5, and 5.1.6. By using OpenPKG the university became more efficient in that regard.

The next problem that got solved is the problem that might occur when software is required more than once. From an Open Source software point of view, the OpenPKG feature discussed in section 5.1.2 solved this issue. This saved a large amount of installation time and enabled the university to automate the installation process.

The eleventh problem that got solved by using OpenPKG was the difference between UNIX systems, as described in section 5.1.1. Not having the issue of the difference kinds of UNIX enabled the university to easily migrate and consolidate. This applies to the process of getting newer and faster hardware as well.

The next problem that got solved was the heavy usage of the NFS server. The ease of compiling and installing of OpenPKG packages on each system locally solved this and did not lead to an increase in maintenance.

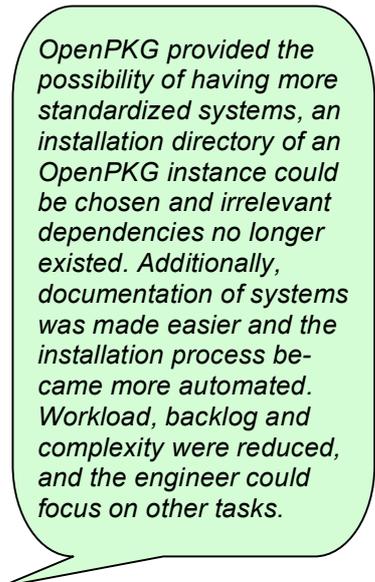
Finally, the mentioned issue of requiring a large amount of manpower in times of security incidents was also solved by using OpenPKG. The solutions provided by OpenPKG for solving this problem were pointed out in the sections 5.1.2 and 5.1.8. In addition to that, the installation of required updates within the same timeframe now requires only one person, instead of the seven needed before.

All in all, using OpenPKG packages instead of using Open Source directly solved thirteen existing problems. This put the environment in a much better position and the engineers were far more satisfied with it. For the university, using OpenPKG did not represent many cost savings, but it did offer huge timesaving potential – which is cost saving at the end of the day. Work and system efficiency increased as well.

6.1.4 Additional Benefits of using OpenPKG

There were additional benefits that were made available to the university by using OpenPKG. Some examples of such benefits are standardized installation directories, standardization in more detail than before, systems that can be documented easier, more automated installation of new systems, improvement in work efficiency, a reduction in work load and backlog, and a reduction in complexity.

The reason why the installation directories could get more standardized was that for each service offered by the server, a service directory – a directory for an OpenPKG instance – could get chosen. For example, if a server is offer-



OpenPKG provided the possibility of having more standardized systems, an installation directory of an OpenPKG instance could be chosen and irrelevant dependencies no longer existed. Additionally, documentation of systems was made easier and the installation process became more automated. Workload, backlog and complexity were reduced, and the engineer could focus on other tasks.

ing ftp and http, a directory structure could look like /services/ftp/... and /services/http/.... This is very useful because all service related software is installed under the same tree.

The standardization was able to be more detailed, because with OpenPKG you know what is required and most of the irrelevant dependencies no longer exist. This means that less software is required to get the service online. They are now able to standardize in terms of the package level and the version.

The documentation of systems is easier because they can get the information about the installed files from the package database. Before OpenPKG they had to get the information from taking a snapshot of the system before and after installing the software. In contrast, now they just have to gather it from the package database.

The installation of new systems was automated because installing OpenPKG always follows the same procedures, hence the automation. This saved the engineer a significant amount of time, which could then be used to work on other tasks.

Work efficiency is another point that got improved by using OpenPKG. This happened because the engineers required less time for installing software and configuring it. This time could then be spent on other important tasks.

The workload and backlog got reduced because installing software takes less time than before. Additionally, while the software is being installed, the engineer can work on other important tasks until user intervention is required.

The level of complexity was also reduced because each system follows the same principles. This enables the engineer to solve upcoming problems faster because he no longer has to search for the configuration and log files.

As you can see, by using OpenPKG, the university got seven additional benefits, which increased the level satisfaction with the system.

6.1.5 Issues after using OpenPKG

Having decided to use OpenPKG the university developed an issue with the release cycle of OpenPKG. The release cycle of OpenPKG is too fast for the university to always have the latest stable revision of OpenPKG installed. They can barely even update their systems by the time the installed revision of OpenPKG is no longer supported. Fortunately, they can afford to be a little bit more mainstream as they are a university. This might be a bigger issue for corporate entities, which could find it much more difficult to deal with this issue because other regulations apply to them.

The release cycles of OpenPKG are too fast for the university.

6.2 International Telecommunications Company

The required case information used in the following subsections was provided by the deputy director of Hosting. The director of Hosting is responsible for offering a flexible, efficient and effective IT environment, at the lowest costs and an availability rate of 99.999% (standard in a telecommunications environment). He has to ensure customer satisfaction and, therefore, very small maintenance

The goal of the Hosting department is a flexible, efficient and effective IT environment at lowest costs and 99.999% availability.

windows were required. Due to the fact that most of the customer servers use Open Source software, customer satisfaction was a challenge.

First of all, I will point out information about the IT environment. Then, I will guide you through the issues that existed prior to using OpenPKG, how these issues got solved and what additional benefits came along through using OpenPKG. Finally, I will discuss the issues that exist after using OpenPKG. The main driver for implementing OpenPKG was the requirement of an easy to use tool for installing, maintaining and de-installing Open Source software. Another driver was the fact that the headcount got reduced and the remaining employees had to take over all of the tasks from the redundant person. A few employees already knew OpenPKG at that time and suggested using it because it would help them in their day-to-day business.

6.2.1 IT environment information

The telecommunications company had an IT environment that consisted of about 100 very heterogeneous servers in 2001. By the end of 2005 it consisted of about 250 servers with OpenPKG installed. They had different kinds of UNIX platforms in use because of their different requirements. They used Free BSD and Linux, both on iX86 architecture and Solaris on SPARC.

The department administered about 100 very heterogeneous servers with 20 engineers in 2001 for internal and external customers.

The Hosting department has internal and external customers, both with different requirements. The fulfillment of the requirements of both is often a challenge. During the implementation of OpenPKG the different departments and external customers were trained and got used to it very quickly.

The telecommunications company uses the same Open Source software – as pointed out in section 6.1.1 – as Portland University to provide their services, and in addition they use:

Software	Service	Software	Service
Radius	An authentication service often used for router logins	postfix	Offers Mail service for sending and receiving mails like sendmail.
Procmail	A program for handling mails e.g. presorting, bounce, etc	Lynx	A command line Web browser.
whois	Tool to find out which IP and domain belongs to whom.	Ghostscript	Ghostscript is software that provides a mechanism to generate PostScript and PDF files.
lftp	A special kind of ftp server.	Vcheck	Vcheck is an Open Source Virus Checker.
Proftpd	Another FTP server	zebra	Zebra is an advanced routing software package that provides TCP/IP based routing services such as RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4, and BGP-4+
different commands	Many different commands provided by the GNU project.	Programming languages	PHP, python, perl and many perl modules, etc.

Table 11: Services provided - International Telecommunications Company

How OpenPKG can support IT business.

Without doubt, the dependencies on other software have to be fulfilled for the mentioned software in table 11. The division of the company on which this case is based had another 100 servers that had no OpenPKG installed. These were either old customer servers where no changes were allowed or possible, or the servers were owned by a different department.

6.2.2 Issues before using OpenPKG

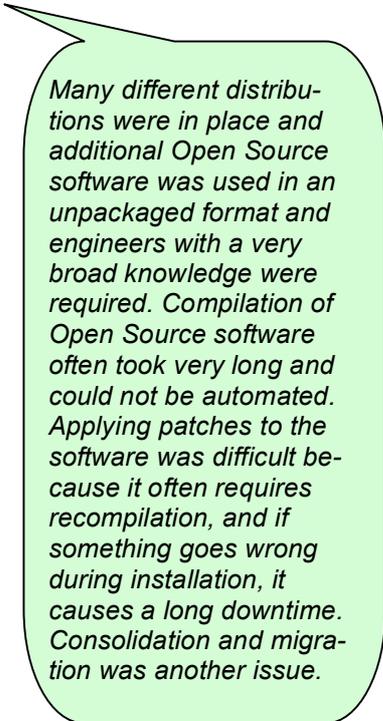
In 2001 the telecommunications company had not yet started using OpenPKG. At that time they had several problems with their IT environment. They used Linux from different distributors, Open Source software often needed to be compiled by the engineer, installing patches was another problem, large time lags existed between the installation of a system and when it actually went live, and an automated installation process was not yet implemented. Later on they also faced migration and consolidation issues.

Due to the fact that the telecommunications company used different versions of SUN Solaris on SPARC, Free BSD on iX86, and Linux from different distributors installed on iX86, they required engineers with a very broad knowledge. Consequently, the company faced all the problems pointed out in section 4.3. This situation meant that the engineer who was most familiar with a particular system, had to solve the problem as fast as possible, often regarding architecture. Whenever the engineer was on vacation, his colleagues had to perform the task, which naturally led to slower response times.

The compilation process of software was required to take place on the system where it would be used afterwards. This often took a very long time because some system related tasks had to be performed. This process could not be automated because it was different from platform to platform and from customer to customer. This issue also tied up a lot of resources and made it very expensive. These issues were pointed out in sections 3.2 and 3.5.

The installation of patches was another issue faced. Installing patches on the operating system was often very easy, but for the software compiled and not yet installed as a package, it was more difficult. These systems were live systems and for installing a patch the system was required to be taken out of service. Taking a system out of service and installing a patch requires a backup plan if something fails so that the system can be brought online again during the scheduled downtime without the patch. This was more or less impossible because of the recompilation of the software and the need for very good documentation. To perform such a task the issues mentioned in sections 3.1 and 3.2, and Business Continuity had to be overcome.

I have already mentioned that the installation of a new system took a very long time – between about one to two weeks, depending on the requirements. One reason was that the installation of the whole system could not be automated because the application installation and configuration needed to be done manually. The customer often had different application installation requirements, and the adjustment of paths was required



Many different distributions were in place and additional Open Source software was used in an unpackaged format and engineers with a very broad knowledge were required. Compilation of Open Source software often took very long and could not be automated. Applying patches to the software was difficult because it often requires recompilation, and if something goes wrong during installation, it causes a long downtime. Consolidation and migration was another issue.

through all stages until the installation and configuration was finished. This is further discussed in section 4.3. If something takes long and requires resources all the time, it is often very costly, which was indeed the case. Another reason was that the systems were very engineer specific, which caused problems during the maintenance stage, as already stated.

As mentioned in the previous paragraphs, the installation process could not be automated, because the customer requirements and engineer specific installation meant

The cross-platform functionality of OpenPKG allowed the automation of the installation process. The same commands are used on all platforms. OpenPKG handles dependencies on other software and is no longer an issue. The setups could be more standardized.

that it was costly and resource consuming. This became a big issue following 9/11 and after the dotcom bubble burst, because companies did not spend the same amount of money as before. This forced the department to become more efficient, which meant it was required to migrate and consolidate its systems. During this procedure the company faced the problems indicated in sections 3.3 and 3.4.

Another important point for the company was business continuity. As a result of using Open Source software without a package mechanism, which needs to be compiled manually based on the engineers experience and on his way of installing software, business continuity was an issue the company faced. By using Open Source in that way the

issues pointed out in section 4.2 have to be seen more as a risk instead of an opportunity.

It's clear then that this company faced many problems as well, and due to changes in the market and the internal changes they made themselves, they had to solve these issues to meet their targets.

6.2.3 Issues solved by using OpenPKG

In this section I will describe how the telecommunications company solved their issues with the IT environment. I will then comment on how they benefited from it. The situation at the telecommunications company was that they used many different kinds of UNIX and, additionally, quite a large amount of Open Source software was in use.

One main issue that got solved by using OpenPKG was that the Packages provided by OpenPKG could be installed on all of the different platforms. This allowed the telecommunications company to automate the installation process of their systems and solve the platform issue in the way described in section 5.1.6. After the installation-configuration was finished, only the command for installing the system needed to be executed. During the installation no engineer intervention was required. Once the installation was finished, the engineer could start configuring the system.

Compiling software on different systems was the next issue that got solved. Using packages from OpenPKG that could be compiled on the different kinds of UNIX by using the same command solved this, as is described in sections 5.1.1 and 5.1.2. In addition, the dependency issue also got solved, because the developers of OpenPKG take care of this, as is described in section 5.1.5. All this supported standardization, which will be pointed out in section 6.2.4. Having such an environment is very useful, because it enables the engineers to work more efficiently and makes the company more profitable. Junior engineers could be used to perform the same task that previously would

have required a senior engineer, which naturally has the additional benefit of driving down costs.

Installing software updates or upgrades was another issue that got solved. The possibility to perform this task by using packages like those described in section 5.1.2, enabled the company to achieve it in a shorter maintenance window. The fact that all package preparation parameters got stored in the package and in the package database (for installed packages), enabled the engineer to prepare the package beforehand. The original packages were still available as a package file on the file system for the backup plan if something failed.

Junior engineers were able to perform tasks that required senior ones before. Installing software was easier because packages were used that enabled shorter downtimes. Problems were found faster because systems followed a certain structure. Consolidation and migration was easy because of OpenPKG.

Using OpenPKG solved the availability of software issue, which was another point. Due to the fact that OpenPKG developers provide a large amount of Open Source software in a packaged format and ensure that the software can be used on different platforms, as mentioned in sections 5.1.1 and 5.1.2, it helps to solve this problem and allows cross-platform functionality.

The standardization of the installation across the systems and the automated installation process enabled the engineers to work more efficiently than before. Inefficient working periods were reduced to a bare minimum and the engineers could concentrate on the important tasks such as finding problems faster because systems were following a certain structure. It enabled the department to administer more servers with fewer engineers than before. All this was implemented just at the time the management announced a reduction in headcount.

The headcount reduction meant that the systems had to be consolidated and migrated for additional cost savings. Firstly, it was required to consolidate systems and, later on, to migrate them. All of these issues got solved by OpenPKG, as was pointed out in sections 5.1.3 and 5.1.4. Without OpenPKG this would not have been possible in such a short period of time and would have caused more downtime for services.

Business continuity was an important point for the telecommunications company. Using OpenPKG solved their business continuity issues in some areas, as mentioned in section 5.1.7. They decided to contribute manpower and their Internet connection to the project. This was very useful for the software development to deploy additional functionality, which helped solve other issues.

6.2.4 Additional Benefits of using OpenPKG

An additional benefit to the company was that less engineer hours were required than before to set up a system. This occurred because OpenPKG provided a command that was the same on all systems, which allowed the automation of the installation process of Open Source software installed on the systems.

Additionally, maintenance efforts were reduced dramatically. Prior to using OpenPKG 20 engineers were required to manage 100 servers. After using OpenPKG, only 9 engineers

Headcount was reduced by about 50% and, over the same period, server numbers increased by 250%. More standardization and documentation was possible. The department was able to have a mixture of different engineer levels.

were required for 250 servers. As you see, the amount of servers increased by 2.5 times and, at the same time, the headcount got reduced by about 50%. This happened over a period of four years (2001 – 2005) and gets represented in the costs of the department. This was the efficiency enabler for the department.

Another benefit was the possibility to standardize systems and to document it from the beginning. OpenPKG enabled the installation of an application under a certain directory structure by offering the possibility to specify the installation path. This ensures that all parts related to that instance get installed under the same directory structure.

The benefit is that OpenPKG provides an environment and deploys Open Source software in a packaged format for different operating systems. OpenPKG provides documentation on their environment which follows the same structure on each operating system. This enabled the company to have different engineer levels and meant they no longer required highly skilled developers.

The fact that OpenPKG provides a release version that is tested on the different systems, enabled the company to have a stable version installed. Every department owned system was always upgraded to the latest release. Customer servers were only upgraded every second release or less, because often a large amount of coordination was required.

6.2.5 Issues after using OpenPKG

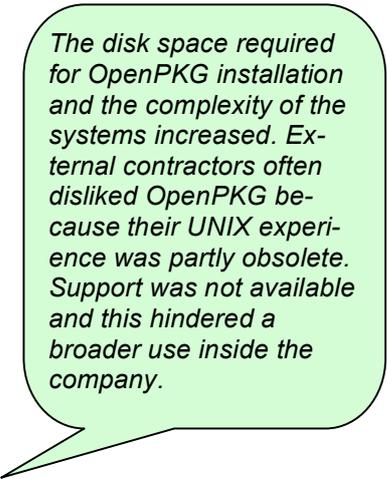
By using Open PKG the company had to deal with a few problems. One problem was the increased disk space requirement. This is a negligible issue because the operating system itself only requires a minimum of disk space, and most of the time the available free space is not used.

Another point is the increased complexity of each server for the reason that more software was installed on each system. This was done to utilize the systems, which were previously about 80% idle. Due to the instance functionality provided by OpenPKG, the complexity was reduced to a minimum.

A problem that was also dealt with was that external UNIX contractors disliked OpenPKG, because it made their UNIX experience partly obsolete. They could no longer demand the high amount of money as before because most of the knowledge was already built into the packages.

The fact that OpenPKG by design is very clean and self-consistent partly breaks down some conventions known to UNIX old timers. This means that those people have to cross a mental barrier before they can accept OpenPKG.

The fact that no support was on offer from OpenPKG was an additional issue. This meant that OpenPKG was only used in some departments because others liked to have support contracts. This was a major requirement of the UK headquarters even if it was defined as a good solution. This is no longer an issue as since January 1st 2006, support can be purchased from OpenPKG GmbH, as pointed out in section 5.2.5, Training, maintenance, consulting, development, software deployment, operations and project management have been other issues. These were only minor issues for this company,



The disk space required for OpenPKG installation and the complexity of the systems increased. External contractors often disliked OpenPKG because their UNIX experience was partly obsolete. Support was not available and this hindered a broader use inside the company.

but might be more important ones for others. The reason for this was that the company contributed developers to the OpenPKG project and resultantly already had the knowledge of OpenPKG in-house. All these issues can be solved now by contracting with OpenPKG GmbH, as pointed out in section 5.2.

6.3 Fraunhofer Institut Information- und Datenverarbeitung

6.3.1 IT environment information

The Fraunhofer Institute consists of about 150 systems at remote sites, SUN Fire v120, with mainly SUN Solaris 10 installed. Additionally, they have about 30 Servers placed in their local datacenter. These servers are either Solaris 10 on x86 or Solaris 10 on SPARC. Within the Fraunhofer Institute there is no UNIX system that is operated without OpenPKG installed. On these systems the Fraunhofer Institute uses about 130 different OpenPKG software packages starting with a shell and ending with server software like DNS-server. Within the Fraunhofer Institute the OpenPKG release gets upgraded every six months to keep on track and utilize the latest features from the software and OpenPKG.

The Fraunhofer Institute has about 150 UNIX servers (SUN Solaris) at remote sites, 30 in the local datacenter and one person providing all OpenPKG related tasks.

OpenPKG is mainly used for installing, updating, upgrading, and de-installing software packages, because it provides a very good packaging mechanism (RPM) and cross-platform functionality. Within the Fraunhofer Institute one person performs further development of OpenPKG to adjust the software to fulfill their needs. Different people perform the software administration itself because these are software related tasks that do not require a very detailed understanding of OpenPKG.

6.3.2 Issues solved by using OpenPKG

The Fraunhofer Institute decided to exchange some of their systems because of a much better price/performance ratio. They started to replace SPARC technology with Solaris 10 installed by x86 technology (Opteron) and Solaris 10. By already using OpenPKG, they were able to easily migrate the required software to the new servers, only having to deal with minor problems along the way. Without OpenPKG, they might have faced some of the migration issues pointed out in section 3.3.

By using OpenPKG the Fraunhofer Institute solved migration, dependency and packaging issues.

Another reason for using OpenPKG is that dependencies on other software packages are solved in an impressive manner and are clearly defined. This enables the Fraunhofer Institute to easily distribute the required software to the systems. Otherwise they might have faced the dependencies issues pointed out in section 3.5. In addition to that, OpenPKG packages provide a clear structure, which is very useful.

In addition, the Fraunhofer Institute benefited from the cross-platform functionality and could easily switch to hardware with a much better price/performance ratio.

6.3.3 Additional Benefits of using OpenPKG

One additional benefit was the cross-platform functionality. Upon starting with OpenPKG this was not required, but with the development of faster

hardware with a better price/performance ratio this became an important point. It enabled the Fraunhofer Institute to easily exchange the hardware, as explained before.

In addition, OpenPKG enables the Fraunhofer Institute to very easily develop “functional modules”. These modules can be installed on the remote systems without too much hassle. Furthermore, they provide a unified look and feel to all users.

6.3.4 Issues after using OpenPKG

The Fraunhofer Institute is using OpenPKG very heavily on their UNIX systems and sometimes they require support for it. The issue faced is that there is only a minimum of support for using OpenPKG as “freeware”.

The drawbacks faced were the lack of availability of support, untidy documentation of OpenPKG tools and only limited support for Solaris 10.

Another issue faced is in regard to the OpenPKG tool documentation. More specifically, this means that the documentation is a bit untidy, which causes OpenPKG tools to become unclear.

Another issue is that the support for Solaris 10 on ix86 does not exist on the whole. This is an important point for the Fraunhofer Institute, because they are exchanging the hardware to one with a better price/performance ratio.

6.4 Summary of Case Studies

These cases have shown how companies solved their problems with Open Source software for UNIX by using OpenPKG, as summarized in the following list:

- Software packaging
- Cross-platform functionality
- Standardized installations
- Automated installations
- Easier documentation
- Easy consolidation and migration of systems.
- Faster implementation of security requirements
- Better efficiency in server administration and installation
- Self developed software can be distributed easily
- Minimized differences between the different kinds of UNIX from a Software administrator perspective
- Outdated or wrong OS patch level
- Hardware supplier dependencies
- Dependencies on other applications
- Complexity reduction
- Less need for highly skilled technicians
- Standardized paths to configuration files
- Reduction of human resource requirements
- Reduction of network resource requirements
- Compiling problems with Open Source software

You can see that there are many advantages of the software; however, there are disadvantages as well, which I will detail in the following list:

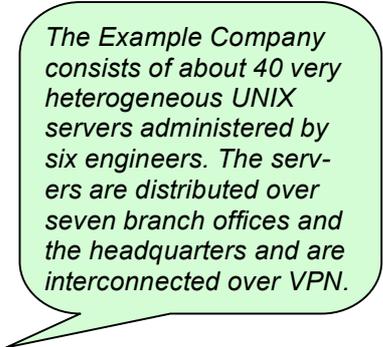
- Too fast release cycles
- Increased complexity when having multiple instances on one server
- More disk space is required

All in all, OpenPKG offers far more advantages than disadvantages, and all companies from my case studies benefited from using it and would like to continue with it.

7 Example Company - implementing and using OpenPKG

In this section I will use an Example Company to show how OpenPKG can support it. I will mainly focus on those parts of the company where OpenPKG can help. Potential issues of the company will be pointed out, and I will show how these issues can be solved. All this is linked to and in line with the company Corporate and IT-Governance. At the end of the chapter you will understand how it helped the company and you will be able to transfer certain solutions to your company.

The company I'm choosing will not belong to any industry, because the chosen issues and solutions can belong to every industry.



The Example Company consists of about 40 very heterogeneous UNIX servers administered by six engineers. The servers are distributed over seven branch offices and the headquarters and are interconnected over VPN.

7.1 Current Situation

The company has about 650 employees using the IT environment. The IT service has about 40 very heterogeneous UNIX servers, and about 150 Linux Workstations, mostly used in the technical department. In addition to these systems, there are about 500 Windows desktops and a few Windows Servers. Twenty administrators manage the entire IT environment. Engineers with a broad knowledge are required to support the complex environment.

The company likes to use software that best supports its business. This means that software with the best price/performance ratio will be used, regardless of whether it is Open Source or not. Whenever Open Source is used for Services, the software gets compiled on the system where it will be installed and no package mechanism is used. Maintenance for system updates mostly takes a long time and has to take place during the weekend because the service has to be up and running during the week. Taking the service down during the weekend is sometimes a problem. Each service is only available once because the company would like to save money in this area. It can sometimes cause problems if in the case of an error the service goes down, which means trying to find the problem can take a while from time to time.

The company is using the following Open Source software to support the business in the best manner:

- NFS
- squid (proxy)
- spamassassin (SPAM prevention)
- DNS
- mysql (database)
- apache (intranet/internet web server)
- ftp
- nagios (monitoring)
- amavis (virus scanning)
- ssh
- imap (cyrus)
- postfix (mail server)

and, in addition, some programming languages like perl, php, etc.

Figure 6 shows the Internet connection for each office site. Each branch office is connected to the headquarters over Internet by a VPN. The blue line is an example of such a connection.

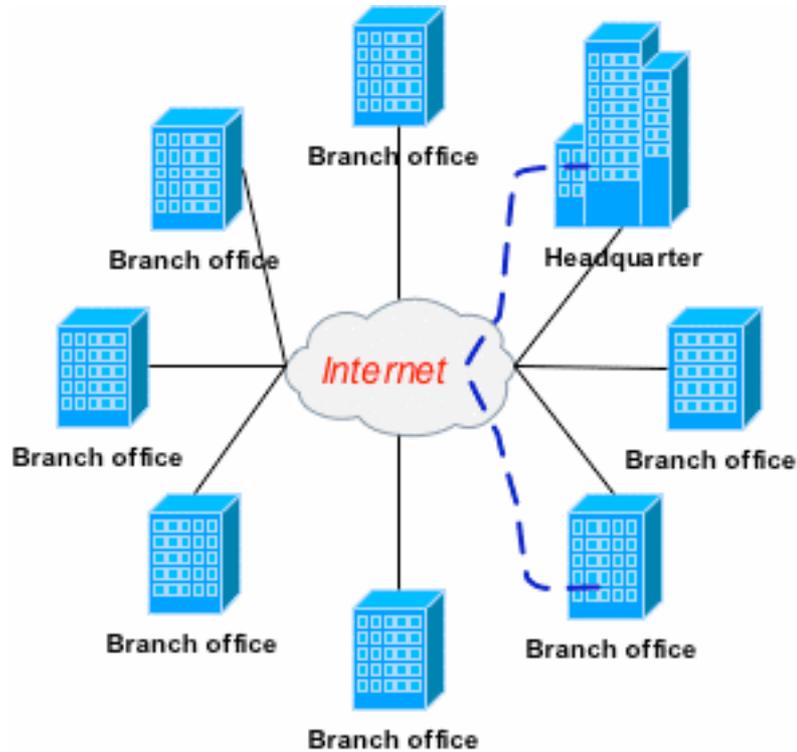


Figure 6: Interconnection - Example Company

All Linux Workstations, independent of whether they are in the headquarters or in the branch office, have the same distribution installed. The setup of the systems is standardized, which means that each system has the same look and feel to it.

The following table will describe what kind of Open Source software is required and has to be installed on the UNIX systems at the headquarters to provide the service.

Server	Open Source Software
Web	apache, openldap, pth, fsl, db, bzip2, gd, mysql, readline, ncurses, sharutils, pdflib, coreutils, jpeg, png, flex, curl, expat, zlib, pam, free-type, sed, bison, imap, openssl, gdbm, gettext, automake, autoconf, m4, grep, pcre, perl, gcc, binutils, make, libiconv, mhash, mm
Mail	imap, imapd, config, postfix, openldap, pth, mysql, procmail, postgresql, readline, gzip, zlib, flex, getopt, perl-time, perl-ds, perl-stats, perl-openpkg, whoson, sasl, openssl, fsl, groff, bison, m4, texinfo, ncurses, sharutils, grep, pcre, perl, db, gcc, binutils, make
DNS	bind, openssl, perl, gcc, binutils, make
ftp	lftp, readline, openssl, perl, gcc, binutils, ncurses, sharutils, libiconv, make,
Database	mysql, readline, zlib, perl, gcc, binutils, ncurses, sharutils, make

How OpenPKG can support IT business.

Server	Open Source Software
Mail relay	postfix, openldap, pth, mysql, procmail, postgresql, readline, gzip, zlib, flex, getopt, perl-time, perl-ds, perl-stats, perl-openpkg, whoson, sasl, openssl, gruff, texinfo, ncurses, sharutils, bison, m4, perl, grep, pcre, fsl, db, gcc, binutils, make, amavisd, perl-comp, lzo, sed, bison, m4, spamassassin, perl-dns, perl-net, perl-xml, expat, zlib, perl-parse, perl-crypto, openssl, fsl, perl-db, db, perl-conv, perl-mail, gzip, perl-util, bzip2, grep, pcre, diffutils, texinfo, ncurses, sharutils, perl-sys, perl-term, readline, perl-time, perl-ds, perl-stats, perl-openpkg, perl, gcc, binutils, make
Bastion	openssh, zlib, fsl, openssl, perl, gcc, binutils, make
Intranet	apache, openldap, pth, fsl, db, bzip2, gd, mysql, readline, ncurses, sharutils, pdflib, coreutils, jpeg, png, flex, curl, expat, zlib, pam, freetype, sed, bison, imap, openssl, gdbm, gettext, automake, autoconf, m4, grep, pcre, perl, gcc, binutils, make, libiconv, mhash, mm
Proxy	squid, fsl, perl, gcc, binutils, make
NFS	Service provide by the Operating System
Samba	samba, popd, openssl, perl, gcc, binutils, make
Monitoring	nagios, inetutils, coreutils, openssh, libedit, gd, jpeg, freetype, snmp, fping, traceroute, png, apache, flex, curl, sed, bison, m4, gdbm, perl-net, perl-xml, expat, perl-parse, libxslt, libxml, libiconv, perl-crypto, perl-mail, gzip, perl-util, bzip2, grep, diffutils, texinfo, ncurses, sharutils, perl-sys, perl-term, readline, nail, postfix, perl-time, perl-ds, perl-module, perl-stats, pcre, db, procmail, fsl, openssl, perl, gcc, zlib, binutils, make

Table 12: Software used for services at the headquarters – Example Company

Due to the fact that each server needs to be administered, the software installed on the bastion host is installed on all other hosts as well.

The IT environment shown in Figure 7, with the focus on the UNIX part of the company, represents the systems at the headquarters. This gives us a common understanding of what kinds of services are provided by UNIX systems.

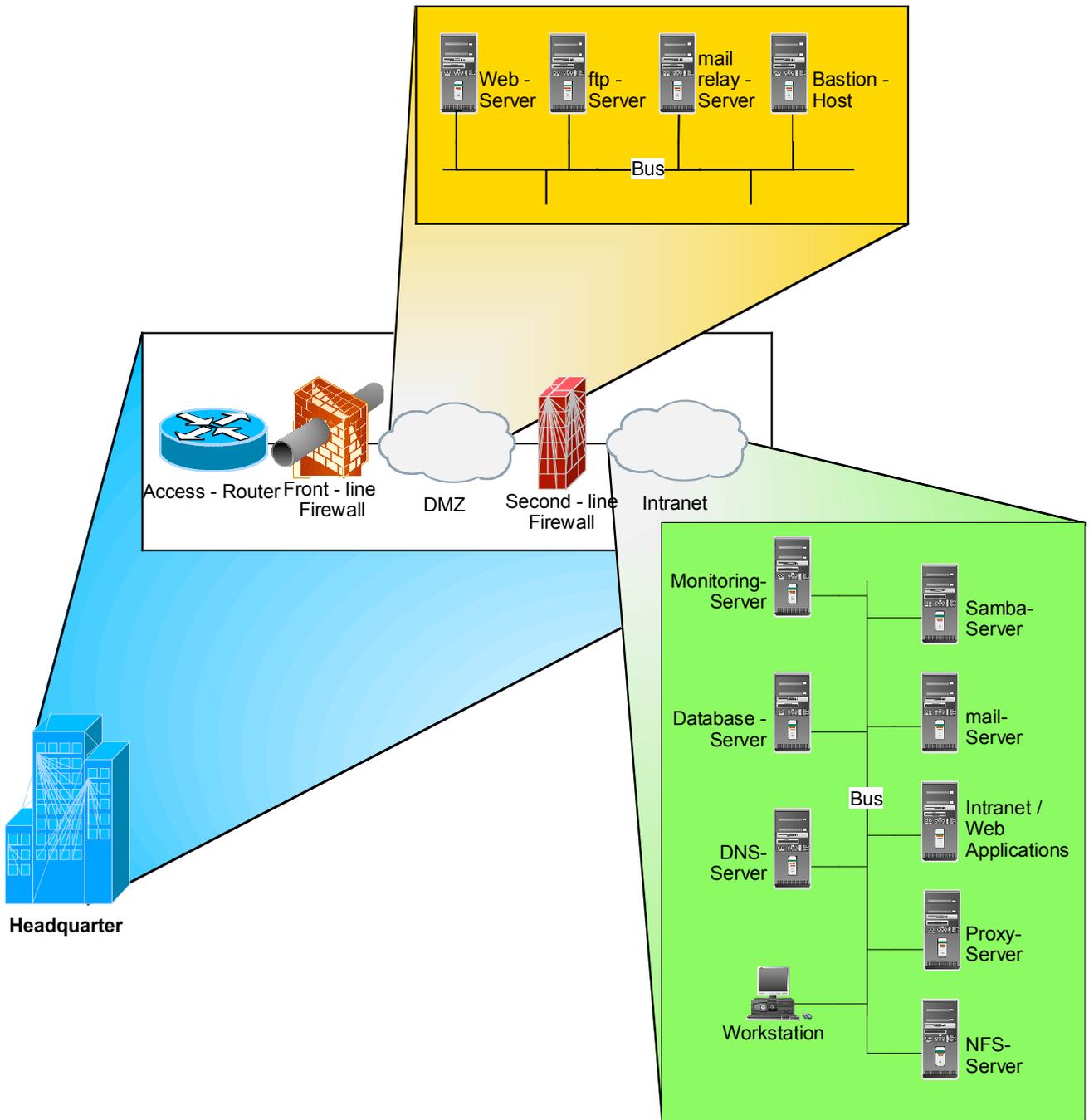


Figure 7: Headquarters - Example Company

The following table describes what kind of Open Source software is required and installed on the UNIX systems at branch offices to provide the service.

Server	Open Source Software
NFS	This service is provided by the Operating System
Samba	samba popl openssl perl gcc binutils make
Proxy	squid, fsl, perl, gcc, binutils, make
DNS	bind, openssl, perl, gcc, binutils, make

Table 13: Software used for services at branch offices – Example Company

How OpenPKG can support IT business.

Figure 8 shows the standardized UNIX environment at the branch offices.

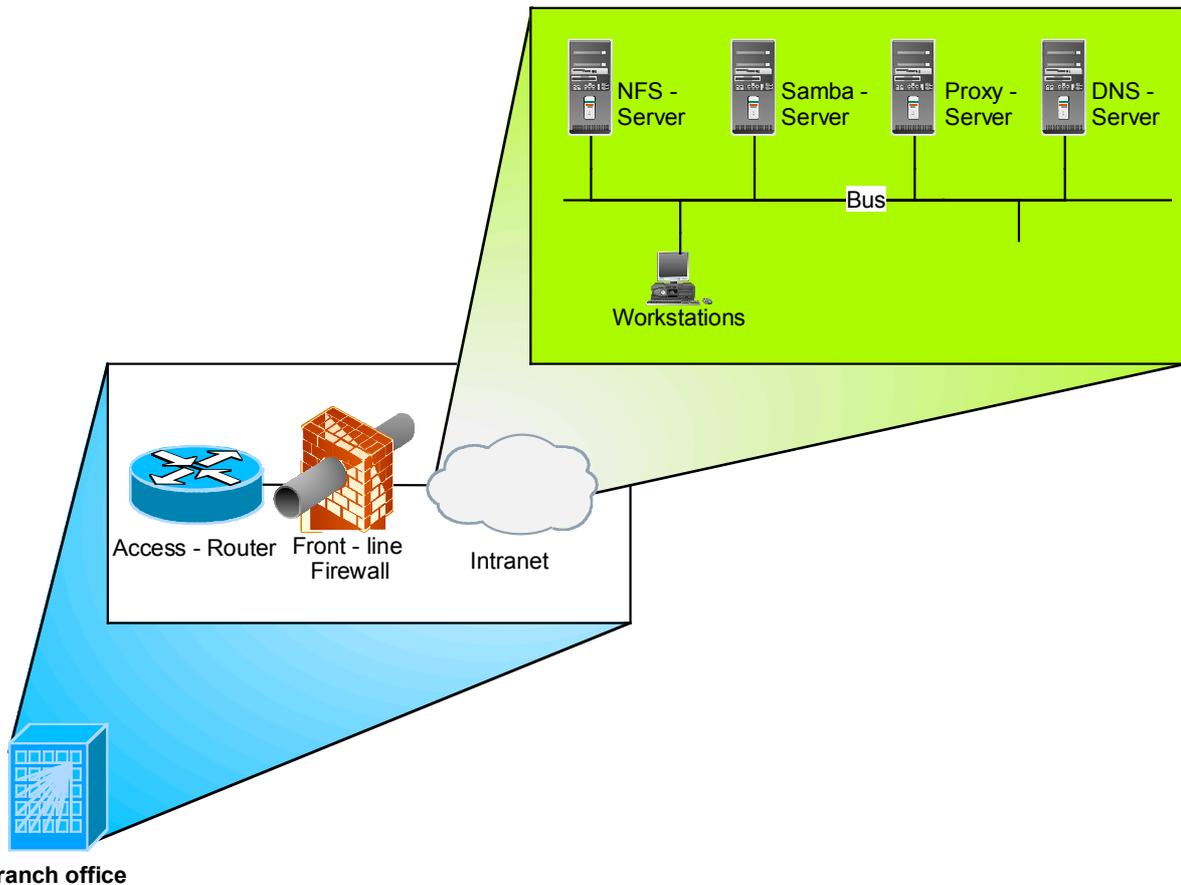


Figure 8: Branch office - Example Company

7.2 Vision³⁶

The vision of the company is:

We individualize services and products for future requirements.

7.3 Mission³⁷

The mission of the company is as follows:

Being reachable and providing the best services and products to customers 24x365 around the globe.

7.4 Shared Values³⁸

The shared values of the company are as follows:

Trust

Each related entity earns trust with our experience, truthfulness and reliability

Respect

We respect and care for our people, partners and customers, independent of their culture and religion.

Pride

We take great pride in providing customer satisfying products and services by our people and activities and products.

Involvement

We are actively involved in providing future products and services, to be profitable and achieve sustainable growth.

Loyalty

We are loyal to our corporation, customers and nation

Professionalism

We are innovative and proactive in striving for excellence.

Integrity

We are honest and conscientious to all related entities.

7.5 Balanced Score Cards³⁹

The corporate objectives are assumptions and shown in the following four balance sheets.

Financial Perspective								
ID	Strategic Objectives	Definition	Measure					Actual Value
			Target					
			06	07	08	09	10	
F-1	Increase Profit	ROI (%)	4	4.5	5	5.5	6	3.5
F-2	Revenue growth with existing customers	Revenue by minimum of 5.7% ROS (m€)	120	127	134	144	155	120
F-3	Revenue growth with new customers	Revenue by minimum of 4% ROS (m€)	5	5	6	6	7	4
F-4	Reduction of the financial leverage	Financial Leverage (%)	28	25	23	20	16	32

Figure 9: Financial Perspective - Example Company

Customer Perspective

ID	Strategic Objectives	Definition	Measure					Actual Value
			Target					
			06	07	08	09	10	
C-1	Increase customer satisfaction	Customer loyalty (%)	62	64	67	72	78	60
		Customer surveys (%)	57	60	64	69	75	55
C-2	Lower cost to customers	Price-performance ratio (Point/t€)	64	68	72	76	80	60
C-3	Faster solution delivery for customer issues.	Average ticket resolution time specified by the customer (hours)	7	6.5	6	5.5	5	8
C-4	Decrease Complaints	Complaint rate (%)	1.1	1.0	0.9	0.8	0.7	1.2

Figure 10: Customer Perspective - Example Company

Internal Perspective

ID	Strategic Objectives	Definition	Measure					Actual Value
			Target					
			06	07	08	09	10	
I-1	Increase Quality of our products	Failure rate	3σ	4σ	5σ	6σ	6σ	3σ
I-2	Increase internal service availability	Service availability hours (%)	99.9	99.95	99.99	99.995	99.999	99
I-3	Improve process responsiveness	Throughput improvement in relation to the past year (%)	10	9	8	7	6	0
I-4	Reduce installation time	Installation time in relation to the past year (%)	8	7	6	5	4	0
I-5	Reduce installation cost	Installation cost in relation to the past year (%)	8	7	6	5	4	0

Figure 11: Internal Perspective - Example Company

How OpenPKG can support IT business.

Learning and Growth Perspective

ID	Strategic Objectives	Definition	Measure					Actual Value
			Target					
			06	07	08	09	10	
L-1	Increase Employee satisfaction	Employee satisfaction (%)	62	64	66	68	70	60
L-2	Decrease fluctuation rate	Fluctuation rate (%)	14	13	12	11	10	15
L-3	Increase competence of all employees	Training days (days)	3	3.5	4	4.5	5	2.5
L-4	Create a continuous improvement culture	Ideas shared across the organization (%)	20	30	40	50	60	10

Figure 12: Learning and Growth Perspective - Example Company

7.6 Strategy Map

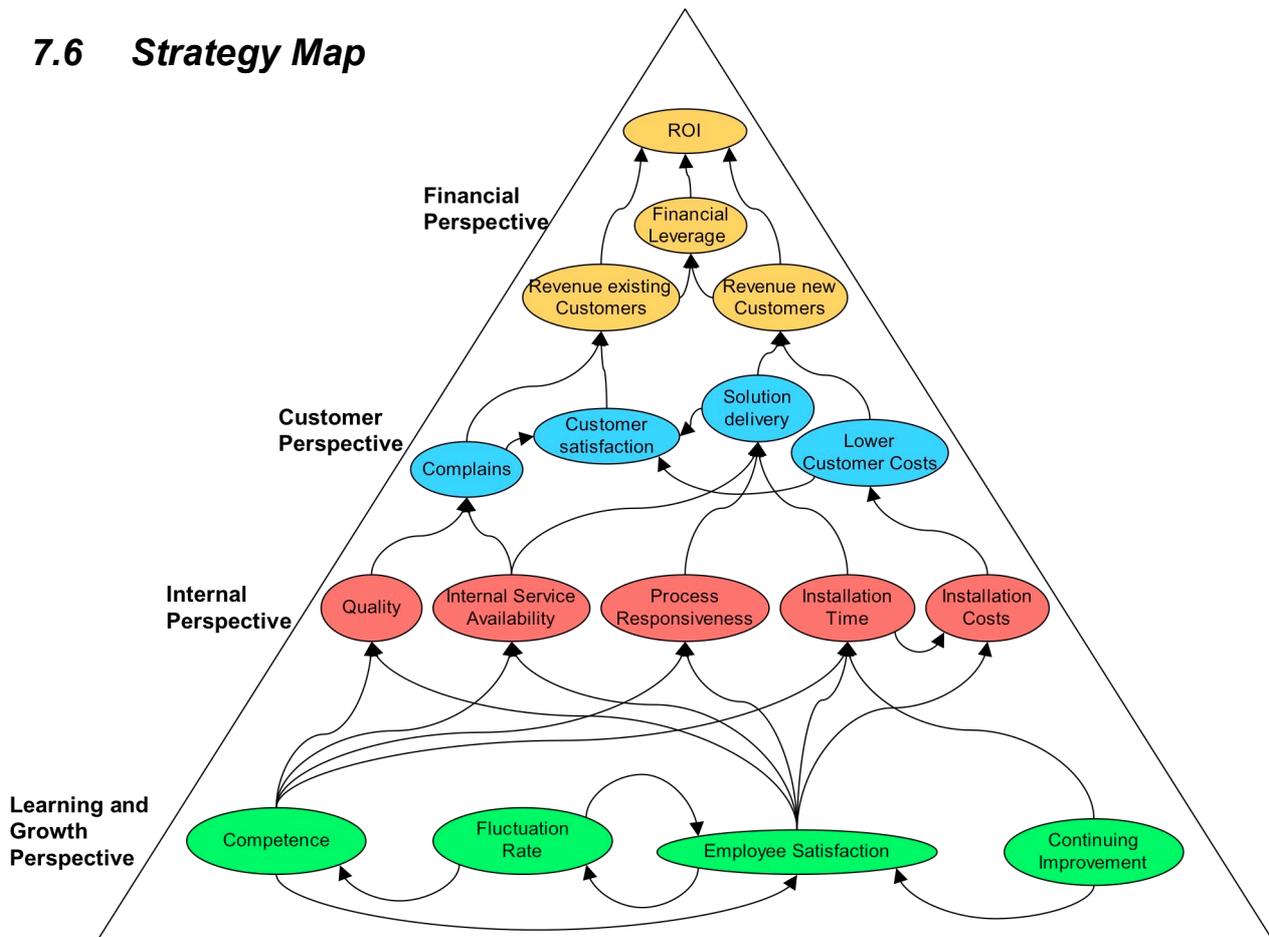


Figure 13: Strategy Map - Example Company

How OpenPKG can support IT business.

7.7 IT Governance

IT governance is an essential part of the corporate governance and the board of directors and management are responsible for it. IT governance consists of management, organizational structures and processes, and to ensure that IT is aligned with the corporate strategy and goals. IT governance has to ensure that expectations of the IT get fulfilled, IT resources get continually planned, steered and optimized, IT performance gets measured, and risks get minimized.⁴⁰

To build IT governance the following five interrelated decisions have to be made.

IT principle decision		
IT architecture decision	IT Infrastructure decision	IT investment and prioritization decision
	Business application needs	

Table 14: Key IT Governance Decisions⁴¹

The Example Company made the following decisions about the IT governance:

1. IT principle decision

- Enable the business.
- Ensure information integrity.
- Ensure information availability.
- Ensure consistent architecture.
- Create a common customer (internal/external) view.
- Utilize industry standards.
- Reuse before you buy; Use Open Source where it makes sense; buy before you build.
- Manage IT like an investment.

2. IT architecture decision

- Applications are standardized.
- Applications get interconnected by using XML.
- Reuse the applications before a new one gets implemented.
- Provide reliable applications.
- Centralized data stores are used.
- Increase availability.
- Enhance information integrity.

3. IT infrastructure decision

- Provide a network that is available 99.999% to ensure communication.
- Provide a telephone system with an availability of 99.999%.
- Provide a server infrastructure with an availability of 99.999%.
- Only standardized software and hardware get implemented.
- All systems have the latest security patches installed to increase security and minimize risks.
- All data is backed up; the restore is checked and finished in a short period of time.
- Remote server administration is done over a secure protocol.
- All systems are monitored.
- IT staff get trained and educated to perform the required tasks.
- Systems get replaced when they are outdated, when they no longer support the business or when consolidation makes sense.

4. Business application needs

- Whenever there is a requirement for an additional business application, a requirements document is needed and a solution has to be developed together with all related departments.
- Before a new application gets implemented, it has to be proven that an already implemented application is not able to provide the same kind of service.
- The IT department and the business unit requiring the service are responsible for the business needs fulfilment.
- If new applications can be more aligned with the business and can support it in a better way, a possible implementation has to be evaluated.
- New applications get implemented if they are more valuable than an existing one.

5. IT investment and prioritization decision

- Changes and enhancements to business processes that make them more efficient, faster, or cheaper have to be prioritized.
- Changes and enhancements to business processes for the entire enterprise have a higher priority than those to a business unit.
- All investments have to be in line with the corporate strategy and treated like an investment with a focus on the value to the business and the costs involved.

7.8 Implementation Plan

The shown implementation plan will concentrate on the realization of OpenPKG within the company. I will show the project plan and the costs involved for implementing this solution. I will provide two options, the first of which will show what kind of costs are involved by doing it in-house, the second will show how much it will cost if it gets implemented through an external partner. For the calculation I assume that per month 8 different kinds of software have to be updated on each server just to have the latest security updates installed. An upgrade of the software is performed once a year.

7.8.1 Project Phases

Both the company and the external partner would like to implement OpenPKG in phases. Due to the fact that the project plan in both cases is sizeable, I will only show the project phases, and this is the same for both. More detailed information about the project, including all tasks and the costs involved, can be seen in the appendix.

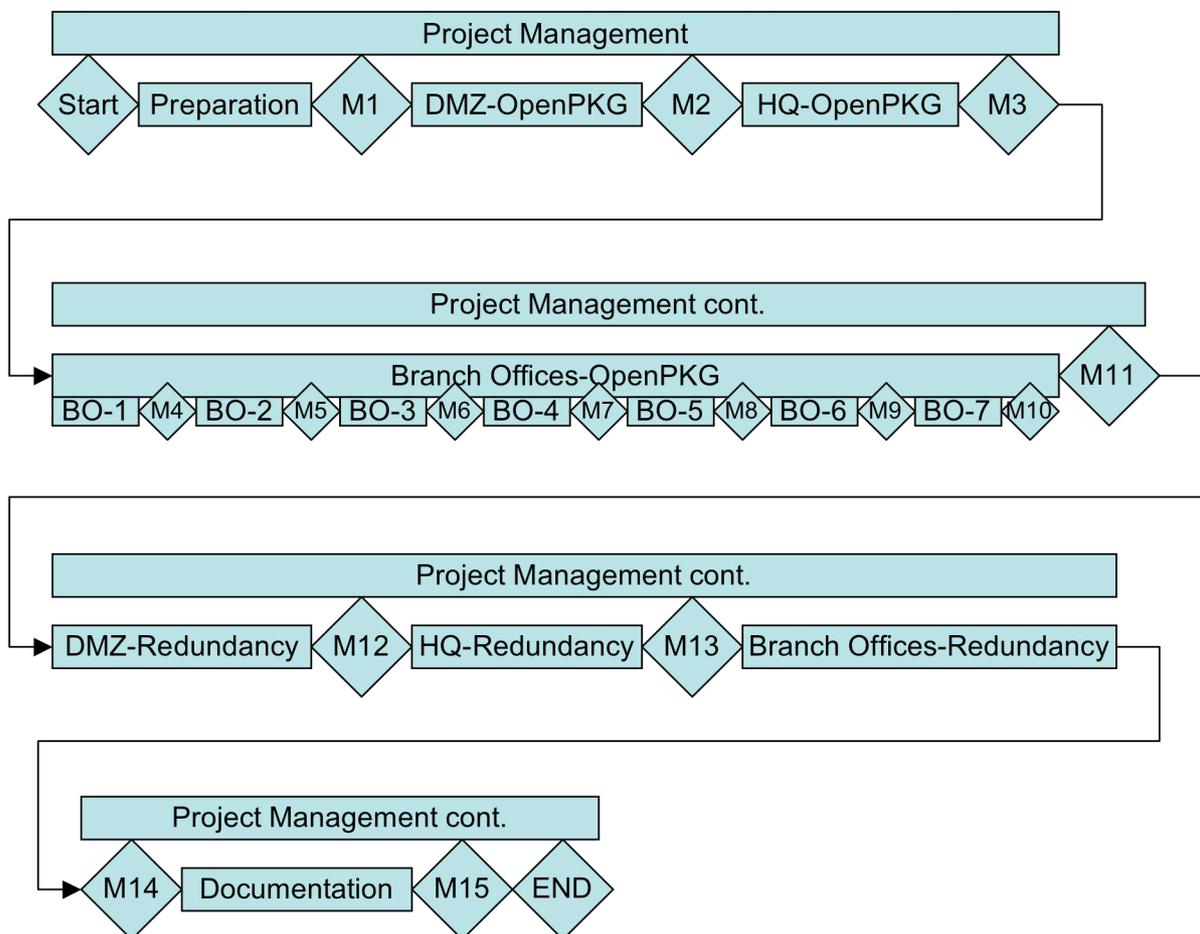


Figure 14: Project Phases - Example Company

7.8.2 Costs

As a basis for the following calculation I decided to use the following cost rates per engineer:

ID	Name	Costs per Hour
E1	George	62.00 €/hr
E2	James	62.00 €/hr
E3	Jan	62.00 €/hr
E4	Jim	62.00 €/hr
E5	John	62.00 €/hr
E6	Manager IT	75.00 €/hr
E7	OpenPKG-Eng-1	85.00 €/hr
E8	OpenPKG-Eng-2	85.00 €/hr
E9	Project Manager	70.00 €/hr
E10	Robert	62.00 €/hr

Table 15: Engineer's cost table - Example Company

Each month the following staff costs shown in table 17 for operating and maintaining the Unix server exist. With the money involved only the continuity of the business is ensured without further improvement. In my calculation 6 engineers are full-time, maintaining the Unix-Server on 220 working days a year before implementing OpenPKG.

Engineers	Costs per Hour	Hours per Day	Working days per Month	Costs per Month
6	€62.00	8	21	€62,496.00

Table 16: UNIX-Server Maintenance Costs per month before OpenPKG - Example Company

Based on the former case studies, after implementing OpenPKG, the expected staff costs for operating and maintaining the Unix servers will be less expensive. My calculation will be a pessimistic one and I assume that two full time equivalents will still be required to perform the required tasks. The remaining four engineers are now able to concentrate on further developing and improving the environment to better support the business.

Engineers	Costs per Hour	Hours per Day	Working days per Month	Costs per Month
2	€62.00	8	21	€20,832.00

Table 17: UNIX-Server Maintenance Costs per month after OpenPKG - Example Company

In-house

Table 19 shows the costs involved for implementing OpenPKG as an in-house project on a monthly basis. Project management will be done in-house, because there are resources that can be used for this implementation project.

Month	Organizational Costs (€)	Implementation Costs (€)	Total Costs (€)
1	83,880.00	0.00	83,880.00
2	11,760.00	10,416.00	22,176.00
3	11,760.00	16,368.00	28,128.00
4	11,760.00	16,368.00	28,128.00
5	11,760.00	19,592.00	31,352.00
6	11,760.00	12,896.00	24,656.00
7	11,760.00	10,416.00	22,176.00
8	11,760.00	10,416.00	22,176.00
9	23,840.00	5,456.00	29,296.00
Summary of Total Costs			291,968.00

Table 18: Monthly project costs in-house - Example Company

With external partner

Table 20 shows the costs involved for implementing OpenPKG within a project together with an external partner. Project management will be done in-house, because there are resources that can be utilized for this implementation project.

Month	Organizational Costs (€)	Implementation Costs (€)	Total Costs (€)
1	111,080.00	0.00	111,080.00
2	12,040.00	34,092.00	46,132.00
3	11,760.00	29,940.00	41,700.00
4	22,600.00	27,120.00	49,720.00
5	12,240.00	0.00	12,240.00
Summary of Total Costs			260,872.00

Table 19: Monthly project costs with external partner - Example Company

If you do not care about internal costs and you do not have cross charges implemented in the company, table 21 shows the monthly payment to the external partner for the project period.

Month	Costs for External Partner (€)
1	27,200.00
2	11,900.00
3	12,920.00
4	12,240.00
5	6,800.00
Total Costs for External Partner (€)	
	71,060.00

Table 20: Costs for External Partner - Example Company

Figure 15 shows you the development of the costs and what kind of savings you can achieve by implementing OpenPKG. As you can see in the figure, by implementing OpenPKG as an in-house project, in my example it will take about 9 months, after which time the company will achieve cost savings of about €41,000 per month just by performing the same tasks as before. Implementing OpenPKG together with an external partner will cost you about €71,000, but will only take 4.5 months. This means that cost savings of about €41,000 will happen at an earlier stage. It only takes about 2 months to get the

investment back through cost savings, and your engineers can concentrate on other business related tasks earlier.

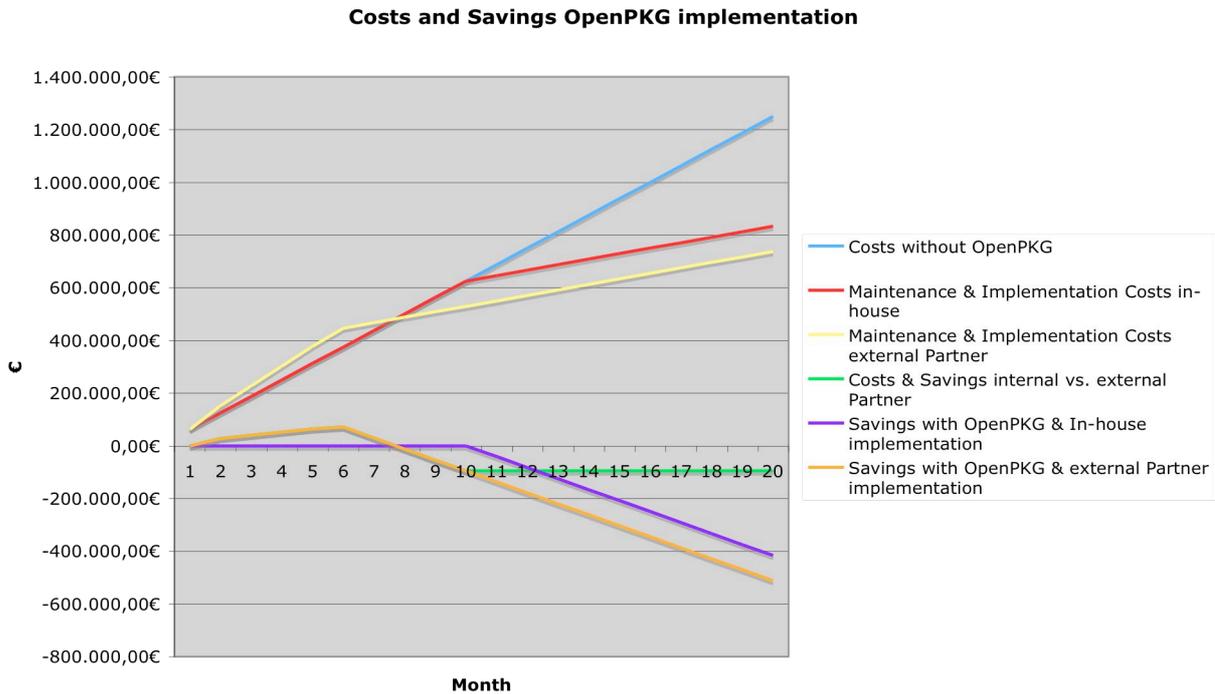


Figure 15: Costs and savings overview OpenPKG implementation - Example Company

7.8.3 Implementation

During the implementation of OpenPKG, the service provided to the business is ensured. All tasks mentioned are mostly performed during office hours; only the switch to the new service provided by OpenPKG, which is the step in the testing part, will be done outside of business hours. Within the in-house project the engineers train themselves by implementing OpenPKG, where naturally each task will take a bit longer in the beginning until they get experienced with it.

In the case that it gets implemented together with an external partner, training for the engineers will be held at the beginning of the project to ensure service support throughout the implementation of OpenPKG and beyond. The engineers gain experience of working with OpenPKG step by step while testing the service. The issues solved and how it supports the business will be shown in the next section.

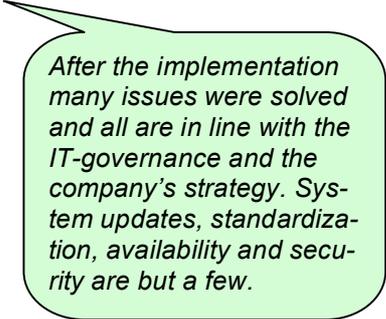
7.9 Issues solved after OpenPKG implementation

7.9.1 System Updates

You might have seen above already that about 6 engineers are working on about 40 servers to ensure the provision of secure and up-to-date systems. When using OpenPKG only two engineers will be required to ensure up-to-date systems with the latest security patches installed, as was shown in the case with Portland University in section 6.1. There are several reasons why there is more time required for installing Open Source software without a packaging mechanism, which has been shown with the telecommunications case in section 6.2 and pointed out in section 5.1.2. This is only

one part where the engineers become more efficient and there are a few more, which will also be discussed.

Installing a new version of the software can take a very long time if you do not have any kind of documentation about how the installed software got compiled. The OpenPKG packaging mechanism solves this issue as detailed in section 5.1.2. If a newer version of the software is available, it can be prepared in advance and installed whenever it is required. That is the same for both installing Open Source software without a packaging mechanism and with it. The point is how long does it take to have the former version of the software reinstalled in the case of a problem? By having a packaging mechanism, only the former package has to be reinstalled. Such a solution saves a lot of time (I-4), and increases the service availability (I-2), which are points in the IT infrastructure decision and are strategic objectives in the BSC.



After the implementation many issues were solved and all are in line with the IT-governance and the company's strategy. System updates, standardization, availability and security are but a few.

Most of these tasks have to be performed during the weekend and, therefore, an additional fee for the engineers has to be paid. The need for shorter installation times (I-4) reflects in fewer installation costs (I-5) for the service, which drives down IT costs and makes the investment in IT more valuable as decided by IT governance.

When compiling the software yourself and installing it straight onto the system without a packaging mechanism, in-depth software development knowledge is required. When using OpenPKG this in-depth knowledge of software development is no longer required for installing software, as was shown in the case of the telecommunications company in section 6.2. This wealth of knowledge can be used for continuing service improvement. Again, this makes the IT department more efficient and valuable, which goes in line with the government decision about IT investment. Another aspect is quality improvement (I-1), which is a strategic objective.

7.9.2 Standardization

Standardization is another important point for a company. With standardized software IT is able to become more efficient because only one kind of software is used to provide one kind of service. Standardization is much easier with OpenPKG, because you are able to specify the packages required for providing a service and do not have many small pieces of software. This also goes in line with the IT governance decision for consistent architecture and the strategic objective quality (I-1) within the BSC.

7.9.3 Availability

The next important point I would like to mention is the availability of services. At the moment services are only available once because it would be too complex to have services installed and maintained more often. With OpenPKG having a service installed more than once on one system or on different systems is not a problem. The feature that enables you to have more than one instance installed on a server solves this. Installing the software for the service on more than one system enables the company to switch the service from one server to another in the case of hardware failure or a system update. This increases the availability of the services and goes in line with the IT

governance and BSC (quality (I-1), internal service availability (I-2), complaints (C-4), and continuing improvement (L-4)).

7.9.4 Security

Security is another important point for the company. Whenever there is a security leak in the software it requires an update to apply the latest security patches. If all systems are affected by it, it requires a lot of manpower to perform this within a certain period of time before implementation of OpenPKG. Similar to a case I mentioned earlier, it can take many resources to perform this task. With OpenPKG the same task can be performed in a much shorter time, as detailed previously. It is in line with the government decisions about the level of IT investments and the infrastructure decision about security. It will support the strategic objectives, increase quality (I-1), internal service availability (I-2), decrease complaints (C-4), and it will increase customer satisfaction (C-1) as well as employee satisfaction (L-1) (greater ability to plan work), as detailed in the BSC.

7.9.5 Complexity of the Environment

The complexity of the environment will more or less stay the same, but with the use of OpenPKG it gets more structured. The structure, based on the company requirements, will be the same on all systems. This allows the company to standardize the installation paths, which enables the engineer to work more efficiently. This reduces the costs for IT and frees up some resources. The freed up resources can work on the backlog of requests to improve employee satisfaction (L-1), which has an impact on the fluctuation rate (L-2), both of which were mentioned in the BSC.

7.9.6 Packaging

Software packaging is a very important point and has connections to most of the other points mentioned in the sub-section of section 7.9. All details will be given whenever packaging is mentioned.

7.9.7 Documentation

Due to SOX and Basel II (which may be enhanced with regulations for the IT in the future) documentation of IT is important. With OpenPKG, documentation is easier than if you just use Open Source Software because each file is referenced in the package database. From this database the required information can be queried and put into the documentation. Only files where changes were made have to be mentioned additionally and the changes have to be documented. Better Basel II ratings will reflect in lower interest and this has a direct effect on the financial leverage (F-4) because more repayment can be performed. Documentation brings me to the newcomer, for whom documentation about the IT systems is a useful point of reference to start in a company.

7.9.8 Newcomer Issue

Having up-to-date documentation provides the newcomer with the required information when starting at the company. Additionally, it provides the newcomer with the required information to become more productive earlier. Even if there is no documentation for the IT systems, providing the newcomer with the standards at least helps. The newcomer only has to learn how to work with OpenPKG, which is done within one week (basic un-

derstanding). This helps them have a smooth start in a new company, which increases employee satisfaction (L-1) and is less costly than just using Open Source software.

7.9.9 Migration & Consolidation & Platform Independence

Within the Example Company there is no reason for migration and consolidation, but in a period of time there might be one. This can happen because of the reasons pointed out in sections 3.3, 3.4 and 4.3. If such a situation happens, the company is prepared for it and just has to perform the required tasks and is not hindered from doing it and, in addition, it will save time and money. Another benefit for the company is that it no longer depends on the hardware supplier. The company with OpenPKG can change it more easily because of the cross-platform functionality provided by OpenPKG. This increases the flexibility of the company and puts the company in a much better negotiating position with the supplier. This reflects in a more efficient IT investment with less installation costs, which goes in line with the IT governance decision.

7.9.10 Support

Another point that applies to the company is that it can get support for OpenPKG. This is very important because in the case of a failure with Open Source software it is often very hard to get support for all the different kinds of software. However, for the Open Source software provided by OpenPKG it can be bought at OpenPKG GmbH. In the case of an internally unsolvable failure this decreases downtime, because you know whom to call for support and the OpenPKG engineers will perform the required tasks. This will support the availability of the internal services (I-2), which is a strategic objective pointed out in the BSC.

7.9.11 Development

The Example Company, for example, requires Open Source Software to be adjusted to their requirements. This task can be performed in-house or by an external partner such as OpenPKG GmbH. The Example Company agreed on a contract with the external partner who takes care of the company specific changes made to the software to make sure they get included in newer releases of the software and that it is only available to the Example Company. This makes the IT of the company more efficient (I-3) and enables them to further improve (L-4), which are two strategic objectives.

7.9.12 Consulting & Training

The last benefit I would like to mention is that whenever there is a need for consulting and training, there is a partner where the company can get it specified to their needs.

7.10 Summary

This example shows that many issues that hinder the company from doing business, as planned in their strategy, can be solved by implementing OpenPKG. This shows, in addition, in which areas OpenPKG can help and what kind of impact it has on the corporate strategy. What's more, it saves money as well.

8 Final Conclusion

It is now time to sum up the document. In the beginning we had an introduction to UNIX and UNIX-compatible systems and how they have developed over the years. We clarified the term Open Source and pointed out which license terms have to be fulfilled in order for software to be treated as Open Source. We discussed free software and freeware and have seen the differences between them. Following that, we looked at OpenPKG and clarified OpenPKG.org – the project, OpenPKG.net – the foundation, and OpenPKG.com, the commercial company. With this common understanding we proceeded to discuss UNIX issues.

Within the UNIX issue section, we discussed the availability of software and divided software into three major branches, namely, commercial, free and Open Source software, and home-grown software and showed the advantages and disadvantages of each. Understanding these categories provides critical input for a decision-maker to decide what kind of software should be used in his or her company.

The next topic we discussed was the difference between the various kinds of UNIX. Understanding these differences can contribute to decisions about computer platform choices. To choose the right platform at the beginning of building or rebuilding an environment is a very important point.

Further on, we discussed migration and consolidation, which often come along together and are related to each other. These two issues might pop up more often in the future because companies that have invested only a minimum in IT might have to migrate and consolidate their systems in order to get more efficient and drive down their costs.

Dependencies were the next topic discussed. Here, we concentrated on the dependencies on other software, which is an important point for ensuring a proper working software package and application. We also pointed to security issues, which are very important, because most of the attacks on servers come from inside the company, not from the outside.

After we finished with the issues concerning the different kinds of UNIX, we turned to issues regarding Open Source software. We started with the support issue, which is a problem for some companies if it is absent. We continued with business continuity, where we elaborated on several key issues. Each issue pointed out can be seen as a risk or an opportunity depending on the person's point of view.

The next important topic discussed was the platform. When changing the platform, you have to consider that the required software has to work on the new platform. Getting Open Source software to do this often involves a lot of work.

Finally, we discussed security for Open Source, which is more or less the same as for UNIX.

In the next section, we discussed what OpenPKG as a solution can do to solve these issues. We started with the non-commercial part of OpenPKG and finished off with the

commercial version. Within the non-commercial section, we showed how OpenPKG can help to solve the issue of the differences between UNIX systems, which can be done by its cross-platform functionality and by providing a standardized environment.

The next subject discussed was packaging of software, which is very often a problem faced when using Open Source. OpenPKG provides binary packages that can be installed on the systems without compilation. Additionally, OpenPKG provides source packages that have to be compiled before you can use the software. For the compilation process, OpenPKG provides a standardized environment. The packaging mechanism helps to achieve SOX compliance and may get a slightly better Basel II rating because of better documentation of the IT environment. Point migration was solved, on the one hand, by having packages and, on the other, by providing cross-platform functionality.

After the solution for migration we continued with the solution for consolidating systems, which includes the multi-instance feature provided by OpenPKG. This easily allows for consolidation of different systems into one system. OpenPKG solves the dependency issue by providing clear information on what is required. We further showed how the platform issue is solved using OpenPKG. We continued by showing how you can solve your issues with Open Source and business continuity because Open Source follows a give and take principal.

The OpenPKG security solution was the last non-commercial point we showed, where we also mentioned how your company can benefit from it.

After the non-commercial section we turned to the commercial software and showed what OpenPKG GmbH offers, and described each commercial service (consulting, development, deployment, training, support, maintenance, operations, and project management).

We continued with the case studies of Portland State University, an international telecommunications company, and the Fraunhofer Institut Information- und Datenverarbeitung. In these cases, we showed that companies using OpenPKG benefit from it significantly and that the solutions provided by OpenPKG are right. The Portland State University case showed that thirteen issues were resolved and that there were additional benefits as well. Only the too fast release cycle of OpenPKG caused a problem for the university.

With the telecommunications company we showed that seven issues were resolved and additional benefits were gained by using OpenPKG. We also showed the issues the telecommunications company had when using OpenPKG, including additional disk space requirement, no available support, etc.

Finally, with the Fraunhofer Institute case, we detailed how they were easily able to migrate to a platform with a much better price/performance ratio and that they solved several dependency problems as well. However, the Fraunhofer Institute has some issues with OpenPKG, such as minimum support, untidy documentation of the OpenPKG tools, and no support for Solaris 10 on ix86 architecture.

Finally, we outlined an Example Company that would like to implement OpenPKG. We clearly showed how OpenPKG supports the business, as described in the Balanced Score Card and IT Governance. Additionally, we showed the money involved to imple-

ment OpenPKG and what amount of money and resources can be saved or used in a different manner by using OpenPKG.

Last of all, I would like to point out that trademarks, service marks, collective marks, design rights, personality rights or similar rights that are mentioned, used or cited are the property of their respective owners.

9 Appendices

9.1 Task list in-house project

The following project plan will show all tasks and, in addition, the associated costs.

ID	Name	Duration	Work	Resource Name	Predecessors	Costs
T11	Start		0 hrs	Manager IT		
T12	Preparation	21,d				
T13	Project start Workshop	2,d	16 hrs	Jim, John, Jan, George, James, Robert, Manager IT	1	7.152,00 €
T14	Analysis	10,d	80 hrs	Jim, John, Jan, George, James, Robert, Manager IT	3	35.760,00 €
T15	Requirement definition	5,d	40 hrs	IT	4	17.880,00 €
T16	Communication to User	1,d	8 hrs	Manager IT Jim, John, Jan, George, James, Robert, Manager	5	600,00 €
T17	Standard definition	3,d	24 hrs	IT	6	10.728,00 €
T18	M1 - Preparation finished	,d	0 hrs	Project Manager	7	0,00 €
T19	Phase 1 - DMZ - OpenPKG	19,d				
T110	Bastion Host	1,d	8 hrs	John	8	496,00 €
T111	Test - Bastion Host	1,d	8 hrs	Jim	10	496,00 €
T112	ftp Server	1,d	8 hrs	Jim	11	496,00 €
T113	Test - ftp Server	1,d	8 hrs	John	12	496,00 €
T114	Web Server	4,d	32 hrs	Jan	13	1.984,00 €
T115	Test - Web Server	4,d	32 hrs	George	14	1.984,00 €
T116	mail relay server	5,d	40 hrs	James	15	2.480,00 €
T117	Test - Mail relay Server	2,d	16 hrs	Robert	16	992,00 €
T118	M2 - Phase 1 finished	,d	0 hrs	Project Manager	17	0,00 €
T119	Phase 2 - HQ - OpenPKG	29,d				
T120	Intranet Server	4,d	32 hrs	George	18	1.984,00 €
T121	Test - Intranet Server	4,d	32 hrs	Jan	20	1.984,00 €
T122	Samba Server	2,d	16 hrs	Robert	18	992,00 €
T123	Test - Samba Server	2,d	16 hrs	James	22	992,00 €
T124	NFS Server	2,d	16 hrs	Jim	21	992,00 €
T125	Test - NFS Server	1,d	8 hrs	John	24	496,00 €
T126	Proxy Server	2,d	16 hrs	George	25	992,00 €
T127	Test - Proxy Server	1,d	8 hrs	Jan	26	496,00 €
T128	DNS Server	2,d	16 hrs	Robert	23	992,00 €
T129	Test - DNS Server	1,d	8 hrs	James	28	496,00 €
T130	Database Server	2,d	16 hrs	John	29	992,00 €
T131	Test - Database Server	3,d	24 hrs	Jim	30;24	1.488,00 €
T132	Monitoring Server	5,d	40 hrs	Jan	31;27	2.480,00 €
T133	Test - Monitoring Server	3,d	24 hrs	George	32	1.488,00 €
T134	Mail Server	5,d	40 hrs	James	33	2.480,00 €
T135	Test Mail Server	2,d	16 hrs	Robert	34	992,00 €
T136	M3 - Phase 2 finished	,d	0 hrs	Project Manager	35	0,00 €
T137	Phase 3 - Branch Offices - OpenPKG	43,d				
T138	Branch Office 1	7,d				
T139	DNS-Server	2,d	16 hrs	John	36	992,00 €
T140	Test DNS-Server	1,d	8 hrs	Jim	39	496,00 €
T141	Proxy-Server	2,d	16 hrs	Jan	36	992,00 €
T142	Test Proxy-Server	1,d	8 hrs	George	41	496,00 €

How OpenPKG can support IT business.

TI43	NFS-Server	2,d	16 hrs	James	40	992,00 €
TI44	Test NFS-Server	1,d	8 hrs	Robert	43	496,00 €
TI45	Samba-Server	2,d	16 hrs	George	42	992,00 €
TI46	Test Samba-Server	2,d	16 hrs	Jan	45	992,00 €
TI47	M4 - Branch Office 1 finished	,d	0 hrs	Project Manager	46;44	0,00 €
TI48	Branch Office 2	7,d				
TI49	DNS-Server	2,d	16 hrs	Robert	47	992,00 €
TI50	Test DNS-Server	1,d	8 hrs	James	49	496,00 €
TI51	Proxy-Server	2,d	16 hrs	Jim	47	992,00 €
TI52	Test Proxy-Server	1,d	8 hrs	John	51	496,00 €
TI53	NFS-Server	2,d	16 hrs	Jan	50	992,00 €
TI54	Test NFS-Server	1,d	8 hrs	George	53	496,00 €
TI55	Samba-Server	2,d	16 hrs	James	52	992,00 €
TI56	Test Samba-Server	2,d	16 hrs	Robert	55	992,00 €
TI57	M5 - Branch Office 2 finished	,d	0 hrs	Project Manager	56;54	0,00 €
TI58	Branch Office 3	7,d				
TI59	DNS-Server	2,d	16 hrs	John	57	992,00 €
TI60	Test DNS-Server	1,d	8 hrs	Jim	59	496,00 €
TI61	Proxy-Server	2,d	16 hrs	George	57	992,00 €
TI62	Test Proxy-Server	1,d	8 hrs	Jan	61	496,00 €
TI63	NFS-Server	2,d	16 hrs	Robert	60	992,00 €
TI64	Test NFS-Server	1,d	8 hrs	James	63	496,00 €
TI65	Samba-Server	2,d	16 hrs	Jim	62	992,00 €
TI66	Test Samba-Server	2,d	16 hrs	John	65	992,00 €
TI67	M6 - Branch Office 3 finished	,d	0 hrs	Project Manager	64;66	0,00 €
TI68	Branch Office 4	6,d				
TI69	DNS-Server	1,5d	12 hrs	Jan	67	744,00 €
TI70	Test DNS-Server	1,d	8 hrs	George	69	496,00 €
TI71	Proxy-Server	1,5d	12 hrs	James	67	744,00 €
TI72	Test Proxy-Server	1,d	8 hrs	Robert	71	496,00 €
TI73	NFS-Server	1,5d	12 hrs	John	70	744,00 €
TI74	Test NFS-Server	1,d	8 hrs	Jim	73	496,00 €
TI75	Samba-Server	1,5d	12 hrs	George	72	744,00 €
TI76	Test Samba-Server	2,d	16 hrs	Jan	75	992,00 €
TI77	M7 - Branch Office 4 finished	,d	0 hrs	Project Manager	74;76	0,00 €
TI78	Branch Office 5	6,d				
TI79	DNS-Server	1,5d	12 hrs	Robert	77	744,00 €
TI80	Test DNS-Server	1,d	8 hrs	James	79	496,00 €
TI81	Proxy-Server	1,5d	12 hrs	Jim	77	744,00 €
TI82	Test Proxy-Server	1,d	8 hrs	John	81	496,00 €
TI83	NFS-Server	1,d	8 hrs	Jan	80	496,00 €
TI84	Test NFS-Server	1,d	8 hrs	George	83	496,00 €
TI85	Samba-Server	1,5d	12 hrs	James	82	744,00 €
TI86	Test Samba-Server	2,d	16 hrs	Robert	85	992,00 €
TI87	M8 - Branch Office 5 finished	,d	0 hrs	Project Manager	84;86	0,00 €
TI88	Branch Office 6	5,d				
TI89	DNS-Server	1,d	8 hrs	John	87	496,00 €
TI90	Test DNS-Server	1,d	8 hrs	Jim	89	496,00 €
TI91	Proxy-Server	1,d	8 hrs	George	87	496,00 €
TI92	Test Proxy-Server	1,d	8 hrs	Jan	91	496,00 €
TI93	NFS-Server	1,d	8 hrs	Robert	90	496,00 €
TI94	Test NFS-Server	1,d	8 hrs	James	93	496,00 €
TI95	Samba-Server	1,d	8 hrs	Jim	92	496,00 €
TI96	Test Samba-Server	2,d	16 hrs	John	95	992,00 €
TI97	M9 - Branch Office 6 finished	,d	0 hrs	Project Manager	94;96	0,00 €
TI98	Branch Office 7	5,d				
TI99	DNS-Server	1,d	8 hrs	Jan	97	496,00 €
TI100	Test DNS-Server	1,d	8 hrs	George	99	496,00 €
TI101	Proxy-Server	1,d	8 hrs	James	97	496,00 €
TI102	Test Proxy-Server	1,d	8 hrs	Robert	101	496,00 €
TI103	NFS-Server	1,d	8 hrs	John	100	496,00 €
TI104	Test NFS-Server	1,d	8 hrs	Jim	103	496,00 €
TI105	Samba-Server	1,d	8 hrs	George	102	496,00 €
TI106	Test Samba-Server	2,d	16 hrs	Jan	105	992,00 €
TI107	M10 - Branch Office 7 finished	,d	0 hrs	Project Manager	104;106	0,00 €
TI108	M11 - Phase 3 finished	,d	0 hrs	Project Manager	107	0,00 €
TI109	Phase 4 - DMZ Availability	9,d				

TE6	Communication to User	1,d	8 hrs	Manager IT Jim, John, Jan, George, James, Robert, Manager IT, OpenPKG-Engineer- 1, OpenPKG-	5	600,00€
TE7	Standard definition	3,d	24 hrs	Engineer-2 Jim, John, Jan, George, James, Robert, OpenPKG- Engineer-1, O- penPKG-Engineer-2	6	14.808,00€
TE8	OpenPKG Training	3,d	24 hrs		7	13.008,00€
TE9	Preparation finished		0 hrs	Project Manager	8	0,00€
TE10	Phase 1 - DMZ - OpenPKG	11,d				
TE11	Bastion Host	,5d	4 hrs	OpenPKG-Eng-1	9	340,00€
TE12	Test - Bastion Host	,5d	4 hrs	Jim	11	248,00€
TE13	ftp Server	,5d	4 hrs	OpenPKG-Eng-2	12	340,00€
TE14	Test - ftp Server	,5d	4 hrs	James	13	248,00€
TE15	Web Server	1,d	8 hrs	OpenPKG-Eng-1	14	680,00€
TE16	Test - Web Server	4,d	32 hrs	John	15	1.984,00€
TE17	mail relay server	2,d	16 hrs	OpenPKG-Eng-2	16	1.360,00€
TE18	Test - Mail relay Server	2,d	16 hrs	George	17	992,00€
TE19	Phase 1 finished		0 hrs	Project Manager	18	0,00€
TE20	Phase 2 - HQ - OpenPKG	7,5d				
TE21	Intranet Server	1,d	8 hrs	OpenPKG-Eng-1	19	680,00€
TE22	Test - Intranet Server	4,d	32 hrs	Robert	21	1.984,00€
TE23	NFS Server	,5d	4 hrs	OpenPKG-Eng-1	21	340,00€
TE24	Test - NFS Server	1,d	8 hrs	Jim	23	496,00€
TE25	DNS Server	,5d	4 hrs	OpenPKG-Eng-1	23	340,00€
TE26	Test - DNS Server	1,d	8 hrs	John	25	496,00€
TE27	Monitoring Server	2,d	16 hrs	OpenPKG-Eng-1	25	1.360,00€
TE28	Test - Monitoring Server	3,d	24 hrs	Jan	27;30	1.488,00€
TE29	Samba Server	1,d	8 hrs	OpenPKG-Eng-2	19	680,00€
TE30	Test - Samba Server	2,d	16 hrs	Jan	29	992,00€
TE31	Proxy Server	,5d	4 hrs	OpenPKG-Eng-2	29	340,00€
TE32	Test - Proxy Server	1,d	8 hrs	James	31	496,00€
TE33	Database Server	2,d	16 hrs	OpenPKG-Eng-2	31	1.360,00€
TE34	Test - Database Server	3,d	24 hrs	George	33	1.488,00€
TE35	Mail Server	2,d	16 hrs	OpenPKG-Eng-2	33	1.360,00€
TE36	Test Mail Server	2,d	16 hrs	James	35	992,00€
TE37	Phase 2 finished		0 hrs	Project Manager	36;28	0,00€
TE38	Phase 3 - Branch Offices - OpenPKG	21,d				
TE39	Branch Office 1	3,d				
TE40	DNS-Server	,5d	4 hrs	OpenPKG-Eng-1	37	340,00€
TE41	Test DNS-Server	1,d	8 hrs	Jim	40	496,00€
TE42	NFS-Server	,5d	4 hrs	OpenPKG-Eng-1	40	340,00€
TE43	Test NFS-Server	1,d	8 hrs	John	42	496,00€
TE44	Proxy-Server	,5d	4 hrs	OpenPKG-Eng-2	37	340,00€
TE45	Test Proxy-Server	1,d	8 hrs	James	44	496,00€
TE46	Samba-Server	,5d	4 hrs	OpenPKG-Eng-2	44	340,00€
TE47	Test Samba-Server	2,d	16 hrs	George	46	992,00€
TE48	Branch Office 1 finished		0 hrs	Project Manager	47;43;41;45	0,00€
TE49	Branch Office 2	3,d				
TE50	DNS-Server	,5d	4 hrs	OpenPKG-Eng-1	48	340,00€
TE51	Test DNS-Server	1,d	8 hrs	Robert	50	496,00€
TE52	NFS-Server	,5d	4 hrs	OpenPKG-Eng-1	50	340,00€
TE53	Test NFS-Server	1,d	8 hrs	Jim	52	496,00€
TE54	Proxy-Server	,5d	4 hrs	OpenPKG-Eng-2	48	340,00€
TE55	Test Proxy-Server	1,d	8 hrs	Jan	54	496,00€
TE56	Samba-Server	,5d	44 hrs	OpenPKG-Eng-2	54	3.740,00€
TE57	Test Samba-Server	2,d	16 hrs	James	56	992,00€
TE58	Branch Office 2 finished		0 hrs	Project Manager	57;53;51;55	0,00€
TE59	Branch Office 3	3,d				
TE60	DNS-Server	,5d	4 hrs	OpenPKG-Eng-1	58	340,00€
TE61	Test DNS-Server	1,d	8 hrs	John	60	496,00€
TE62	NFS-Server	,5d	4 hrs	OpenPKG-Eng-1	60	340,00€
TE63	Test NFS-Server	1,d	8 hrs	Robert	62	496,00€

TE64	Proxy-Server	,5d	4 hrs	OpenPKG-Eng-2	58	340,00€
TE65	Test Proxy-Server	1,d	8 hrs	George	64	496,00€
TE66	Samba-Server	,5d	4 hrs	OpenPKG-Eng-2	64	340,00€
TE67	Test Samba-Server	2,d	16 hrs	Jan	66	992,00€
TE68	Branch Office 3 finished		0 hrs	Project Manager	63;67;61;65	0,00€
TE69	Branch Office 4	3,d				
TE70	DNS-Server	,5d	4 hrs	OpenPKG-Eng-1	68	340,00€
TE71	Test DNS-Server	1,d	8 hrs	Jim	70	496,00€
TE72	NFS-Server	,5d	4 hrs	OpenPKG-Eng-1	70	340,00€
TE73	Test NFS-Server	1,d	8 hrs	John	72	496,00€
TE74	Proxy-Server	,5d	4 hrs	OpenPKG-Eng-2	68	340,00€
TE75	Test Proxy-Server	1,d	8 hrs	James	74	496,00€
TE76	Samba-Server	,5d	4 hrs	OpenPKG-Eng-2	74	340,00€
TE77	Test Samba-Server	2,d	16 hrs	George	76	992,00€
TE78	Branch Office 4 finished		0 hrs	Project Manager	73;77;71;75	0,00€
TE79	Branch Office 5	3,d				
TE80	DNS-Server	,5d	4 hrs	OpenPKG-Eng-1	78	340,00€
TE81	Test DNS-Server	1,d	8 hrs	Robert	80	496,00€
TE82	NFS-Server	,5d	4 hrs	OpenPKG-Eng-1	80	340,00€
TE83	Test NFS-Server	1,d	8 hrs	Jim	82	496,00€
TE84	Proxy-Server	,5d	4 hrs	OpenPKG-Eng-2	78	340,00€
TE85	Test Proxy-Server	1,d	8 hrs	George	84	496,00€
TE86	Samba-Server	,5d	4 hrs	OpenPKG-Eng-2	84	340,00€
TE87	Test Samba-Server	2,d	16 hrs	James	86	992,00€
TE88	Branch Office 5 finished		0 hrs	Project Manager	83;87;81;85	0,00€
TE89	Branch Office 6	3,d				
TE90	DNS-Server	,5d	4 hrs	OpenPKG-Eng-1	88	340,00€
TE91	Test DNS-Server	1,d	8 hrs	John	90	496,00€
TE92	NFS-Server	,5d	4 hrs	OpenPKG-Eng-1	90	340,00€
TE93	Test NFS-Server	1,d	8 hrs	Robert	92	496,00€
TE94	Proxy-Server	,5d	4 hrs	OpenPKG-Eng-2	88	340,00€
TE95	Test Proxy-Server	1,d	8 hrs	George	94	496,00€
TE96	Samba-Server	,5d	4 hrs	OpenPKG-Eng-2	94	340,00€
TE97	Test Samba-Server	2,d	16 hrs	Jan	96	992,00€
TE98	Branch Office 6 finished		0 hrs	Project Manager	93;97;91;95	0,00€
TE99	Branch Office 7	3,d				
TE100	DNS-Server	,5d	4 hrs	OpenPKG-Eng-1	98	340,00€
TE101	Test DNS-Server	1,d	8 hrs	Jim	100	496,00€
TE102	NFS-Server	,5d	4 hrs	OpenPKG-Eng-1	100	340,00€
TE103	Test NFS-Server	1,d	8 hrs	John	102	496,00€
TE104	Proxy-Server	,5d	4 hrs	OpenPKG-Eng-2	98	340,00€
TE105	Test Proxy-Server	1,d	8 hrs	James	104	496,00€
TE106	Samba-Server	,5d	4 hrs	OpenPKG-Eng-2	104	340,00€
TE107	Test Samba-Server	2,d	16 hrs	George	106	992,00€
TE108	Branch Office 7 finished		0 hrs	Project Manager	103;107;101;1	0,00€
TE109	Phase 3 finished		0 hrs	Project Manager	108	0,00€
TE110	Phase 4 - DMZ Availability	2,5d				
TE111	Mail relay Server Availability	,5d	4 hrs	OpenPKG-Eng-1	109	340,00€
TE112	Test - Mail relay Server Availability	2,d	16 hrs	Robert	111	992,00€
TE113	Web Server Availability	,5d	4 hrs	OpenPKG-Eng-2	109	340,00€
TE114	Test - Web Server Availability	2,d	16 hrs	Jan	113	992,00€
TE115	Phase 4 finished		0 hrs	Project Manager	114;112	0,00€
TE116	Phase 5 - HQ Availability	5,d				
TE117	Intranet Server Availability	,5d	4 hrs	OpenPKG-Eng-1	115	340,00€
TE118	Test - Intranet Server Availability	2,d	16 hrs	Jim	117	992,00€
TE119	Samba Availability Branch Offices	,5d	4 hrs	OpenPKG-Eng-1	117	340,00€
TE120	Test - Samba Availability	1,d	8 hrs	John	119	496,00€
TE121	DNS Server Availability Branch Offices	,5d	4 hrs	OpenPKG-Eng-1	119	340,00€
TE122	Test - DNS Server Availability Branch Offices	1,d	8 hrs	Robert	121	496,00€
TE123	Mail Server Availability	1,d	8 hrs	OpenPKG-Eng-1	121	680,00€
TE124	Test - Mail Server Availability	2,d	16 hrs	Jim	123;118	992,00€
TE125	Proxy Server Availability	,5d	4 hrs	OpenPKG-Eng-2	115	340,00€
TE126	Test - Proxy Server Availability	1,d	8 hrs	James	125	496,00€
TE127	NFS Server Availability	,5d	4 hrs	OpenPKG-Eng-2	125	340,00€
TE128	Test - NFS Sever Availability	1,d	8 hrs	George	127	496,00€
TE129	Database Server Availability	1,d	8 hrs	OpenPKG-Eng-2	127	680,00€

TE130	Test - Database Server Availability	2,d	16	hrs	Jan	129	992,00€
TE131	Monitoring Server Availability	1,d	8	hrs	OpenPKG-Eng-2	129	680,00€
TE132	Test - Monitoring Server Availability	2,d	16	hrs	James	131;126	992,00€
TE133	Phase 5 finished		0	hrs	Project Manager	132;120;122;1 24;128;130	0,00€
TE134	Phase 6 - Branch Office Availability	11,d					
TE135	Samba Availability Branch Offices	3,5d	28	hrs	OpenPKG-Eng-1	133	2.380,00€
TE136	Test - Samba Availability	4,d	32	hrs	John	135	1.984,00€
TE137	Proxy Server Availability Branch Offices	3,5d	28	hrs	OpenPKG-Eng-1	135	2.380,00€
	Test - Proxy Server Availability Branch Of-						
TE138	fices	4,d	32	hrs	Robert	137	1.984,00€
TE139	NFS Server Availability	1,d	8	hrs	OpenPKG-Eng-2	133	680,00€
TE140	Test - NFS Sever Availability	2,d	16	hrs	George	139	992,00€
TE141	DNS Server Availability Branch Offices	3,5d	28	hrs	OpenPKG-Eng-2	139	2.380,00€
TE142	Test - DNS Server Availability Branch Offices	4,d	32	hrs	Jan	141	1.984,00€
						142;136;138;1	
TE143	Phase 6 finished		0	hrs	Project Manager	40	0,00€
TE144	Phase 7 - Documentation	5,d					
					Jim, John, Jan, George, James, Robert, OpenPKG- Engineer-1, O- penPKG-Engineer-2		
TE145	Documentation	5,d	40	hrs		143	21.680,00€
TE146	Phase 7 - finished					145	
TE148	Project Management	87,d	696	hrs	Project Manager	1	48.720,00€
TE147	Project END		0	hrs	Project Manager	146;147	0,00€
Total Project Costs							261.212,00€

Table 22: Task List external partner - Example Company

10 Glossary

Application is a software package that offers services and performs functions specified by users.

Balanced Score Cards (BSC) is a strategic management system that enables the management to measure their strategy.⁴²

Basel II is a tool that enables banks to equally measure and rate balance sheets in order to issue a loan to their customer.

Binary see executable.

Binary Package is a package that contains executable software that was already compiled on a system.

Bug is a program failure, which causes an unexpected result.

C is a programming language in which many programs are written.

Current version is the latest version of software or a compendium of software.

Complain rate is the rate of complaints per product sold.

Compile is the process concerned with translating the software from the source code to an executable program, which runs on a certain operating system.

Configuration files are files with which software gets configured and features of software get enabled or disabled.

Datacenter is the place where many backend computer systems are located, most of the time it requires special access rights, is located in the underground, is air-conditioned and automatic fire distinguishers or fire prevention systems are installed.

Dependencies are the reliance on other software that could be binaries or libraries.

Deploy in a software environment means install, upgrade and de-install.

Development cycle is the time that is required to further develop and improve the software to fulfil the next milestone.

Distribution is an Open Source software bundle that puts together several kinds of software to one compendium.

Distributor is the entity that sells software or a software compendium.

Executable is a file, which can be executed on a system to perform a special task like starting a service.

Failure rate is the rate of failures of a sold product; also known as Six Sigma.⁴³

Features are enhancements to software that might be required by some people but not by others.

Financial Leverage is the ratio between debt and equity capital.

Fluctuation Rate is the ratio between people who terminate their contract and people employed.

FOSS is Free and Open Source Software.

Framework is a specified environment, which is the same on all systems.

Free Software Foundation (FSF) is a non-profit organization that supports the free software movement “free in freedom”.

FTP file transfer protocol is used to transfer files from a server to a client.

Full-time equivalent represents more or less a full-time employee because it measures the productivity of a worker.

GNU is not UNIX. The GNU project was launched to develop a UNIX like operating system. Mostly a Linux kernel is used by these systems.

Home-grown software is software developed inside the company and only used inside the company.

Heterogeneous environment is when a variety of different systems are used within a company.

ix86 is a processor type very widely used.

Jumpstart is the mechanism from SUN to automatically install systems.

Kickstart is the mechanism for LINUX to automatically install systems.

Mission describes what our business will be, what our business should be and the value to the customer.⁴⁴

Mount point is the point that describes where files get stored on UNIX systems. This could be either on local disks or on a remote storage.

Net groups are special groups that get special access to systems. They are used together with the yellow pages system under UNIX.

NFS network file system for UNIX systems invented, which was invented by SUN Microsystems to share files across the network.

Operating system is the piece of software that controls the hardware installed on a computer. Depending on the functionality and installed features it may provide, for example, a graphical user interface.

OSPF Open Shortest Path First is a routing protocol to distribute routing information from router to router.

Package collection of software that can consist of all different kinds of files, required by the program to work. It includes a mechanism to show dependencies on other software or packages.

Patch is a piece of source code that needs to be applied to another source code to fix a problem. After this the source code needs to be compiled again.

PDP-7 a minicomputer produced by digital equipment, introduced in 1965.

Platform is the bundle of hardware and the operating system.

Porting software is when software is made available for systems where it is not available yet.

Price/performance ratio is the ratio between the delivered performance and the price for the product or service.

RedHat is a distributor of a special kind of Linux.

Release cycle is the period of time from point of one release until the next release becomes available.

Release version is the version at the time a milestone is reached and the software is tested on the required operating systems.

RIP Routing Information Protocol is used to distribute routing information from router to router.

ROI describes the interest on the total investment made by the company.

Root has two meanings in UNIX. One is the default system administrator account of a UNIX system, and the second one is the top directory / of a UNIX system.

ROS is the ratio between net income before taxes and sales revenue.

Sarbanes-Oxley Act (SOX) is a regulation to which US companies have to comply when they are traded on the US stock exchange.

Server offers a service to clients.

Service from a system point of view it is something offered from a server and can be used by a client.

Shared Value are the company values shared with all related parties

Six sigma is a methodology to manage process variations that cause defects. Six sigma is a maximum of one failure for each 1 million produced parts.⁴⁵

Snapshot is taken at a specific point in time and is a copy of a stable current version used for the release version.

Source code is a human readable code in which a program for a computer is.

Source package is a package that only contains files that are not compiled yet.

SPARC is hardware architecture like ix86, but it's instructions (commands for a processor) are totally different.

Spyware is a piece of software that is installed on a computer to get information out of it in an illegal way. It might be installed by a virus or by mistake by the user themselves.

Strategy Maps is a tool that helps to visualize the strategy of a company and to explain it to the employees.⁴⁶

sudo is a program that allows a normal user of a system to do certain tasks where under normal circumstances a system administrator account is required.

SUSE is a distributor of a special kind of Linux

Update is an reinstallation of a software or program which includes bug fixes and minor enhancements

Upgrade is a reinstallation of a software or program, which includes major enhancements and changes.

Value benefit analysis several products get ranked on each value they provide and what kind of benefit it has to the company.

Vision is a picture of the future, near enough so that we can see how to realize it, but far away enough to motivate the organization.

XML (extensible Markup Language) is a language for Web and non-Web applications and describes the structure of a document.

11 Figures

<i>Figure 1: Filiations of UNIX and UNIX-like systems</i>	3
<i>Figure 2: Open Source Software installation without OpenPKG</i>	8
<i>Figure 3: Open Source installation with OpenPKG</i>	8
<i>Figure 4: Environment before consolidation</i>	26
<i>Figure 5: Environment after consolidation</i>	26
<i>Figure 6: Interconnection - Example Company</i>	48
<i>Figure 7: Headquarters - Example Company</i>	50
<i>Figure 8: Branch office - Example Company</i>	51
<i>Figure 9: Financial Perspective - Example Company</i>	52
<i>Figure 10: Customer Perspective - Example Company</i>	53
<i>Figure 11: Internal Perspective - Example Company</i>	53
<i>Figure 12: Learning and Growth Perspective - Example Company</i>	54
<i>Figure 13: Strategy Map - Example Company</i>	54
<i>Figure 14: Project Phases - Example Company</i>	57
<i>Figure 15: Costs and savings overview OpenPKG implementation - Example Company</i>	60

12 Tables

<i>Table 1: Supported Operating Systems</i>	4
<i>Table 2: Commercial Software - Advantages</i>	9
<i>Table 3: Commercial Software - Disadvantages</i>	10
<i>Table 4: Free and Open Source Software - Advantages</i>	10
<i>Table 5: Free and Open Source Software - Disadvantages</i>	11
<i>Table 6: Home-Grown Software – Advantages</i>	11
<i>Table 7: Home-Grown Software - Disadvantages</i>	11
<i>Table 8: Business continuity - Issues, Risks, and Opportunities</i>	20
<i>Table 9: Business continuity solution</i>	30
<i>Table 10: Services provided - Portland State University</i>	34
<i>Table 11: Services provided - International Telecommunications Company</i>	39
<i>Table 12: Software used for services at the headquarters – Example Company</i>	49
<i>Table 13: Software used for services at branch offices – Example Company</i>	50
<i>Table 14: Key IT Governance Decisions</i>	55
<i>Table 15: Engineer’s cost table - Example Company</i>	58
<i>Table 16: UNIX-Server Maintenance Costs per month before OpenPKG - Example Company</i>	58
<i>Table 17: UNIX-Server Maintenance Costs per month after OpenPKG - Example Company</i>	58
<i>Table 18: Monthly project costs in-house - Example Company</i>	59
<i>Table 19: Monthly project costs with external partner - Example Company</i>	59
<i>Table 20: Costs for External Partner - Example Company</i>	59
<i>Table 21: Task List in-house - Example Company</i>	69
<i>Table 22: Task List external partner - Example Company</i>	72

13 References

- Abrahams Paul W. and Larson Bruce R.: "UNIX for the Impatient", 2nd Edition, Addison Wesley: 1995
- Dorf, Richard C. and Bayer Thomas: "Technology Venntures", 1st edition, Mc Graw Hill: New York, 2005
- Drucker Peter: "Management: Tasks, Responsibilities, Practrices", 1st edition, Harper Business: New York, 1993
- Engelschall Ralf S.: "OpenPKG Slide Set" Internet: <http://www.openpkg.org/doc.html> (visited February 22nd, 2006).
- Herold Helmut: "UNIX Grundlagen", 3rd edition, Addison-Wesley (Deutschland) GmbH: Bonn, 1994.
- Holstein William K.: "WKHEnt2-Imper.ppt" GSBA MBA program: Horgen, 2006.
- Horvarth Peter: "Balanced Scorecards" 1st edition, Schäffer-Poeschel Verlag: Stuttgart, 1997
- Kaplan Robert S. and Norton David P.: "Strategy Maps", 1st edition, Havard Business School Publishing: Boston, 2004
- Laudon Kenneth C. and Laudon Jane P: "Management Information Systems", 9th edition, Prentice Hall: New Jersey, 2004
- Nemeth Evi, Snyder Garth, Seebass Scott and Hein Trent R.: "Systemadministration unter UNIX", 2nd Edition, Prentice Hall: New Jersey, 1997
- Niemann Klaus D.: "Von der Unternehmensarchitektur zur IT-Governance", 1st Edition, Vieweg: Wiesbaden 2005
- Openpkg.com: "OpenPKG.com", Internet: <http://www.openpkg.com/>, (visited May 20th, 2006)
- Openpkg.com - Service: "OpenPKG.com Service", Internet: <http://www.openpkg.com/services/> (visited May 20th 2006)
- Openpkg.net: "OpenPKG.net", Internet: <http://www.openpkg.net/>, (visited May 20th, 2006)
- Openpkg.org: "OpenPKG.org", Internet: <http://www.openpkg.org/>, (visited May 20th, 2006)
- OSI: "Open Source Initiative", Internet: <http://www.opensource.org/>, (visited February 21st, 2006)

OSI definition: “Open Source Initiative”, Internet:

<http://www.opensource.org/docs/definition.php>, (visited February 21st, 2006)

Schultz Eugene E.: “A framework for understanding and predicting insider attacks”, Internet:

<http://www.itsec.gov.cn/webportal/download/2002-A%20framework%20for%20understanding%20and%20predicting%20insider%20attacks.pdf> (visited March 3rd, 2006)

Thomen Jean-Paul: “Managementorientierte Betriebswirtschaftslehre”, 7th Edition, Versus: Zürich, 2004

Weill Peter and Ross Jeanne W.: “IT Governance” 1st edition, Havard Business School Publishing: Boston, 2004

Wikipedia Free software: “Free software”, Internet:

http://en.wikipedia.org/wiki/Free_software (visited March 8, 2006)

Wikipedia Freeware: “Freeware”, Internet: <http://en.wikipedia.org/wiki/Freeware> (visited February 22nd, 2006)

Wikipedia Open source Software: “Open Source Software”, Internet:

http://en.wikipedia.org/wiki/Open-source_software (visited February 21st, 2006).

Wikipedia Information technology controls: “Information technology controls” Internet:

http://en.wikipedia.org/wiki/Information_technology_controls (visited June 20, 2006)

Wikipedia UNIX: “UNIX”, Internet: <http://en.wikipedia.org/wiki/UNIX> (visited February 21st, 2006).

14 Endnotes

¹ Nemeth Evi, Snyder Garth, Seebass Scott and Hein Trent R., Chapter 1

² Abrahams Paul W. and Larson Bruce R., Chapter 1

³ Wikipedia, UNIX

⁴ Helmut Herold, Chapter 1.4

⁵ Wikipedia, UNIX

⁶ Laudon Kenneth C. and Laudon Jane P, Chapter 6.4

⁷ OSI

⁸ Wikipedia, Open source Software

⁹ OSI definition

¹⁰ Wikipedia, Free Software

¹¹ Wikipedia, Free Software

¹² Wikipedia, Free Software

¹³ Wikipedia, Free Software

¹⁴ Wikipedia, Freeware

¹⁵ Openpkg.org

¹⁶ Openpkg.net

¹⁷ Openpkg.com

¹⁸ Openpkg.org

¹⁹ Engelschll Ralf S., Page 7

²⁰ Engelschll Ralf S., Page 8

²¹ Weill Peter and Ross Jeanne W., Chapter 2

²² Weill Peter and Ross Jeanne W., Chapter 2

²³ Thomen Jean-Paul, Chapter 2

²⁴ William K. Holstein, Page 9

²⁵ Niemann Klaus D., Chapter 3

²⁶ Schultz Eugene E.

²⁷ Wikipedia Information technology controls

²⁸ Openpkg.com - Service

²⁹ Openpkg.com - Service

³⁰ Openpkg.com - Service

³¹ Openpkg.com - Service

³² Openpkg.com - Service

- ³³ Openpkg.com - Service
- ³⁴ Openpkg.com - Service
- ³⁵ Openpkg.com - Service
- ³⁶ Kaplan Robert S. and Norton David P., Chapter 2
- ³⁷ Dorf, Richard C. and Bayer Thomas, Chapter 3.2
- ³⁸ PAIB, Chapter 2.4
- ³⁹ Horvarth Peter, Part 1
- ⁴⁰ Niemann Klaus D., Chapter 2.6
- ⁴¹ Weill Peter and Ross Jeanne W., Chapter 2
- ⁴² Horvarth Peter
- ⁴³ Laudon Kenneth C. and Laudon Jane P, Chapter 14
- ⁴⁴ Drucker Peter, Chapter 7
- ⁴⁵ Laudon Kenneth C. and Laudon Jane P, Chapter 14
- ⁴⁶ Kaplan Robert S. and Norton David P.