Bachelorthesis

Björn Kiencke

Conception and Implementation of a Software Distribution to Facilitate the Usage of an Energy Data Monitoring

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Björn Kiencke

Conception and Implementation of a Software Distribution to Facilitate the Usage of an Energy Data Monitoring System

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Thomas Klinker
Zweitgutachter : Prof. Dr. Franz Schubert

Abgegeben am 20. Februar 2013

Björn Kiencke

Thema der Bachelorthesis

Entwurf und Implementierung einer Softwarezusammenstellung mit Energiedatenmonitoring-Funktionalität

Stichworte

Direktstartbetriebssystem, Energiedatenmonitoring, Remastering, Java, JEVis, Ubuntu, Kernel

Kurzzusammenfassung

Diese Arbeit umfasst den Entstehungsprozess einer Lösung, die Energiedatenmonitoring für Test- und Schulungszwecke zur Verfügung stellt. Programme zum Monitoring von Energiedaten werden eingesetzt, um die Energieeffizienz von Unternehmen zu erhöhen und somit den Anforderungen von Energiemanagementsystemen, wie sie beispielsweise die ISO 50001 vorschreibt, gerecht zu werden. Das genutzte Monitoring Programm setzt mehrere Komponenten, u. a. eine Datenbanksoftware, voraus und benötigt daher erheblichen Konfigurationsaufwand. In der vorliegenden Arbeit wird der Ansatz eines Direktstartbetriebssystems mit Energiedatenmonitoring-Funktionalität umgesetzt. Zur Vereinfachung des Remastering-Prozesses wird ein Programm entworfen.

Björn Kiencke

Title of the paper

Conception and Implementation of a Software Distribution to Facilitate the Usage of an Energy Data Monitoring System

Keywords

Live System, Energy Data Monitoring, Remastering, Java, JEVis, Ubuntu, Kernel

Abstract

This report comprises the development process of a solution which offers energy data monitoring for testing and training. This kind of program is used to increase the energy efficiency of industrial companies and hence to implement the requirements of energy management systems according to standards like ISO 50001.The JEVis Data Monitoring System depends on several components, e.g. a database server. Therefore a huge overhead on configuration is demanded. In the thesis presented the approach of a live system to facilitate using an energy data monitoring system is taken. Furthermore, a program to simplify the remastering-process is designed.

Contents

1 Introduction

Since the age of industrialization world's economy is reliant on resources of any kind, enabling humanity to run all kinds of power taking devices, may they deliver mobility, comfort, productivity or medical capability. However, during the recent decades especially non-renewable resources are running short and although renewable energy sources are on the rise, new technology can not close the gap as quickly. Moreover, political decisions contribute to this problem significantly which has lead to a dramatic increase of energy prices since the turn of the millennium. The price of oil can be taken as a good example for this and it is commonly accepted that the peak was already reached during the last years (cf. Kausch et al. 2011, p. 11). For industry, mother earth's signs of exhaustion and the above mentioned lack of alternatives means facing the costly effects of this development. Therefore, energy efficiency has become the key word for companies CEO's to get control over this problem. In addition, costumers get increasingly conscious about ecological sustainability during the method of production of a product. A promising way to realize a resource-optimized method of production of such a kind is the implementation of an energy management system basically enabling the company to monitor the energy the process needs in every production step over time.

To implement a suitable energy management system companies are commonly advised by service providers. One of these is the Envidatec GmbH where this thesis was performed. For the purpose of analysis, control and surveillance energy data monitoring programs are used. In line with this thesis a modular software solution is dealt with. This implies a pedestrian and complex installation which is not appropriate for an - concerning this project - unskilled person. Within this thesis multiple concepts where established to create an easy way of testing allowing users to let the program run out of the box and by that enabling a broad audience to work with this software.

Therefore, the aim of this thesis is the establishment of an energy data monitoring live system. For this a Linux distribution was remastered. This means, that the energy data monitoring application was integrated in the OS and further minor and major modifications were performed.

As an example of a minor modification the adaption in design and the addition of icons for easy handling of the system can be mentioned. Furthermore, major modifications enable the user to install the OS together with the ready to run monitoring software on a local drive. With respect to the hardware available this leads to a faster system and can be used for test and research purposes. Last but not least a continuously advancing development of this software shall be facilitated since this is an open source project. For this the remastering process is implemented in a self-programmed application to provide developers an easy and time-saving way of creating a new version.

To relieve reading this thesis here is an overview of the contained sections. The second chapter gives detailed information about the motivation of implementing energy management systems. Furthermore it deals with background facts on the company, their monitoring software, issues of founding an open source community and subjects of live systems. The third and fourth chapter document the process of conception. This includes the procedure given by modern software engineering: generating use-cases, extracting requirements and specifying the design. Subsequently the next chapter is about the implementation containing explanations of the used programming techniques and examples of implementation in source code. Furthermore the results of the development are presented. The last chapter reviews the implemented solution and sketches out its advantages and disadvantages. Finally, the outlook section gives information about further steps, which are already planned.

2 Background

2.1 Motivation

During the last years energy economic issues have become increasingly important for companies. Limited resources available and the growing demand for energy are leading to higher prices. In addition, political decisions have impact on the price of energy. Especially in Germany, companies are encouraged to optimize their energy consumption. The feed-in tariffs for renewable energies whose additional costs are known as EEG surcharge raise by 2013 up to almost 5.3 ct/kWh. The contribution of the industry to achieve the planned aims implies exploiting energy saving potentials. The Policy Report by the Fraunhofer Institute by order of the Federal Ministry for the Environment Nature Conservation and Nuclear Safety sees a reduction potential of 52 percent on final energy demand compared to the baseline. The major amount of 75 percent savings can be exploit by making use of cross-cutting technologies[1] (cf. Boßmann, Eichhammer, and Elsland 2012, p. 18).

An indicator for the success of the efforts is the energy productivity. It is defined by the consumption of primary energy on products and services in comparison to the gross domestic product.

The reference for the derived statistic is set to 100 percent on the data of 1990. The German government is aiming to double the energy productivity until 2020 compared to 1990. As shown in figure 2.1 the value already improved by almost 50 percent. Considering the aims increased efforts are necessary. In particular the industry has to take activity for optimization. A common way to improve the energy efficiency is given by the specification ISO 50001. The process of implementing an energy management system (EnMS) according to this standard provides the tasks of monitoring, measurement, and analysis. There are certain good reasons using energy data monitoring software to implement this. The most important are the quantity of data samples

[1]Efficient steam and hot water generation as well as optimization of entire systems relying on electric drives.

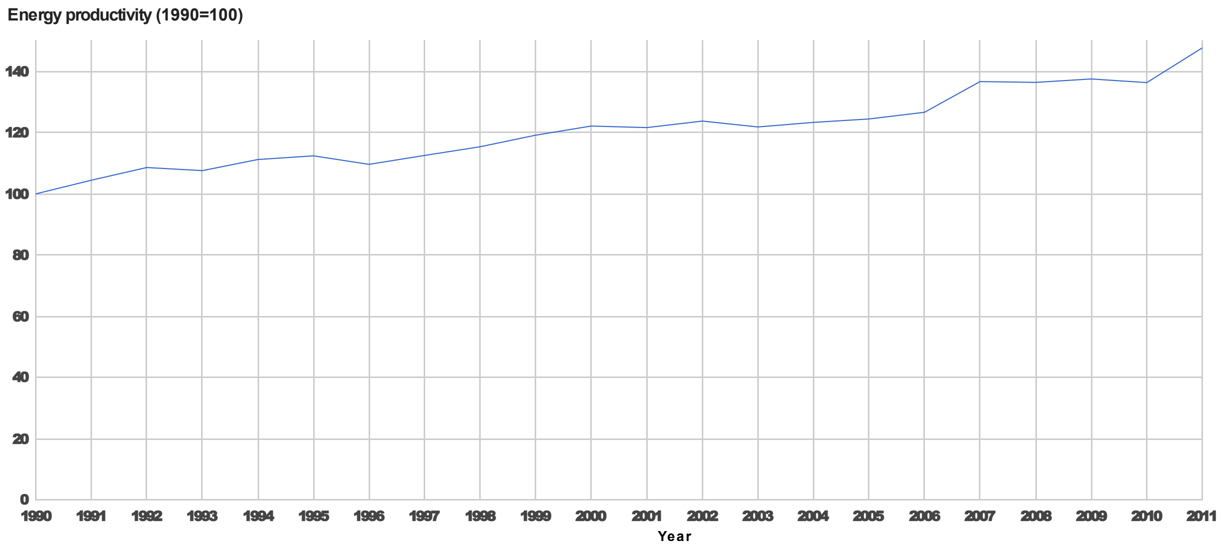


Figure 2.1: Germany's energy productivity (Data: DESTATIS)

delivered by energy meters and clustering of distributed data devices. That gives the opportunity to store data of various measurement points centralized to visualize, combine, and compare them. In this way data monitoring systems contribute to save energy. Therefore, a wide distribution of them is desirable. In particular the availability for university education, partners, and interested persons establishes the idea of monitoring energy data. Furthermore these interested parties participate with ideas, feedback, and work on development to improve the solution. The work done as part of the thesis aims to facilitate the distribution to encourage these processes.

The German company Envidatec GmbH, where the author was working on this thesis, offers services in the field of energy management. Company's philosophy encloses the ideas of innovating energy data monitoring in the sketched way. The following section gives basic information about the Envidatec GmbH and its business area.

2.2 Envidatec GmbH

Envidatec GmbH was founded in 2001. The company with headquarter in Hamburg offers energy services for industrial customers. Since 2008, new tasks arising from new laws and standards concerning energy management systems, e.g. the international standard ISO 50001, have been evolved for the company.

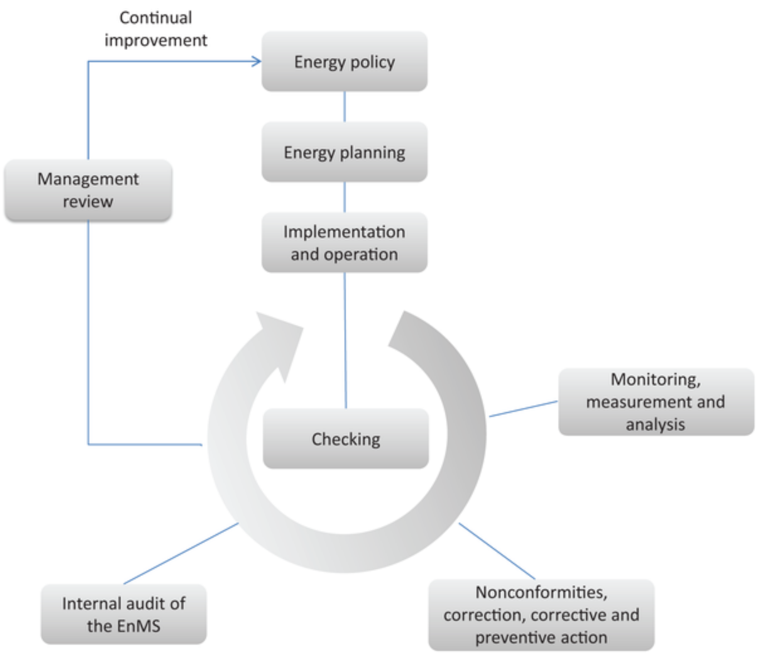

Continual improvement
Energy policy
Energy planning
Management review
Implementation and operation
Checking
Monitoring, measurement and analysis
Internal audit of the EnMS
Nonconformities, correction, corrective and preventive action

Figure 2.2: Energy management system model (Source: International Organization for Standardization 2011, p. 8)

The model shown in figure 2.2 describes the approach of the energy management system model according to ISO 50001. It is implemented as a plan do check act circle that is often used in business economics.

Plan: conduct the energy review and establish the baseline, energy performance indicators, objectives, targets and action plans necessary to deliver results in accordance with opportunities to improve energy performance and the organization's energy policy.

Do: implement the energy management action plans.

Check: monitor and measure processes and the key characteristics of its operations that determine energy performance against the energy policy and objectives and report the results.

Act: take actions to continually improve energy performance and the EnMS.

(see International Organization for Standardization 2011, p. 7)

Appropriate to services like energy efficiency analyzes or creating guidelines for energy management Envidatec provides its own energy data monitoring software:

> Additionally, the Envidatec GmbH provides a software for automated logging, analyzing and visualization of energy and operating data, the JEVis system. This monitoring solution has been distributed meanwhile as open source license. It will not only be used by commercial companies but also from many universities worldwide. For this reason the Envidatec GmbH has built up an international university network for research and development topics based on the JEVis system.
>
> (see Envidatec GmbH 2013, p. 2)

As a part of the already mentioned research and development program the author handled a project at the Envidatec GmbH, which is the topic of this thesis. The structure of Envidatec's data monitoring software is briefly described in the following section to phase in technical background of the work.

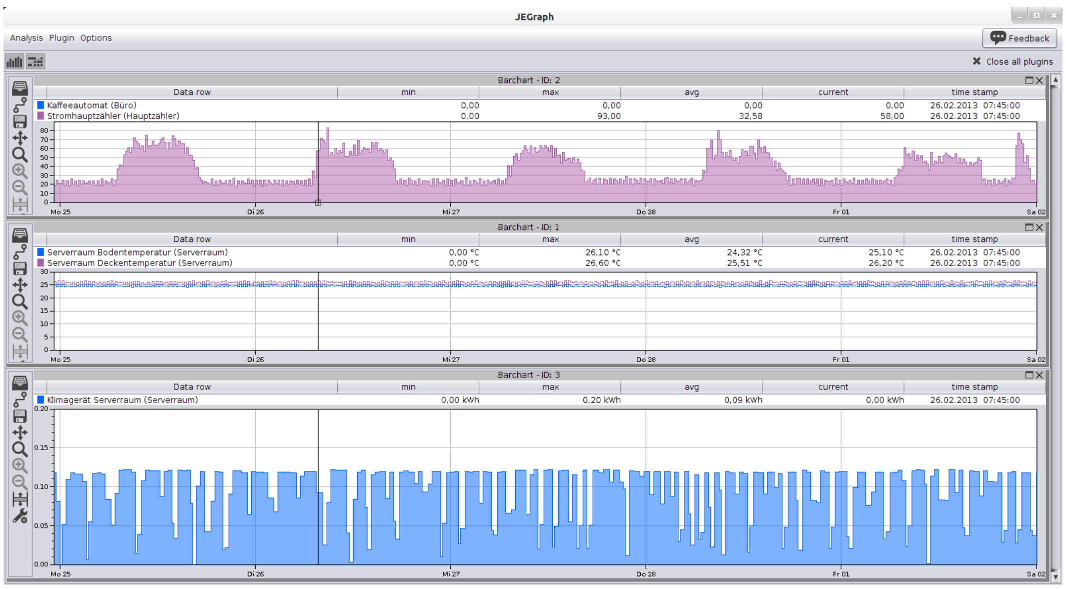
2.3 Data Monitoring Software

The role of data monitoring software in the process of implementing an energy management (according to ISO 50001) is enabling identification of weaknesses, measurement of improvements, and a possibility to protocol efforts.

2.3.1 JEVis

The JEVis Energy Data Monitoring solution appears to the end user in form of a portal page that contains links to two java programs. To launch these programs Java Web Start technique is used. To launch a program according to this technique a Java Network Launching Protocol (JNLP) file is deposited on a server. It contains information about the path to the program, the name of the main class and additional parameters for the program (cf. Marinilli 2002, p. 272). Since the path can be a network resource the user has the benefit of getting an always up to date software.

After logging in to the JEGraph program the user can load previously saved analyses. The user is able to add data rows to plugins, set threshold limits, zoom into graphs and link plugins.



Barchart - ID: 2

Data row	min	max	avg	current	time stamp
🟦 Kaffeeautomat (Büro)	0,00	0,00	0,00	0,00	26.02.2013 07:45:00
🟪 Stromhauptzähler (Hauptzähler)	0,00	93,00	32,58	58,00	26.02.2013 07:45:00

Barchart - ID: 1

Data row	min	max	avg	current	time stamp
🟦 Serverraum Bodentemperatur (Serverraum)	0,00 °C	26,10 °C	24,32 °C	25,10 °C	26.02.2013 07:45:00
🟪 Serverraum Deckentemperatur (Serverraum)	0,00 °C	26,60 °C	25,51 °C	26,20 °C	26.02.2013 07:45:00

Barchart - ID: 3

Data row	min	max	avg	current	time stamp
🟦 Klimagerät Serverraum (Serverraum)	0,00 kWh	0,20 kWh	0,09 kWh	0,00 kWh	26.02.2013 07:45:00

Figure 2.3: JEVis data monitoring software

Taking a closer look on the third plugin displaied in figure 2.3 a vibration of the machine can be seen. Knowledge of consumption helps to scale machines correctly to exploit their best efficientcy point. Advanced tasks can be done in the second program JEConfig, which offers alarming, set up of devices, user management and Octave (MATLab syntax) calculations.

Structure of the JEVis System

Like quite a number of other complex software the JEVis data monitoring system contains multiple modules and various participants. The server is the most complex component in the system. Its sub-modules provide different functionality. For example this contains the task of controlling necessary operations like requesting data from measurement points will be done by the server. The second type of component are data-sources in the measurement infrastructure. Typically these are devices like common industrial energy-meters. The task of visualization is implemented in software on client machines. The technical structure of the JEVis Energy Data Monitoring solution is shown in the following figure.

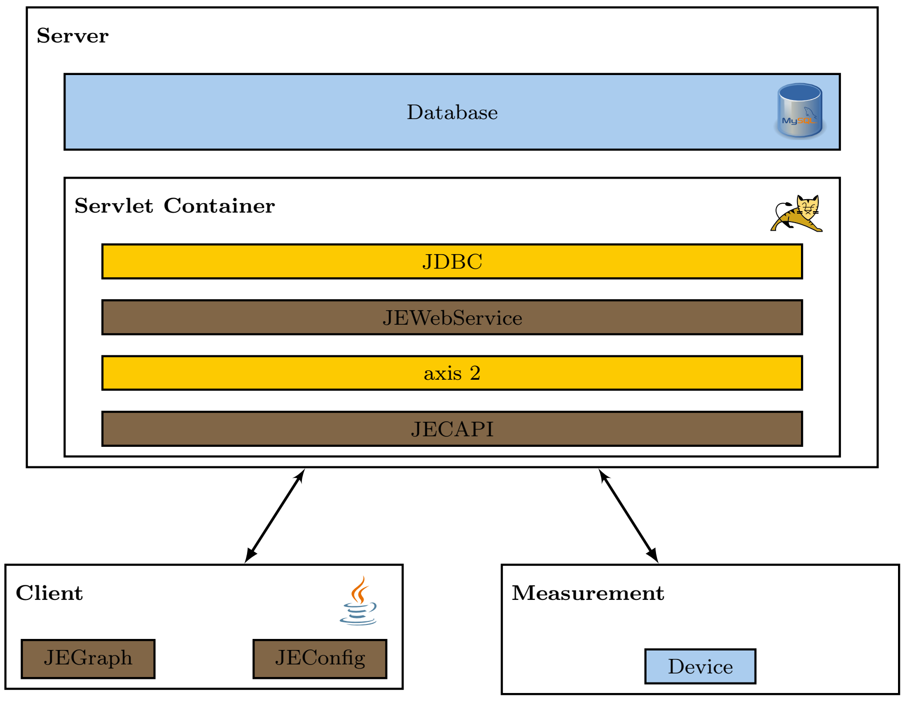


Figure 2.4: JEVis System Setup

The activities of the JEVis System components shown in figure 2.4 can be briefly described as follows:

Server

The server assumes the role of data storage. It is the interface between the data producers that are summarized as measurement infrastructure and client devices that display charts and analyses.

Measurement Infrastructure

JEVis supports connecting different types of engergy meters. Connections are realized via the JEADFWeb tool, that supports various web protocols. It includes functionality for HTTP, FTP, SOAP, and SQL web protocols. The data can have CSV or XML format. At the moment a second but already deprecated option exists, that implements a direct connection to the JEWebService. The principle is explained in detail within the section about including data sources 5.2.1.

The JEADFWeb can read out data from any device, that is reachable via that protocols and which's data is stored in one of that formats. If a device needs a protocol or format that has not been added yet, the tool is easily extendable: When creating a data structure in the JEVis System, every device gets its driver. You can add the driver parts offered by the system, or add you own one.

(see OpenJEVis.org 2013[a], p. 2)

Clients

Every PC with Java environment can be a JEVis client if the server is reachable inside client's network. The client software programs JEGraph and JEConfig offer a login dialog on start-up. Therefore, users can log in to the monitoring system using their accounts on any machine.

Remarkable on the JEVis Energy Data Monitoring Solution is the fact it is an open source software. Thus, the next section gives an insight into topics around open source.

2.4 Open Source

The success of a software essentially depends on users acceptance. Therefore software developers rely on feedback from them. Particularly in the case of software with limited distribution and any budget for usability testing the contact to the customer is crucial. This way of software development is a remarkable feature about open source and specifically about OpenJEVis. With other general characteristics deals the following chapter, while section 2.4.2 outlines the benefits of open source.

2.4.1 General Principles

The open source initiative was founded in 1998 by Eric Raymond and Bruce Perens. It is the committed aim to encourage the idea of open source. Therefore they are distributing an overview about existing open source licenses (see Bandel 2010, p. 8). There is a list of ten points available on the official Open Source website that enumerates the general principles of this kind of software. They are meant to be essential preconditions for successful and effective software engineering:

1. Free redistribution

2. Source code
3. Derived works
4. Integrity of the author's source code
5. No discrimination against persons or groups
6. No discrimination against fields of endeavor
7. Distribution of license
8. License must not be specific in a product
9. License must not restrict other software
10. License must be technology-neutral

(cf. Open Source Initiative 2012)

Without getting too much into detail the most important points shall be explained: One of the main aspects about open source besides its free distribution (no. 1) is that modification and copying of a source has to be permitted (no. 4). This concept is realized with a specified license model (no. 7 to 10). To be more concrete, both use and release must be allowed as an executable file (no. 2). Furthermore, it is to stress that there do not exist any restrictions against certain groups of users and fields of activity what can be seen as a great enrichment of open source (no. 5 and 6). (see Envidatec GmbH 2012).

2.4.2 Benefits for Business

What is the motivation for businesses to support open source projects? Which specific benefits can be advertised? These questions will be answered in this section. In the concept of open source proprietors, a management or employees do not exist. Therefore anybody has the right to profit from the usage, development or change of the software (see Nüttgens and Tesei 2000, p. 10). As explained in the section below, the concept of open source is network based meaning that many participants are developing a product. Also commercial businesses are getting more and more interested in open source – in other words: They pursuit being part of that network. In general, a successful process results in a collective increase of efficiency, which in turn has positive effects on the competitive position of a single company (see ibid., p. 13). Getting more into detail, new business models which belong to the new strategy of involving open source software can result in profit (see Bandel 2010, p. 5). An abstract of business models that are already implemented in companies business strategies are listed below:

- Companies distribute open source software to generate potential customers for additional, compatible software products that are chargeable.
- Companies like RedHat offer chargeable support and service contracts binded to the open source software they distribute.
- Companies like Apple offer chargeable hardware, e.g. a mobile phone, in combination with software applications that base on open source software (see ibid., pp. 53-54).

To summarize the increase of the consumer surplus is the main motivation of enclosing open source software in a business strategy. Businesses are highly interested in the know-how, skills, and creativity of the open source community to have advantages in the competition. Furthermore, they have low hardware costs and benefit from the free testing within the community (see ibid., p. 54). All in all the coexistence of open source software and proprietary software products inspirit the market.

2.5 Open JEVis

Envidatec points out to offer the monitoring software as open source. As its basic strategy for further charged services this is an unique selling point of the company. The realization of the development as open source keeps several advantages. This business model enables external partners to participate on increasing the function range of JEVis. Hardware manufactures and universities are typical partners.

2.5.1 Analyzing the Open Source Community Structure

The approach of an open source project needs some special infrastructure. One basic thought is to make hurdles to entry as low as possible. The mindmap show in figure 2.5 deals with the important aspects when getting in touch with JEVis.

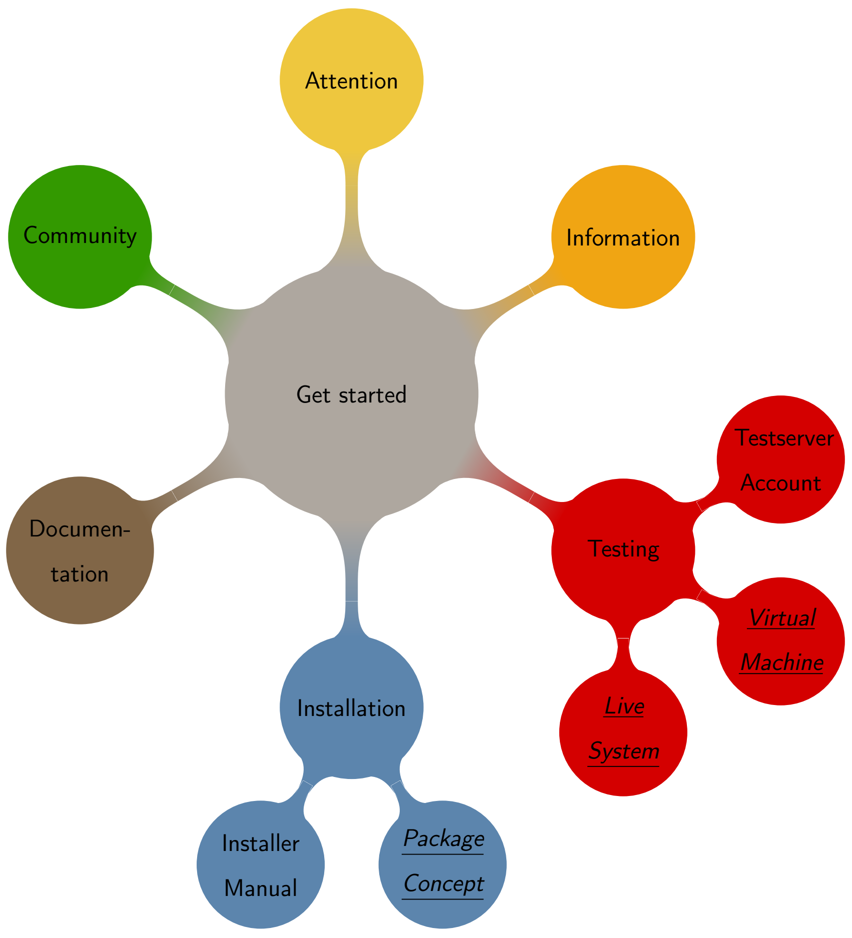


Figure 2.5: Get in touch with JEVis

The mindmap displays the process of getting in touch with Jevis resp. the OpenJEVis Community from the viewpoint of a person which is interested in an energy data monitoring solution. Starting from this the aspects are visualized in figure 2.5 and annotated in the following. Thus, the figure contains the analysis of the existing structure and ideas for improvements which are formatted underlined and italic.

Attention

The first hurdle for a software is to be noticed by potential users. In addition to the traditional distribution channels and online presence there are some special ideas fitting well to the open source philosophy. Workshops about energy efficiency and online events like the OpenJEVis Days are already practiced by the company.

Information

In the following the user's impression depends on quality of the information and communication. The most important platform for sharing information is the community website.

Testing

In the case of the information have convinced the user, he will make concrete efforts to check whether the software fits to his requirements. While working on the thesis the common option to test the software was to create a guest account. New ideas are addressed to advanced users who want to have their own system. Both concepts base on an operating system with an "out the box" running monitoring system. Benefits of using a separate system are undisturbed trying without registration and full control of own data.

Installation

For productive usage of the system a customized installation is necessary. A comfortable technology to install and update software known as "package managers" exists for linux systems. This tool may be interesting for simple JEVis installations.

Community and Documentation

All other nodes deal with structural organization. As usual for open source projects documentation and communication issues are placed on the community website.

2.5.2 Improvement suggestions

The structure of an open source project has to be considered to assess weaknesses correctly. Aspects like transparency, quality of communication, and documentation contribute a lot to the success of a project. Detecting the most critical aspects was done before start working on this thesis and are identified within the offers of testing and installation: The option of a guest account requires the user's action on registration. An incentive for users that do not want to register is a live system which offers easier testing and the possibility to apply own changes without any risk. The work on this live system with energy data monitoring functionality is the central purpose of this thesis. Another suggestion in the scope of installation is the package concept which is used on Linux operating systems to organize installations and updates. Intending to fulfill as many improvement suggestions as possible chapter 3 specifies usage scenarios and the resulting requirements on the software distribution.

2.6 Live Operating Systems

Thus, the solution of a live system seems adequate the following section overviews its history and its actual importance.

2.6.1 Basics

One of the most important milestones in the young history of live systems was Knoppix Linux. It was developed by the German engineer Klaus Knopper, who published the first version of his live system in the year 2000. In the preface of "Knoppix Hack", he said his idea was driven by giveaways. At that time, Linux based rescue discs were given away at computer expos and Knopper started his work on a "personal rescue CD" (cf. Jones 2008, pp. 11-12).

> That way, it would be possible to use software that I needed from a CD, rather than carry around an expensive and fragile laptop. Computers are available everywhere anyway, so why not just have the software in your pocket instead.
>
> (see Knopper and Rankin 2007, foreword)

The sparsely standardized hardware for computers is reasoning the central problem of a live system: the hardware support. Because the approach of minimal hardware requirements is not suitable for most of the software the live system, first of all, has to provide automatic configuration of PC hardware. The second important technology is the compression of containing software that makes it possible to provide a whole working system on a single CD. The question that arises is why open source operating systems could occupy the field of live systems without appreciable competitive. R. Hattenhauer instances economic interest of the big software manufacturers to sell as much as possible full versions (cf. Hattenhauer 2005, p. 16). Since that time, more and more projects of live operating systems with highly diverse functional range have been started. Nowadays, the majority of Linux distributions are shipped with the functionality of a live system. There are almost hundred active Linux distributions that provide a live medium listed at www.distrowatch.com, a website that provides overview on Linux distributions. In fact, the popularity of different distributions varies quickly but there are some base frames that are steadily gaining popularity. The family of Debian based distributions is the biggest of the base frames. The most famous member is Ubuntu and its fork Mint. The second big player is Fedora that is supported by the company RedHat. The next section deals with basics on operating system and the differences between the underlying concepts.

2.6.2 Operating Systems and Kernels

To facilitate the comprehension of the work, this section gives overview on the underlying concepts of operating systems. Most operating systems are using kernels and the basic idea is for all of them the same: provide the interface between hardware and software.

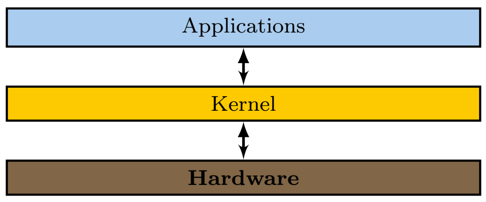


Figure 2.6: Layers on a PC system

As shown in figure 2.6 the kernel can be regarded as the connecting part between application and hardware layer. It provides the lowest level abstraction layer for resources (as devices, CPU and memory) that usually uses system calls and inter-process communication. [2]

There are two classical designs of kernels with different advantages. One concept known as monolithic kernels uses address space to execute all operating system code. The other concept is the family of micro kernels that are using the user space to run most parts of the operating system. The benefit is a modular implementation with better maintainability. On the other hand using servers in user space generates much more communication overhead compared with direct function calls used in monolithic kernels. That promises better performance of systems with monolithic kernels.

[2]For more detailed information about functionality of Unix based operating systems the lecture script of "Echtzeitbetriebssysteme" by Prof. Dr. Schneider (HAW) is highly recommended.

The evolution of operating systems led to new kernel concepts with the intention of combining advantages of both kernel implementations. The approach of hybrid kernels makes use of both concepts in the way of running some important services (e.g. network stack) in kernel space to reduce performance overhead. Specific codes like drivers are still running as servers in user space. Modern generations of Microsoft Windows and Apple Mac OS X belong to this group. Linux belongs to the family of Unix-like operating systems that use monolithic kernels. However the monolithic concept is enhanced of the functionality to provide loading of modules at running time. These modular kernels are able to load binaries to kernel space on demand. Once the background of live operating systems is considered, the next chapter focuses on the origin problem again.

3 Conception and Design

3.1 Use Cases

Before start thinking on concrete implementation some basic work on the definition of scenarios and the resulting requirements has to be done. The idea of textual, structural and visual modeling techniques for specifying use cases were formulated by Ivar Jacobson in 1986. Nevertheless, these techniques are part of modern software engineering approaches like eXtreme Programming. The template style used here is the casual use case structure defined by Alistair Cockburn (cf. Cockburn 2001, p. 236). This use case template consists of title (goal), primary actor, scope, level and the story. The story is the body of the use case and contains a short text that describes informally what happens. The description of possible scenarios is split up in two general points of view. On the one hand scenarios of end users like customers and potential new community members are shown. On the other hand there are the necessities of the company have to be taken into consideration.

3.1.1 Energy Data Monitoring System

The following use case include the scenarios that belong to the features of the live energy data monitoring system. They are formulated as generic as possible to enable different approaches. The idea is to document the interests in a short and clear way.

Use Case 1	**Test JEVis Energy Data Monitoring System**
Primary Actor:	End-User
Stakeholders and Interests:	• Customer: Get an impression of the software. • Training: Provide a training system for each participant. • Marketing: Showcase JEVis even if network is not available. Get in contact with interested parties. • Developer: Test developed software in a sandbox environment.
Scope:	JEVis Energy Monitoring System
Level:	User-goal
Main Scenario:	
	1. The actor is interested in testing the monitoring solution. He can get the software online or on a memory medium. With the help of the enclosed documentation the user can start the monitoring software and try it out by means of sample data.

Extensions:

1.a Communication:

1. The actor is informed about news concerning the monitoring software.
2. Extentions are offered to the actor after registration on website.

1.b Sandbox system:

1. The actor can save changes.
2. The actor can restore test system to original state.

1.c The actor can install the monitoring system.

To visualize the interests listed above a use case specialized UML diagram can be used. Figure 3.1 shows use case 1.

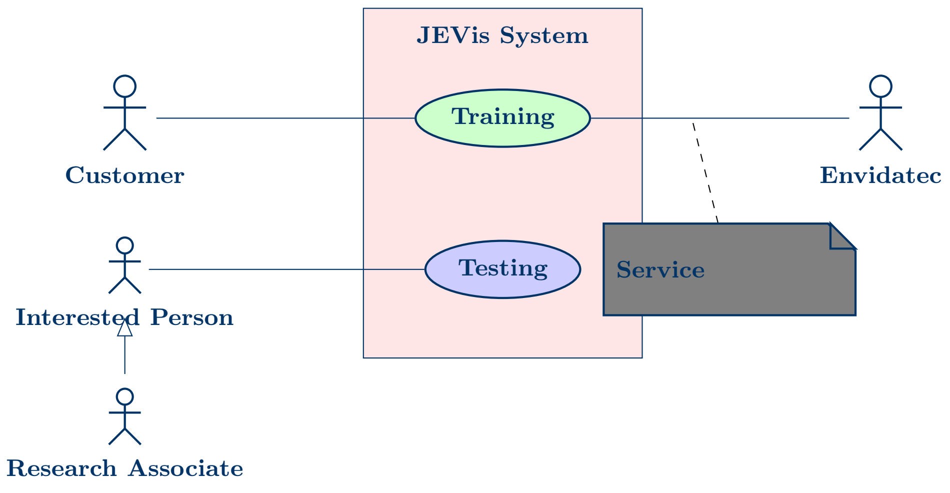


Figure 3.1: UML use case diagram of use case 1

The use case package "JEVis System" includes the use cases training and testing. The actors are associated to the use cases with lines. The actor "Interested Person" inherits the "Research Associate" actor. Figure 3.1 simplifies the tabular use case.

3.1.2 Creating a JEVis Monitoring Test System

The continuous development of Linux and the monitoring software leads to the use case of managing to create new versions of the live system. The stakeholder is in this case the company.

Use Case 2	**Create JEVis Monitoring Test System**
Primary Actor:	IT specialist of the company
Stakeholder and Interest:	Company: Offer easily up to date versions
Scope:	JEVis Energy Monitoring System
Level:	Company-goal

Main Scenario:

1. The actor can create a test system according to use case 1.
2. The actor can update the test system when a new version of JEVis data-monitoring solution is released.

Extensions:

2.a Customization:

1. The actor can customize the test system.
2. The actor can distribute the test system.

2.b Distribution:

1. The actor can choose between different distribution media.

On basis of the use cases the requirements on the solution are determined in chapter 3.2.

3.2 Requirements

As done before in the use cases chapter, the requirements on the test system and the creation process will be discussed separately. For professional development a standard of IEEE on software requirements specification exists. Although a specification of requirements according to the aforementioned IEEE 830 goes beyond the scope of this thesis, a minimal discourse follows. Requirements specify what features are expected. In the context of software engineering this means the expectation of a stakeholder on a software system. The term stakeholder generalizes all parties involved in system's requirements (cf. Balzert 2009, pp. 455-456). Requirements are classified in functional and non-functional ones. The functional group defines functions provided by the software system. Non-functional requirements by contrast define the way of doing. Requirements should fulfill the following characteristics: unitary, complete, consistent, atomic, traceable, current, unambiguous and verifiable (cf. IEEE Computer Society 1998, p. 4).

The figure 3.2 of the process splits the problem into three different aspects.

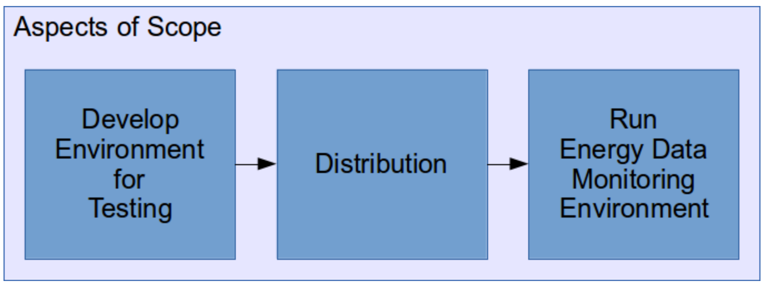

Aspects of Scope
Develop Environment for Testing
Distribution
Run Energy Data Monitoring Environment

Figure 3.2: The aspects of scope

The first aspect represents the requirements that result from use case 2. The second box deals with the options of distribution. The last and most important aspect contains the requirements on the implementation of the solution to provide the data monitoring system.

3.2.1 Data Monitoring Test System

The table below lists the direct (1) and indirect (2) requirements that result of requirement no. 1.4.

No.	Description
1	Test data monitoring system
1.1	Can run on PCs
1.2	Can run without network connection
1.3	Can display news
1.4	Gets along with any installation
1.5	Offers a complete JEVis system
1.6	Offers an installation routine for JEVis system
2	Environment
2.1	Installed Java JDK runtime environment
2.2	Installed and configurated Tomcat Java Servlet and server
2.2.1	Axis2 Web Services Engine
2.3	Installed and configurated database

Table 2.1: Requirements on the test system

3.2.2 Distribution

No.	Description
3	Distribution
3.1	Provide creation of copys
3.2	Supports various distribution channels

Table 2.2: Requirements on distribution

3.2.3 Solution to Create Test Systems

No.	Description
4	Creation
4.1	Process is reproducible
4.2	Can be done within appropriate time

Table 2.3: Requirements on the solution to create test systems

Once the requirements are defined the concept for further development can be fleshed out in the following.

3.3 Design

This section is about the design decisions. In the first step the requirements that call for decisions will be identified and the questions on design will be formulated. In the second step approaches to solve tasks begged by the questions are discussed and finally design specifications are generated.

3.3.1 Approach of a Live System

The requirements on the environment for the data monitoring test system (prefix 2) are leading to a central problem. As the section 2.3.1 about the structure of JEVis points out the requirements no. 2.1 to 2.4 are the basis for running JEVis. The JEVis Installation Manual (see OpenJEVis.org 2013[b]) shows on nine pages how to install and configurate Java JDK runtime environment, Tomcat Java Servlet Server, Axis2 Web Services Engine, MySQL Database and finally JEVis. This overhead for a simple installation contradicts to requirement no. 1.4 that implies no installation is necessary for testing the energy monitoring solution. Consistently the test solution needs to contain the environmental requirements. The options to afford this facility are the approaches of both portable applications and live systems. A Portable Application is a standalone program that runs on compatible operating systems without being installed. Although there exist portable applications that have managed the implementation of a needed environment[1] managing the major difficulty to accomplish the requirements on the environment.

[1]To name an example XAMPP is working as an completely portable webserver.

Moreover requirement no. 1.6 that implies offering an installation possibility is hard to realize.

The second approach of a Live System contains many capabilities. As already mentioned in section 2.6.1 several derivatives of live systems with highly diverse scopes already exsist. Almost all JEVis system installations up to now are running on Linux servers. While only a marginal group of desktop computers uses Linux, the number of servers running Linux keeps growing (cf. Vaughan-Nichols 2013). Even today working with Linux on servers is not unusual. Pondering the pros and cons of the possible solutions leads to the design decision to use a Linux Live System to manage the tasks.

3.3.2 Approach of a Remastering Program

In due consideration of requirements no. 4.1 and 4.2 it is incidental to automate the process of creating new versions of a live system. The program shall be able to guide the remastering process without resticting the options on customization. Furthermore, the program has to offer a possibility to copy the remastered live system onto media. Because of the necessary shell commands the program is only runnable on Linux operating systems. Many design decissions on the program design depend on the further work on the live system and can not be formulated at this stage. The next chapter starts with the implementation of remastering.

4 Implementation

This chapter is structured as follows: Each scope of implementation is justified, exemplary carried out and finally presented. Structuring the tasks like previously done in the requirements chapter lead to the topics remastering, distribution and live system as shown in figure 4.1.

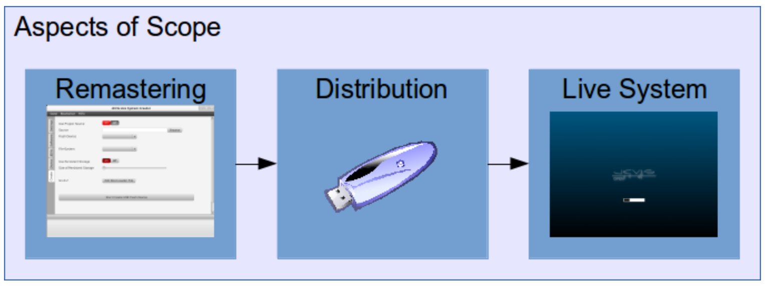


Figure 4.1: Aspects of scope

The result of an energy data monitoring live system requires a distribution and the process of remastering. The term remastering is taken from the audio production process and describes the process of customizing an operating system or software. Taking account of the structure of operating systems which was topic of section 2.6.1 remastering of an operating system requires many steps to customize the different parts of it.

4.1 Remastering

Before the decision to implement remastering with low level commands was taken, existing tools were tested for their suitabiliy for the task of creating a data monitoring live system. Systems that are qualified to be base frame are Knoppix, Ubuntu and Fedora. Taking into account the wide distribution, dynamics of development, and good community structure Ubuntu is favored.

For Ubuntu exists the customization tool “Ubuntu Customization Kit”. The user can enter a path to an up to date Ubuntu image. With UCK it is possible to chose the display manager, set names, and add software. The tool offers to create a set of languages that will be available on live system.

> UCK is a tool that helps you customizing official Ubuntu Live CDs (including Kubuntu/Xubuntu and Edubuntu) to your needs. You can add any package to the live system like, for example, language packs, applications, etc.
>
> Features: Create bootable LiveCD with predefined languages based upon an original Ubuntu/Kubuntu live CD using graphical wizard. Build live CD with special features using scripts. It is possible to customize the root filesystem (for example install/remove packages), ISO contents (add/remove docs, change names) and initrd (add modules to boot, change boot sequence).
>
> (see Balliano and Lichota 2013)

An other approach is implemented via the Remastersys tool. This program offers the option to create a backup or a live system from the running operating system. Many customizations like default live session user, choice of plymouth boot theme or GRUB background image can be applied. This program appeals to be most matured of all tested alternatives. Certainly the options for customizing other modules of the live system are not given and on testing occurred problems.

> Remastersys is a tool that can be used to do 2 things with an existing Debian, Ubuntu or derivative installation.
>
> It can make a full system backup including personal data to a live cd or dvd that you can use anywhere and install. It can make a distributable copy you can share with friends. This will not have any of your personal user data in it. The resulting iso file can be used on any other PC that still meets the original minimum requirements of Ubuntu or Debian. Things like the graphics card and other hardware will be configured and setup automatically and you do not have to use identical hardware. Ubuntu’s live boot tool, casper, currently blacklists Nvidia and AMD proprietary drivers so they will not be available on the live system and will need to be reinstalled after installation of your custom system.

(see Remastersys.com 2013)

The main disadvantage of the tested tools is the missing possibility to run the database server during the remastering process. For the installation of JEVis a database is required. Taking into account the previously listed problems the way of manual remastering which means doing it without existing programs with graphical user interface is chosen.

4.2 Remastering in Detail

To understand the functionality of live systems and spot the steps of remastering the parts of a live system are shown in 4.2 and are discussed in the following.

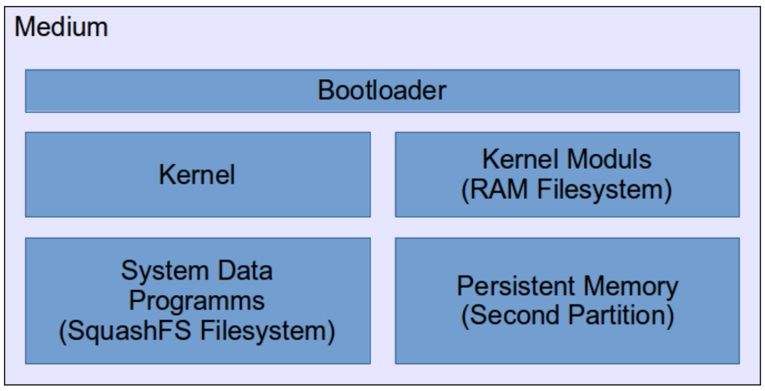

Medium
Bootloader
Kernel
Kernel Moduls (RAM Filesystem)
System Data Programms (SquashFS Filesystem)
Persistent Memory (Second Partition)

Figure 4.2: Components of a live system

Usual mediums to distribute operating systems are compact discs (CD), digital versatile disc (DVD) and newly USB flash drives or flash media cards. The crucial difference between disks and flash drives is the option of savings on the medium at runtime. Furthermore the time needed for loading the system depends on used medium. The medium contains a bootable partition, bootsectors and is marked as bootable. This enables the start of bootmanager when BIOS calls system start on the device. The bootmanager organizes start of basic operating system with optionally state of arguements. As explained in 2.6.1 linux basic systems are made up of kernel and kernel modules. For live systems this includes a lot of drivers in basic configuration. To complete the setup basic programs like a window manager are provieded. The possibility of persistent storage can be realized by adding a partition or a persistent file.

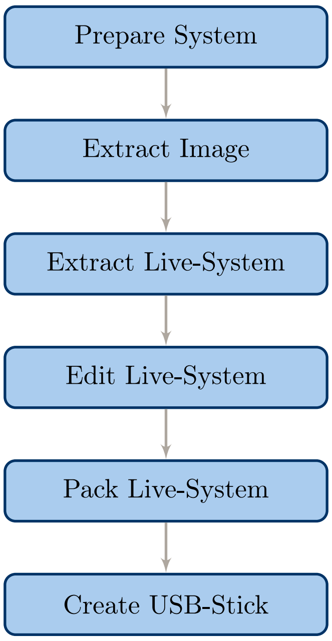


Figure 4.3: Sequence of remastering process

The remastering steps shown in figure 4.3 result from the knowledge of the components of live system media. The different images are extracted into folders on the host system. Once the data is copied changes can be applied and finally the new system can be packed again. The following paragraphs are dealing with more detailed information about the steps in the remaster process.

Prepare Creating System

Starting point for the system to create a new version of live system is Ubuntu 12.04 LTS installation with default software selection. In expectation of using a 32 bit architecture image it is a good idea to use a creating system with same architecture. Some additional tools are needed during remastering process.

```
sudo apt-get install squashfs-tools
```

Listing 4.1: Prepare

To install the needed programs the command line tool `apt-get` is used. Squashfs-tools are needed to extract and compress file system images in squashfs file system format. In the following many commands need higher rights. These rights can be granted with the `sudo` command.

of lines

Extract Image

```
 1 # Make Work Folder and Enter
 2 mkdir JEVisLive_v${VERSION}
 3 cd JEVisLive_v${VERSION}/
 4 
 5 # Mount Image and Copy
 6 mkdir iso
 7 sudo mount -o loop ${SRCIMAGE} ./iso/
 8 mkdir cpiso
 9 sudo rsync -a ./iso/ ./cpiso/
10 sudo umount ./iso
11 rmdir ./iso
```

Listing 4.2: Extract Image

By using the `mount` and the `rsync` commands (line 7-10) a copy the content of an Ubuntu hybrid image[1] is saved to the remaster directory. Many tries of remastering failed because of using the `rm` command. It seems like some files or links are not copied correctly with it. The workaround on this problem is using `rsync` for exactly copying files. Related to figure 4.2 the components are now located in the directory as listed in table 2.1.

[1]Images that can be used to create CDs and USB media

Part	Location
Bootloader configuration	boot/grub/grub.cfg
Kernel	casper/vmlinuz
Kernel modules (RAM filesystem)	casper/initrd.lz
System data (SquashFS filesystem)	casper/filesystem.squashfs
Persistent memory	Second partition labeled as *casper-rw*

Table 2.1: Location of system contents

Extract Live-System

The compressed image that contains the system data of the live system gets unpacked in a separate directory.

```
sudo mount -t squashfs -o loop ./cpiso/casper/filesystem.squashfs ./casper/
mkdir cpcasper
sudo rsync -a ./casper/ ./cpcasper/
sudo umount ./casper
rmdir ./casper
```

Listing 4.3: Extract Live-System

When the copy with `rsync` was successful the mount-folder can be deleted. The copy is now the starting point for editing the live system.

Edit Live-System

To customize the operating system the extracted live system can be entered with the `chroot` command. With this command it is possible to change the root directory on terminal. Several tasks have to be done to customize the system. Many of them can be realized by using the Debian Package concept which is topic of section 4.6.

```
# Prepare Chroot
sudo cp /etc/resolv.conf /etc/hosts ./cpcasper/etc
mkdir ./resources
```

```
sudo mount -o bind ${RESOURCES} ./cpcasper/mnt

# Enter Chroot Install Software and Create Default User
sudo chroot ./cpcasper/
  mount -t proc none /proc
  mount -t sysfs none /sys

  # Add Repositorys
  sudo add-apt-repository --yes "deb␣http://archive.ubuntu.com/ubuntu␣$(lsb_release␣-sc)␣
      universe"
  sudo add-apt-repository --yes ppa:webupd8team/java
  sudo add-apt-repository --yes ppa:gwendal-lebihan-dev/cinnamon-stable

  sudo apt-get update
  # sudo apt-get upgrade

  # Install Java
  sudo apt-get install oracle-java6-installer
  echo "JAVA_HOME=/usr/lib/jvm/java-6-oracle/" | tee /etc/environment

  # Remove Software
  sudo apt-get purge openjdk-*
  sudo apt-get purge gnome-games-data
  sudo apt-get purge unity-lens-shopping
  sudo apt-get purge rhythmbox

  sudo dpkg --install /mnt/plymouth-theme-openjevis.deb
  sudo update-alternatives --config default.plymouth
  sudo apt-get install --yes cinnamon

  # JEVis Installation Dependencies
  sudo apt-get install --yes tomcat6 libtcnative-1 mysql-server openssl unzip octave
  echo "JAVA_HOME=/usr/lib/jvm/java-6-oracle/" | tee /etc/default/tomcat6

  # Add Default User
  adduser jevis
  # passwd -d jevis # Remove Password
  usermod -aG sudo jevis
exit

# Enter Chroot with Default User
sudo chroot ./cpcasper/
  su jevis
    # JEVis Installation
    sudo mkdir /opt/jevis
    sudo chown jevis:jevis /opt/jevis
    sudo usermod -a -G tomcat6 jevis
```

```
50      exit
51    su jevis
52      unzip /mnt/JEVis_2_2_0_20121026.zip -d ~/
53      cd ~/JEVis_2_2_0_20121026/
54      java -jar ./JEInstaller.jar
55      mv /var/lib/tomcat6/webapps/ROOT /var/lib/tomcat6/webapps/OLDROOT
56      ln -s /var/lib/tomcat6/webapps/jevis /var/lib/tomcat6/webapps/ROOT
57      sudo service mysql restart
58      sudo service tomcat6 restart
59    exit
60  exit
```

Listing 4.4: Edit Live-System

The preparation of the chroot environment is done in lines 1-4 and includes overwriting the network configuration with the files of the host system. This is necessary to etablish a connection within the chroot environment. Additionally a shared folder is linked to the environment. Up next one of the most important steps for remastering is taken: The `chroot` command allows to change the root directory within the terminal window. Furthermore the current user of the terminal changes to root user. This implies the possibility to apply changes to the content of the directory in the way it can be done usually only within a running system. The proc and sys folders contain system information which is necessary for execution of programs. The `mount` commands (lines 7-10) provide the integration of this file system contents. At this point the requirements to install and uninstall software are fulfilled. Lines 11 to 35 illustrate the removal and the adding of software packages and repositories exemplarily. The following lines perform the installation and configuration of the data monitoring system. The biggest problems while the installation of JEVis as illustrated are caused by system services: the Tomcat server and the MySQL database service have undocumented requirements on the chroot environment. For example the installation fails when a MySQL service is already running on the host system. Furthermore the re-logins done are required to apply the set user-rights.

Pack Live-System

When the customization of the live system is finished the directory containing the edited files has to be packed again. Furthermore a list of installed packages is needed. The *filesystem.manifest* file lists packages used on the live system. The *filesystem.manifest-desktop* contains a list that is needed for a system installation of the distribution.

```
sudo su
  # Generate Package-Lists
  chroot ./cpcasper dpkg-query -W --showformat='${Package} ${Version}\n' > ./cpiso/casper
      /filesystem.manifest
  cp ./cpiso/casper/filesystem.manifest ./cpiso/casper/filesystem.manifest-desktop

  # Compress Live-System
  rm ./cpiso/casper/filesystem.squashfs
  mksquashfs ./cpcasper/ ./cpiso/casper/filesystem.squashfs

  # Override Initrd.lz
  cp ./cpcasper/boot/initrd.img* ./cpiso/casper/initrd.lz
exit
```

Listing 4.5: Pack Live-System

Create USB-Stick

Finally the USB flash device has to be prepared for the generation of live system device. In order to this old partitions are removed, new partitions are created, formatted, and the edited system is copied to the device.

```
DEV=/dev/sdc

# Create Partions
sudo umount ${DEV}*
sudo parted -s ${DEV} rm 1
sudo parted -s ${DEV} rm 2
sudo parted -s ${DEV} rm 3
sudo parted -s ${DEV} mklabel msdos
sudo parted -s ${DEV} mkpart primary fat32 5 30%
sudo parted -s ${DEV} mkpart primary ext2 30% 60%
sudo parted -s ${DEV} mkpart primary fat32 60% 100%
sudo parted -s ${DEV} set 1 boot on
sudo mkfs.vfat -n JEVisLive ${DEV}1
sudo mkfs.ext2 -L casper-rw ${DEV}2
sudo mkfs.vfat -n Resource ${DEV}3

sudo umount ${DEV}1

# Copy
sudo mkdir ./par1
```

```
21 sudo umount ${DEV}1
22 sudo mount ${DEV}1 ./par1
23 sudo rsync -a ./cpiso/ ./par1
24
25 # Install Bootloader
26 sudo grub-install --boot-directory=./par1/boot --no-floppy ${DEV} --force
27 sudo cp ./resources/grub.cfg ./par1/boot/grub/grub.cfg
```

Listing 4.6: Create USB-Stick

To copy the live system onto a USB flash device the media first has to be partitioned and formatted. This tasks are done with lines 4 to 17. After the preparation of the media is complete the system is copied to the device (line 23). Finally the boot manager has to be installed onto the media und the configuration of it gets overwritten (lines 26 and 27).

4.3 Customization

There are certain parts of a live system that can be customized to match the style guidelines of the community. In addition the data included in delivery has to be defined.

4.3.1 Look and Feel

The desired aim of an uniform appearance of products, projects and documentation implies customizing parts of the live system. Some components are editable with appropriate effort. Regarding a live system from users point of view the boot routine is the first seen component which is editable. The boot manager offers two operating modes: The default mode starts the system with the opportunity of savings while the save mode does not permit storage of data. Once the user has chosen an option the boot routine continues. It encloses loading of kernel, kernel-modules and systems data while displaying a boot animation. The live system uses a customized boot screen which is implemented on a system named Plymouth. There exist examples of creating own boot animations. When booting is finished an automatic log in with the default JEVis user is performed. Now the display manager is already started and the user is able to handle the system. Assuming that most of the users are familiar with Microsoft Windows the Cinnamon Display Manager that is similar to it is installed.

4.3.2 JEVis

To test the monitoring system feasible data to analyze are necessary. A set of suitable data for testing exists on the OpenJEVis Alpha Test Server. Data can be copied with the `mysqldump` command. During the remastering process the exported file has to be imported to the live system's database. No longer used data have to be deleted and additionally a clean up of the database has to be performed.

4.3.3 Presentation

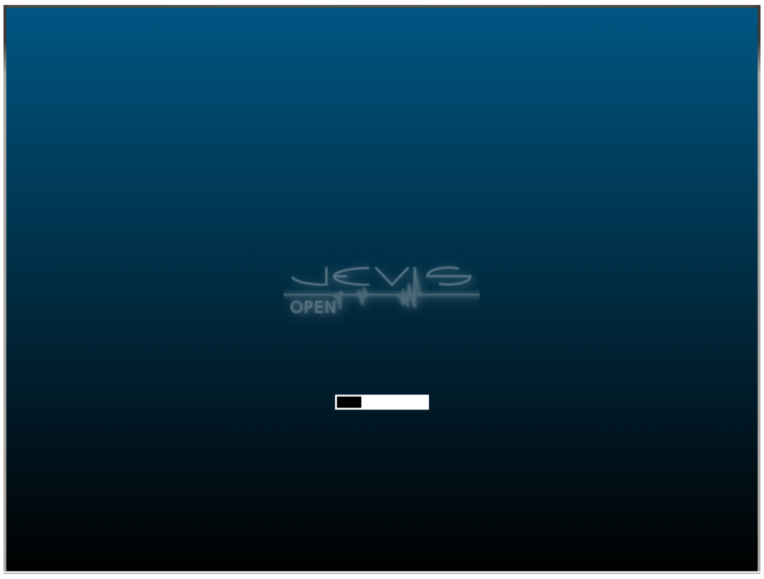


Figure 4.4: Customized boot screen

Figure 4.4 shows the animation that is displayed while starting the system. The logo is a transparent png image and the animation scripts enables glowing of the images contours.

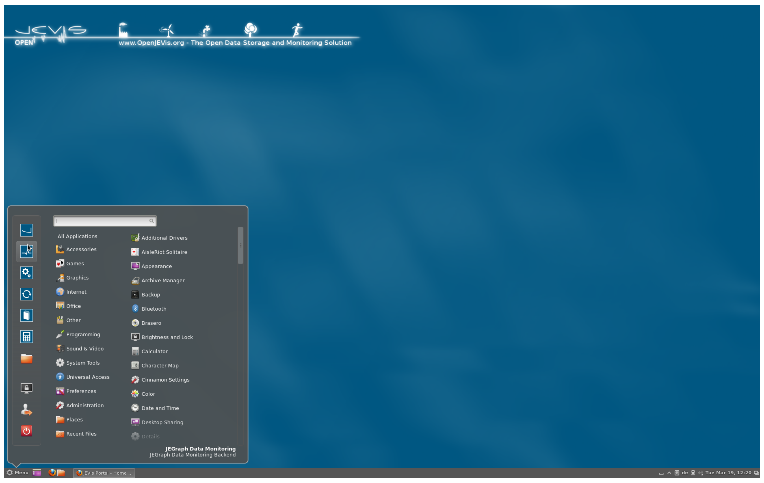
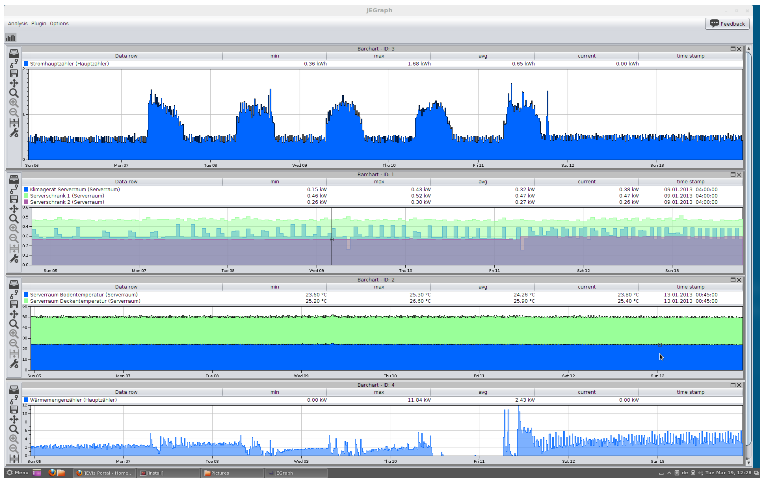


Figure 4.5: Desktop environment

Figure 4.5 shows the desktop provided by the Cinnamon display manager. The customizations aims to create a intuitive for most users with similar desktop concepts to Microsoft Windows.



Barchart - ID: 3

Data row	min	max	avg	current	time stamp
Stromhauptzähler (Hauptzähler)	0.36 kWh	1.68 kWh	0.65 kWh	0.00 kWh	

Barchart - ID: 1

Data row	min	max	avg	current	time stamp
Klimagerät Serverraum (Serverraum)	0.15 kW	0.43 kW	0.32 kW	0.38 kW	09.01.2013 04:00:00
Serverschrank 1 (Serverraum)	0.46 kW	0.52 kW	0.47 kW	0.47 kW	09.01.2013 04:00:00
Serverschrank 2 (Serverraum)	0.26 kW	0.30 kW	0.27 kW	0.26 kW	09.01.2013 04:00:00

Barchart - ID: 2

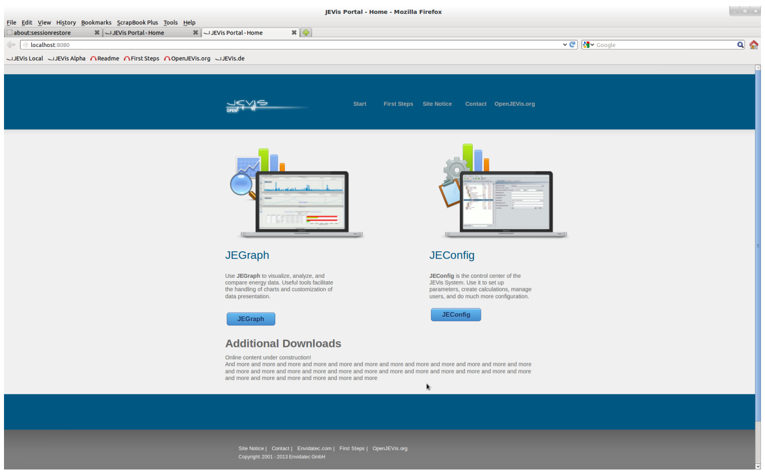
Data row	min	max	avg	current	time stamp
Serverraum Bodentemperatur (Serverraum)	23.60 °C	25.30 °C	24.26 °C	23.80 °C	13.01.2013 00:45:00
Serverraum Deckentemperatur (Serverraum)	25.20 °C	26.60 °C	25.90 °C	25.40 °C	13.01.2013 00:45:00

Barchart - ID: 4

Data row	min	max	avg	current	time stamp
Wärmemengenzähler (Hauptzähler)	0.00 kW	11.84 kW	2.43 kW	0.00 kW	

Figure 4.6: Test the Monitoring Solution

The JEVis Live System contains a customized portal page. This page is able to detect an internet connection and can display content loaded from the update.openjevis.org server. If no connection is available the portal page displays the default content.



JEGraph

Use **JEGraph** to visualize, analyze, and compare energy data. Useful tools facilitate the handling of charts and customization of data presentation.

JEConfig

JEConfig is the control center of the JEVis System. Use it to set up parameters, create calculations, manage users, and do much more configuration.

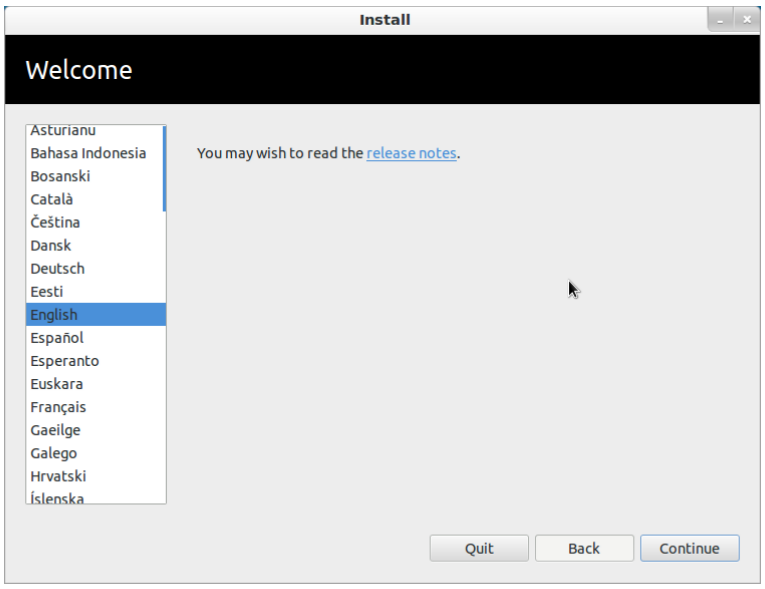
Additional Downloads

Online content under construction!
And more and more

Site Notice | Contact | Envidatec.com | First Steps | OpenJEVis.org
Copyright: 2001 - 2013 Envidatec GmbH

Figure 4.7: Test the Monitoring Solution

The already included example data enables the possibility to test the functions of the JEVis programs. For the most users JEGraph is the starting point like shown in 4.7 For advanced users jobs like MATLab calculations can be configured with JEConfig and the menu items provide the possibility to start the calculation, which is on a normal system started periodically.



You may wish to read the release notes.

Figure 4.8: Installation Routine

For users that decides to work with JEVis the installation routine of the live system is useful. The dialog captured in figure 4.8 guides the installation and offers many options for the installation.

4.4 Distribution

The options on distribution of the presented live systems are DVD's, offering download and USB flash devices. The option of using DVDs as media is impracticable because savings are not possible. Envidatec's management decided to use the option of USB flash devices. The reasons are the usability of them in connection with events like workshops and trainings.

4.4.1 Presentation

Figure 4.9: JEVis USB flash device

The chosen USB flash devices have a size of 8 GB and are labeled with the JEVis logo. The speed of the live system depends highly on the used flash media. While a high performance USB flash device boots the live system within 50 seconds [2] ordinary flash devices needs approximability two and a half minute.

4.5 JEVis Live Creator

The approach a of remastering tool previously discussed in section 3.3.2 aims to facilitate more comfortable editing of the live systems. That includes on the one hand the possibility to do settings on the customization and on the other hand to provide the procedure to copy it on

[2]tested with a Kingston DataTraveler Ultimate G2 3.0 on a USB 3.0 supporting PC.

flash devices. In consideration of the possibilities for program's design the questions arises which programming language is to use. Of course C has to be debated because it offers good access to command line functions. Even today many programs especially on Linux are written in C and indeed there are also libraries available to create graphical user interfaces. For example the GTK+ (Gimp Tool Kit) contains a lot of GUI components. Nevertheless an object orientated programming language establishes more comfortable development and usage of design patterns. Within the scope of the study of electrical engineering the opportunity of learning fundamentals on object orientated programming with Java exists. That encloses first steps with the Java Swing library. However the task of combining shell commands and a Java user interface begs problems. This will be discussed in section 4.5.2. Previously some general program design decisions have to be ruled.

4.5.1 Programming Techniques

The implementation makes use of the JavaFX library to create the graphical user interface. The program is realized with programming patterns. The following paragraphs describe the Singleton and the Model View Control (MVC) Pattern.

Singleton

The Singleton Pattern ensures that only one instance of a class is created. Listing 4.7 shows an example of a Singleton implementation in Java.

```java
public final class Singleton {
  private static Singleton theInstance = null;
  private Singleton() {};
  public static Singleton getInstance() {
    if (theInstance == null)
      theInstance = new Singleton();
    return theInstance;
  }
}
```

Listing 4.7: Singleton Pattern

The chosen implementation does not provide thread safety because the program does not make use of multithreading (cf. Eilebrecht and Starke 2010, p. 35).

Model View Control Pattern

The MVC Pattern splits the tasks within a program into three different parts shown in 4.10.

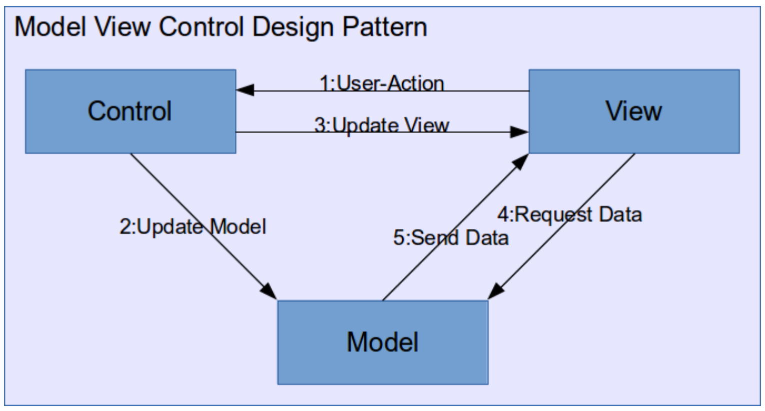


Figure 4.10: Model View Control Design Pattern (cf. Heinisch, Goll, and Müller-Hofmann 2007, p. 822)

The view is used for outputting representation. It requests information from the model that the view needs to generate output. The user sees the data shown by the view and uses the controller for commanding the program. The controller implements the event handling. It is able to send commands to its associated view and model to update them. Methods that touches data of the model belong to the model itself. The update of the view is bind to the state of the model which notifies its associated views and controllers when there has been a change in its state. This notification allows the view to produce updated output, and the controller to change the available set of commands (see Eckstein 2008). Other programming techniques used are explained in the following documentation of concrete implementations.

4.5.2 Execute Scripts within JAVA Programs

In exception of the tasks the program has to offer a solution to run scripts. This enables executing scripts generated by the script parser that is topic of section 4.5.3.

Executing scripts usually causes output that is written to the terminal window the script is launched. By launching the scripts within the program the output has to be redirected:

```java
static class StreamHandler extends Thread {
  InputStream is;

  String type;

  StreamHandler(InputStream is, String type) {
    this.is = is;
    this.type = type;
  }

  public void run() {
    try {
      InputStreamReader isr = new InputStreamReader(is);
      BufferedReader br = new BufferedReader(isr);
      String line = null;
      while ((line = br.readLine()) != null)
        System.out.println(type + "␣" + line);
    } catch (IOException ioe) {
      ioe.printStackTrace();
    }
  }
}
```

Listing 4.8: Stream

Java offers a runtime class to execute external programs. The path to bash which is the programm that executes shell scripts and the path to the script are combined to the command.

```java
// Executes the script
public static void execute(String script) {
  if (script.length()<1) {
    System.out.println("USAGE:␣java␣ShellScriptExecutor␣script");
    System.exit(1);
  }

  try {
    String osName = System.getProperty("os.name");
    String[] cmd = new String[2];
    cmd[0] = "/bin/sh";   // Path to shell
    cmd[1] = script;

    Runtime rt = Runtime.getRuntime();
    System.out.println("Run␣" + cmd[0] + "␣" + cmd[1]);
    Process proc = rt.exec(cmd);
```

```
      // Error messages
      StreamHandler errorStreamHandler = new StreamHandler(
          proc.getErrorStream(), "ERR");

      // Output messages
      StreamHandler outputStreamHandler = new StreamHandler(
          proc.getInputStream(), "OUT");

      // Start handlers
      errorStreamHandler.start();
      outputStreamHandler.start();

      // Error messages of process
      int exitVal = proc.waitFor();
      System.out.println("ExitValue:␣" + exitVal);
    } catch (Throwable t) {
      t.printStackTrace();
    }
  }
```

Listing 4.9: Module to run scripts

4.5.3 Script Parser

At this point the program is able to execute scripts. The next task is to find a way to combine the users choice in the graphical interface with the remastering scripts of chapter 4.2. This can be done according to figure 4.11 by reading a template script and replacing previous set wild-cards. This procedure offers the possibility to show the user the generated script before it gets executed.

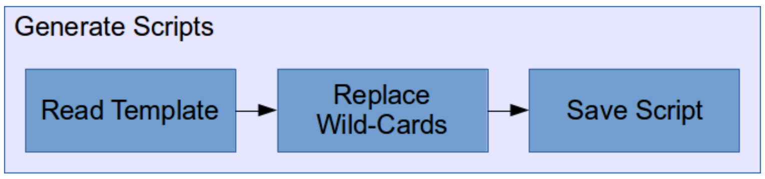


Figure 4.11: Principle of generating scripts

The interface shown in listing 4.10 specifies the necessary methods when programming a class that implements the interface.

```java
package org.openjevis.jevislivecreator.model;

import java.util.Map;

public interface Scriptable{
  /**
   * Initialize script. Template needs to be set
   */
  void initialize();

  /**
   * @return Map<string,string> var
   * set pairs to be replaced
   */
  Map<String,String> defVariables();

  /**
   * @return boolean status
   * start replace
   */
  Boolean replace();

  void setScript(String script);
  String getScript();
  void setTemplate(String template);
  String getTemplate();
  void setVars(Map<String, String> vars);
  Map<String, String> getVars();
  void generateScript(String script);
}
```

Listing 4.10: Interface of script creation

The abstract class provides shared used methods.

```java
package org.openjevis.jevislivecreator.model;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Map;

```

```java
public abstract class Script implements Scriptable{
  private String template;
  private String script;
  private Map<String,String> vars;

  public void setScript(String script){
    this.script=script;
  }
  public String getScript(){
    return script;
  }
  public void setTemplate(String template){
    this.template=template;
  }
  public String getTemplate(){
    return template;
  }
  public void setVars(Map<String, String> vars){
    this.vars=vars;
  }
  public Map<String, String> getVars(){
    return this.vars;
  }
  public void generateScript(String script){
    setScript(script);
    replace();
  }
  public Boolean replace(){
    try{
      // Open Files
      File infile=new File(getTemplate());
      if(!infile.exists()){
        System.out.println("Template␣not␣exists");
        return false;
      }

      File outfile=new File(getScript());
      if(!outfile.exists()){
        if(!outfile.createNewFile()){
          System.out.println("Script␣can␣not␣be␣created");
          return false;
        }
      }

      BufferedReader in = new BufferedReader(new FileReader(infile));
      BufferedWriter out = new BufferedWriter(new FileWriter(outfile));

      // Create map of variables
      setVars(defVariables());

      String line;
      // Read template file by line
      while((line = in.readLine()) != null){
        // For each key-value-pair
        for (Map.Entry<String, String> e : vars.entrySet()){
          // Replace key with value
          line=line.replace(e.getKey(), e.getValue());
        }
        // Write to script file
        out.write(line);
      }
      // Close files
      in.close();
      out.close();
    }
    catch( IOException e ){
      e.printStackTrace();
    }
    return true;
  }
}
```

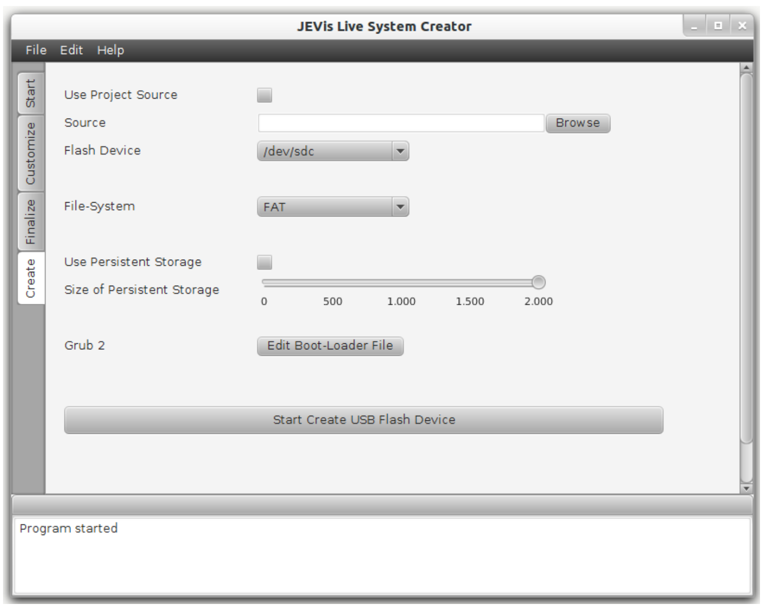
Listing 4.11: Abstract methods for script creation

To give an example of the functionality the implementation of the script that copies the system to an USB device is shown below.

```java
package org.openjevis.jevislivecreator.model;

import java.util.HashMap;
import java.util.Map;

public class ScriptCreate extends Script{

  @Override
  public Map<String, String> defVariables() {
    Map<String,String> vars = new HashMap<String, String>();

    // Model
    ModCreate modCreate=new ModCreate();

    // Add variables
    vars.put("$WORKDIR", model.getWorkDir() );
    vars.put("$DEV", modCreate.getDevice() );
    vars.put("$MOUNT_USB", modCreate.getUSBMount() );
    vars.put("$SIZE_LIVE", modCreate.getSizeParLive() );
    vars.put("SIZE_PERS", modCreate.getSizeParPers() );
    vars.put("$NO_PERS", modCreate.getPers() );
    vars.put("$useDef", modCreate.useDefaultIso.getValue().toString());
    return vars;
  }

  @Override
  public void initialize() {
    setTemplate(model.getScriptTemplateUsb() );
  }
}
```

Listing 4.12: Example for script implementation

4.5.4 Presentation



Use Project Source ☐
Source
Flash Device /dev/sdc
File-System FAT
Use Persistent Storage ☐
Size of Persistent Storage 0 500 1.000 1.500 2.000
Grub 2

Program started

Figure 4.12: Copy live system to media

Figure **??** shows the dialog to copy the JEVis Live System to a flash device.

4.6 Packages

Distributing software centralized offers benefits like predefined paths, enhanced configuration, version control, satisfy of dependencies, and update mechanisms. New versions of popular OS are mostly shipped with a build-in solution of software centers. While this feature on MS Windows has been implemented in 2012 (Windows 8) for the first time onto Linux systems concepts of an unified way to install software has been established for a long time. However there exist different implementations inside the distribution groups. One of the leading concepts is the RPM Package Manager [3], which has been developed since the 1990ies and has been specified in the Linux Standard Base. The RPM is for instance used by Fedora and Suse Linux. This packages are managed on client's side with the `yum` tool. The second important implementation is provided on Debian based distributions and are called Debian packages (.deb). They are managed with the `dpkg`, `aptitude` or usually `apt-get` tool.

Because Ubuntu Linux which belongs to the Debian based Linux derivatives is used for the realization Debian Packages are created. The procedure of making a package is exemplary shown in the following.

The file name of a Debian Package shall be built up containing the program's name, its version information and the architecture of target system.

```
programs-name_version.sub_version_debian_revision_version_architecture.deb
```

Debian packages have a default structure of folders and configuration files as shown in figure 4.13, that contains the default configuration for a local running JEVis system. The creation of the packages can be automated by the following script:

```
echo "=== Make Package ==="
SRC=./jevis-live-jevis-
DEB=./jevis-live-jevis.deb

read -p "Path to package sources: " -i $SRC -e SRC
read -p "Target file: " -i $DEB -e DEB

echo "Change permissions"
find $SRC -type f -print0 | xargs -0 sudo chmod 644   # set files to 644
find $SRC -type d -print0 | xargs -0 sudo chmod 755     # set folders to 755
find ./ -type f -name "*.sh" -print0 | xargs -0 sudo chmod +x # make scripts executable
```

[3] primary meaning RedHat Package Manager

```
#find ./ -type f -name "*.jar" -print0 | xargs -0 sudo chmod +x # make scripts executable

chmod 0755 -R $SRC/DEBIAN/postinst         # rwx, rx, rx to scripts
chmod 0755 -R $SRC/DEBIAN/prerm          # rwx, rx, rx to scripts

echo "Clean␣sources"
find $SRC -type f -name "*~" -exec rm -f {} \;       # delete gedit auto-save
rm -rf `find $SRC -type d -name .svn`        # delete svn info

echo "Make␣deb␣package"
fakeroot dpkg -b ${SRC} ${DEB}
lintian ${DEB}
```

Listing 4.13: Create a Package

To build packages of good quality the file permissions have to be set correctly. Furthermore, not needed files are deleted and the package is created under usage of a fakeroot environment. The program lintian offers a check of the built package.

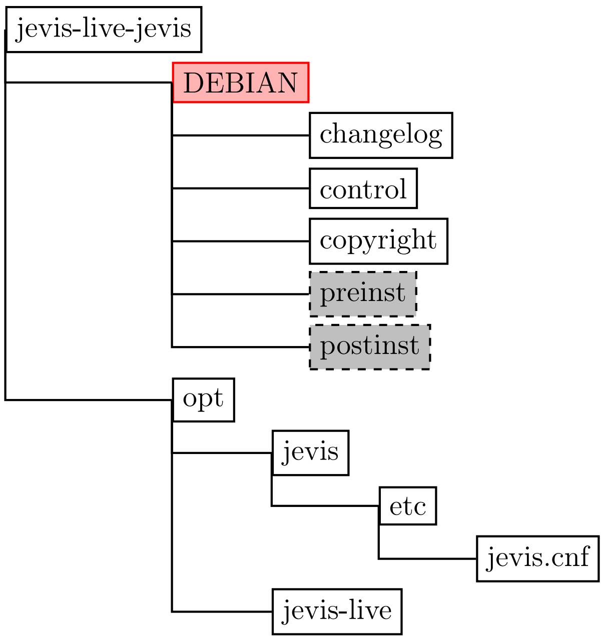


Figure 4.13: Folder structure of an example package

The *DEBIAN* folder contains configuration files. They are used to provide information about license, platform, depencies and more. Analog to the procedure sketched ahead within the work on the live system the packages listed in table 6.1 are created. These packages take part during the remastering process.

Name	description
jevis-live-bin-links	Places run scripts in bin.
jevis-live-casper	Place settings for casper environment.
jevis-live-cinnamon	Overrides default session-setting.
jevis-live-defaultconf	Default pre-configuraton for JEVis-installation.
jevis-live-desktop-icon-jecalc	Shows JECalc button on desktop.
jevis-live-desktop-icon-jeconfig	Shows JEConfig button on desktop.
jevis-live-desktop-icon-jegraph	Shows JEGraph button on desktop.
jevis-live-desktop-icon-portal	Shows JEPortal butoon on desktop.
jevis-live-desktop-icon-readme	Shows readme-icon on desktop.
jevis-live-documentation	Local documentation of JEVis.
jevis-live-exampledata	Copy example data to JEVis system.
jevis-live-fix-apport	Suppress crash reports of apport.
jevis-live-icons	Icons for JEVis.
jevis-live-jeconfig	JEConfig - set up the monitoring system configuration.
jevis-live-jevis	Copy and run JEInstaller.
jevis-live-languages	Install language packages (DE,RU,EN,UK)

Table 6.1: Created packages

4.6.1 Repository

To create an repository the command line program `reprepro` is used.

> “Creating an APT repository involves creating a set of directories to hold the packages and using a tool to create the supporting files for the APT tools to use to access the repositories.”
>
> (Sally 2009)

In the first step packages that are added to the repository have to be signed with with a GPG key.

```
# install tool to sign packages
sudo apt-get install dpkg-sign
```

```
# export key
gpg --armor --output ./repository/publickey.gpg --export info@openjevis.org
# sign a package
dpkg-sig --sign builder ./packages/jevis-live-icons.deb -k 47E56ABD
```

Listing 4.14: Sign packages

Next steps are creating a folder named `conf` and a configuration file named `distributions` in it. The repository needs the configuration to organize clients access to the repository. Clients need to know what types of packages they can expect to find.

```
Origin: openjevis.org
Codename: experimental
Architectures: i386 amd64
Components: main
Description: Debian-Packages of OpenJEVis.org
SignWith: 47E56ABD

Origin: openjevis.org
Codename: quantal
Architectures: i386 amd64
Components: main
Description: Debian-Package of OpenJEVis.org
SignWith: 47E56ABD
```

Listing 4.15: Distributions configuration file

After the repository is set up package files can be added to the repository with this command:

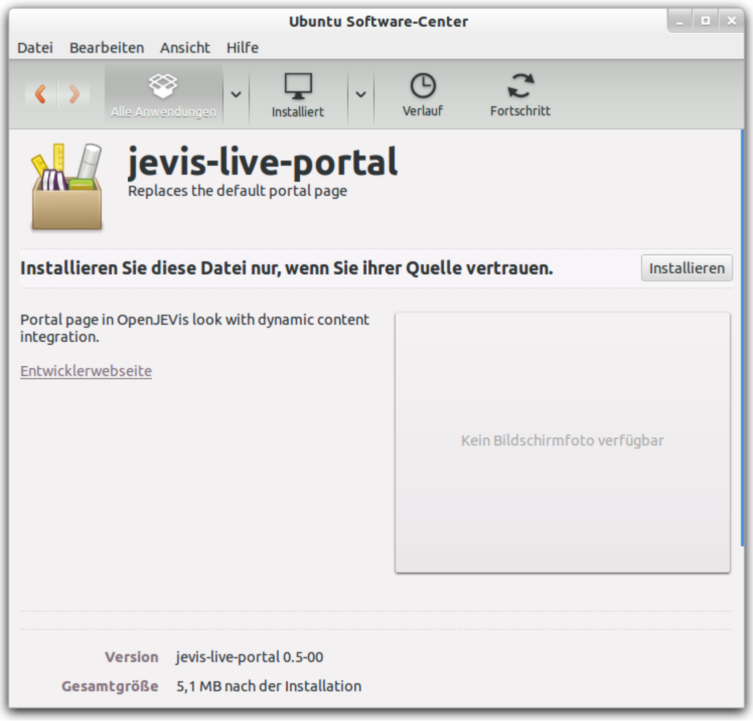
```
# install needed tool
sudo apt-get install reprepro

# set repository base dir
export REPREPRO_BASE_DIR='/repository'

# add packages with reprero
reprepro includedeb experimental jevis-live-icons.deb

# To remove a package from
reprepro remove experimental jevis-live-icons
```

Listing 4.16: Add and remove packages

Subsequently the content of the repository folder has to be uploaded except of `db` and `conf` which are only needed locally for configuration.

4.6.2 Presentation

Once the repository is available clients can adapt it to their package sources. The procedure is described briefly at `http://update.openjevis.org`.



jevis-live-portal

Replaces the default portal page

Installieren Sie diese Datei nur, wenn Sie ihrer Quelle vertrauen.

Portal page in OpenJEVis look with dynamic content integration.

Kein Bildschirmfoto verfügbar

Version	jevis-live-portal 0.5-00
Gesamtgröße	5,1 MB nach der Installation

Figure 4.14: Ubuntu software center

Figure 4.14 shows how packages are presented on the software center. If new versions of the packages are uploaded to the repository clients are notified about the update. The short presentations of the results are associated within the last chapter and the impact on the initial situation is revealed.

5 Conclusion and Outlook

5.1 Conclusion

To estimate the impact of the implemented improvements on usability of OpenJEVis, and especialy the necessary efforts to test and install JEVis, subsequently the initial and actual situation have to be compared. The fields of initial usability and distribution infrastructure became the main topics of this thesis. To remind the reason for starting research on distribution alternatives figure 5.1 shows the weaknesses again.

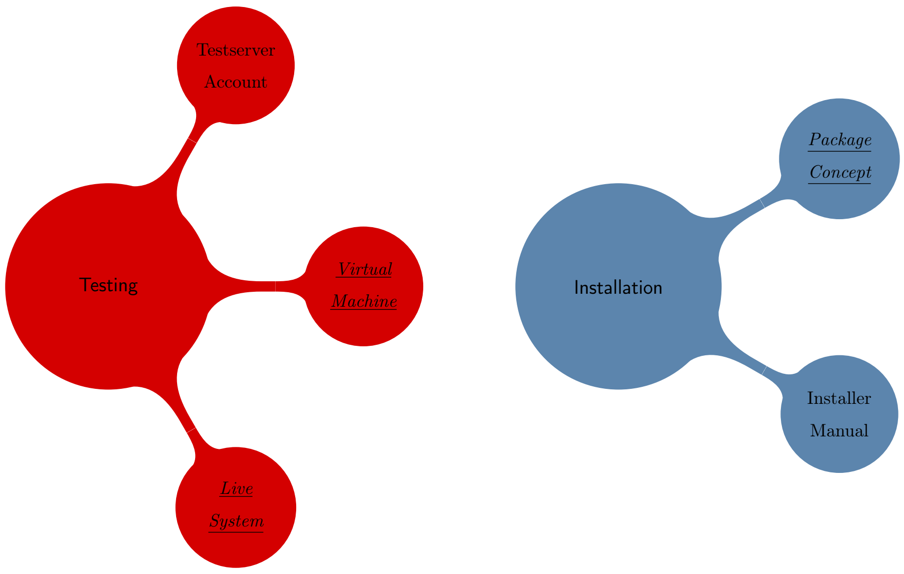


Figure 5.1: Fields of work

The figure displays the existing and the within the project planned (underlined italic) options of testing and installing. The initial situation required the registration on the website and requesting of a guest account to try out JEVis without complex installation. The only way to install JEVis was given by the JEInstaller and a nine page long installation manual.

The main efforts were taken on the development of the JEVis Live System. The final result of the research on it is a solution that eliminates many weak points. By taking a closer look on intentions and actual results the live system approach provides a crucial easier way for testing. The live system is able to save data, do calculations, display news, perform installation and contains example data. Up to this point about 100 JEVis Live USB flash devices have been produced. Particularly new possibilities to perform installations were created which are discussed later on. For time reasons the planned feature of integrating a data source was not implemented.

Aiming at creating a sustainable solution becomes apparent on the developed program. The *JEVis Live System Creator* offers creating new versions of the Live System and the reproduction of Live System USB flash devices. A graphical user interface based on JavaFX and the realization of remastering tasks which go back to terminal scripts are chosen in this implementation. The adaption of scripts bases on templates and a parsing algorithm of the program. Adapted scripts can be executed with the implemented shell script launcher.

New options besides the live system are offered by the concept of Debian Packages. The used synergies flow into alternative installation options. Within the scope of this thesis the presentation of possibilities are published on an own domain[1] which is integrated on the OpenJEVis.org community. Approaching to distribute the software via repository would offer a technical solution to perform updates more comfortable. Thus, this topic is examined in the following outlook section.

[1]The results are available on `http://update.openjevis.org/`

5.2 Outlook

5.2.1 Integration of Datasources

The initial purpose includes the integration of datasources. As already mentiond in the previous section the current version of the JEVis Live System does not implement an own active datasource. The reflections done on this topic during working on this thesis are used to outline possibilities of datasource integration in the following.

Normally the live system is only running for limited periods. That causes undynamic data for long sequences. Interrupted data recording has impact on the group of matching sources because hardly any of them can provide definite data rows. Some sources provide data that changes only while the system is running. As a consequence values for undetermined data sequences can be reconstructed. Examples for this kind of data are disk space or network traffic that do not change when the system is turned off.

Other data values change while the system is not running. In times the recording service does not capture data the values are undetermined. This thought can be proved taking CPU temperatures: Sensors can only deliver temperatures in uptime but the value will change after turning off the system.

Data Sources

There are different possibilities to realize the transmission of data to the JEVis software. The concepts differ in level of abstraction and deepness of access to the monitoring software.

Taking into account the limitation that the data source is only used on local system a service program is a solution to connect data sources to the monitoring system. Within the JEVis architecture the JECAPI contains the functionality of basic input- and output-operations.

Former devices often use the way of a direct connection to the energy monitoring system as shown in the figure above. Direct binding leads to the benefit of less overhead and thus a good performance. The following described approach enables a more flexible and easier possibility to realize binding energy meters.

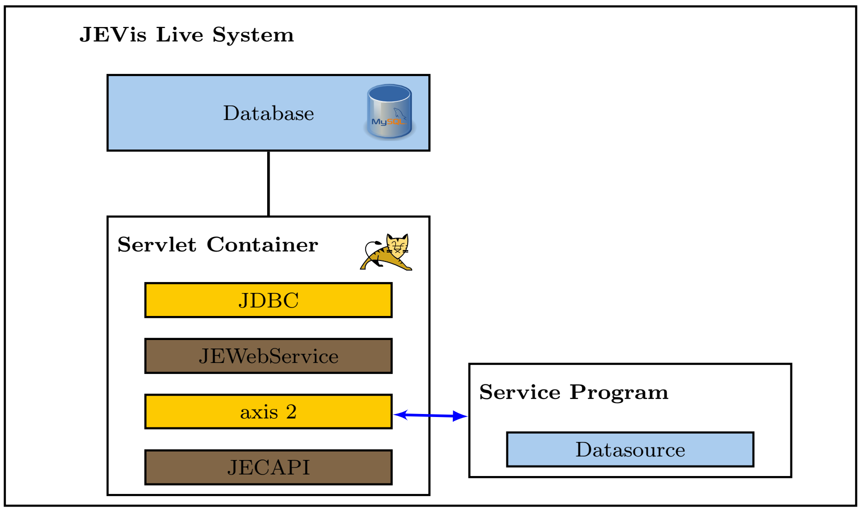


Figure 5.2: JECAPI direct binding

JEADFWeb Concept

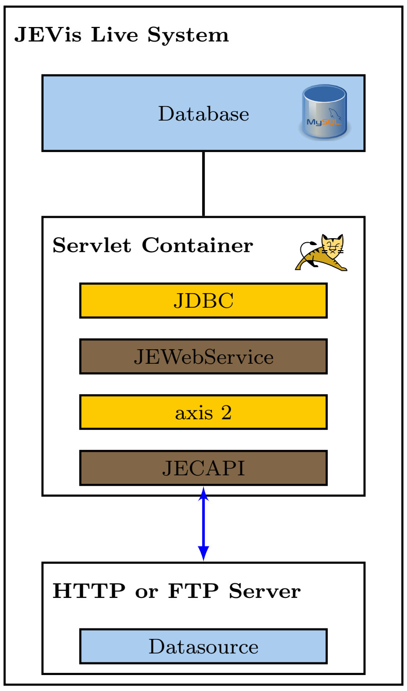


Figure 5.3: JEADFWeb concept

The JEADFWeb concept sketched in figure 5.3 is the common used solution to connect energy meters. As denoted within the 2.3.1 section it supports various protocols. The better choice to demonstrate function principle, advantages and conception of JEVis is therfore the JEADFWeb concept.

5.2.2 Further Projects

As mentioned before Envidatec GmbH cooperates with different universities in various countries to expedite research and development and education, too. A concept of laboratory experiment at Perm National Research Politechnic University in Russia contains the installation of measurement hardware. The laboratory places shall be equipped with computers running JEVis energy data monitoring software. For this concept the JEVis Live System can be considered as a valuable innovation especially for pre- and postprocessing experiments.

Another point deals with the delivery of drivers and custom written software to customers. The idea of Envidatec's management is to send them a portfolio including the manual and an USB flash device containing the ordered software and additionaly the JEVis Live System. Boxing software, even if a physical media is not mandatory, is not unusual, e.g. mass market software like office products.

All in all these examples prove that the project contributes to future development both in the company and for interested open source users.

Glossary

C programming language is a general-purpose programming language initially developed by Dennis Ritchie between 1969 and 1973 at AT&T Bell Labs.

Canonical Ltd.) is a UK-based privately held computer software company founded (and funded) by South African entrepreneur Mark Shuttleworth to market commercial support and related services for Ubuntu and related projects.

Chief Executive Officer is the highest-ranking corporate officer (executive) or administrator in charge of total management of an organization.

Cinnamon (user interface) is a user interface. It is a fork of GNOME Shell, initially developed by (and for) Linux Mint.

Debian is a computer operating system composed of software packages released as free and open source software.

Energy Management System is a system of computer-aided tools used by operators of electric utility grids to monitor, control, and optimize the performance of the generation and/or transmission system.

German Renewable Energy Act (in German: Erneuerbare-Energien-Gesetz, EEG) was designed to encourage cost reductions based on improved energy efficiency from economies of scale over time.

GNU Privacy Guard is a GPL Licensed alternative to the PGP suite of cryptographic software.

Graphical User Interface is a type of user interface that allows users to interact with electronic devices using images rather than text commands.

Institute of Electrical and Electronics Engineers is a professional association headquartered in New York City that is dedicated to advancing technological innovation and excellence.

International Organization for Standardization is an international standard-setting body composed of representatives from various national standards organizations.

ISO 50001 is a specification created by the International Organization for Standardization (ISO) for an energy management system.

Java programming language is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible.

JavaFX software platform is a software platform for creating and delivering rich internet applications (RIAs) that can run across a wide variety of devices.

JEConfig is a program to configure the JEVis software.

JEGraph is a program used to visualize energy data; it is a part of the JEVis software.

JEVis is a energy data monitoring software developed by Envidatec GmbH.

Linux is a generic term referring to the family of Unix-like computer operating systems that use the Linux kernel.

Model–View–Controller is a software architecture pattern which separates the representation of information from the user's interaction with it.

My Structured Query Language is an open source relational database management system.

Plymouth is a bootsplash for Linux. It supports animations. It makes use of Direct Rendering Manager (DRM) and kernel-based mode-setting (KMS). It gets packed into the initrd.

Primary energy is an energy form found in nature that has not been subjected to any conversion or transformation process.

Structured Query Language is a special-purpose programming language designed for managing data held in a relational database management systems.

Ubuntu is a computer operating system based on the Debian Linux distribution and distributed as free and open source software.

Overview of Commands

This contains an overview of commands used in the thesis with a short description.

add-apt-repository Adds a repository into the /etc/apt/sources.list or /etc/apt/sources.list.d or removes an existing one

adduser create a new user or update default new user information

apt-get is the command-line tool for handling packages, and may be considered the user's "back-end" to other tools using the APT library. Several "front-end" interfaces exist, such as synaptic and aptitude.

- **update** Used to re-synchronize the package index files from their sources. The indexes of available packages are fetched from the location(s) specified in /etc/apt/sources.list(5). An update should always be performed before an upgrade or dist-upgrade.

- **upgrade** Used to install the newest versions of all packages currently installed on the system from the sources enumerated in /etc/apt/sources.list(5). Packages currently installed with new versions available are retrieved and upgraded; under no circumstances are currently installed packages removed, nor are packages that are not already installed retrieved and installed. New versions of currently installed packages that cannot be upgraded without changing the install status of another package will be left at their current version. An update must be performed first so that apt-get knows that new versions of packages are available.

- **install** this option is followed by one or more packages desired for installation

cd change directory

cp copy files and directories

chown change file owner and group

chroot run command or interactive shell with special root directory

exit cause normal process termination

grub-install install GRUB to a device

ln make links between files

mkdir make directories

mkfs build a Linux filesystem

mksquashfs tool to create and append to squashfs filesystems

mount mount a filesystem

mv move (rename) files

nano small editor

parted a partition manipulation program

rsync a fast, versatile, remote (and local) file-copying tool

service run a System V init script

su run a shell with substitute user and group IDs

sudo execute a command as another user

umount unmount file systems

unzip list, test and extract compressed files in a ZIP archive

usermod modify a user account

List of Figures

List of Tables

List of Listungs

Bibliography

Balliano, Fabrizio and Krzysztof Lichota (January/2013). *Ubuntu Customization Kit.* Available online, visited on February 20. 2013. URL: {http://sourceforge.net/projects/uck/} (cit. on p. 26).

Balzert, H. (2009). *Lehrbuch Der Softwaretechnik: Basiskonzepte Und Requirements Engineering.* Lehrbücher der Informatik. Spektrum Akademischer Verlag GmbH. URL: {http://books.google.de/books?id=vmfIb9R2QikC} (cit. on p. 21).

Bandel, G. (2010). *Open Source Software: Fördert Oder Hemmt Diese Art Der Softwareentwicklung Den Wettbewerb?* Diplomica Verlag Gmbh. URL: {http://books.google.de/books?id=1498NifAmlEC} (cit. on pp. 9, 11).

Boßmann, Tobias, Wolfgang Eichhammer, and Rainer Elsland (2012). *Policy Report Contribution of Energy Efficiency Measures to Climate Protection within the European Union until 2050.* Fraunhofer ISI for the Federal Ministry for the Environment, Nature Conservation and Nuclear Safety (BMU) of Germany. URL: {http://www.isi.fraunhofer.de/isi-media/docs/e/de/publikationen/BMU_Policy_Paper_20121022.pdf} (cit. on p. 3).

Cockburn, A. (2001). *Writing effective use cases.* Agile software development series. Addison-Wesley. URL: {http://books.google.de/books?id=VKJQAAAAMAAJ} (cit. on p. 17).

Eckstein, Robert (2008). *Java SE Application Design With MVC.* URL: {https://blogs.oracle.com/JavaFundamentals/entry/java_se_application_design_with} (cit. on p. 40).

Eilebrecht, Karl and Gernot Starke (1/2010). *Patterns kompakt: Entwurfsmuster für effektive Software-Entwicklung.* 3. Aufl. 2010. Spektrum Akademischer Verlag. URL: {http://amazon.de/o/ASIN/3827425255/} (cit. on p. 39).

Envidatec GmbH (2012). *JEVis Energy and Operating Data System.* Internal presentation (cit. on p. 10).

— (Februrary/2013). *Company Profile. An Overview of the Company Envidatec GmbH, Reference Projects and Expertise.* Internal document (cit. on p. 6).

Hattenhauer, Rainer (4/2005). *Linux-Livesysteme: Knoppix, Ubuntu, Morphix, Kanotix, Mepis, Quantian & Co. (Galileo Computing).* 1. Negernbötel: Galileo Computing. URL: {`http://amazon.de/o/ASIN/3898426319/`} (cit. on p. 14).

Heinisch, C., J. Goll, and F. Müller-Hofmann (2007). *Java als erste Programmiersprache.* Teubner. URL: {`http://books.google.de/books?id=Q8\_zG0muPiEC`} (cit. on p. 40).

IEEE Computer Society, Software Engineering Standards Committee (1998). *IEEE recommended practice for software requirements specifications.* IEEE (std.) Institute of Electrical and Electronics Engineers. URL: {`http://books.google.com.ec/books?id=MYBGAAAAYAAJ`} (cit. on p. 21).

International Organization for Standardization (2011). *Win the energy challenge with ISO 50001.* URL: {`http://www.iso.org/iso/iso_50001_energy.pdf`} (cit. on p. 5).

Jones, P. (2008). *Knowing Knoppix: The Beginner's Guide to Linux That Runs from Cd.* CreateSpace. URL: {`http://books.google.de/books?id=KW-6TU19THkC`} (cit. on p. 14).

Kausch, P., J. Gutzmer, M. Bertau, and J. Matschullat (2011). *Energie Und Rohstoffe: Gestaltung Unserer Nachhaltigen Zukunft.* Spektrum Akademischer Verlag GmbH. URL: {`http://books.google.de/books?id=ixajES\_KlfsC`} (cit. on p. 1).

Knopper, Klaus and Kyle Rankin (2007). *Knoppix hacks - tips and tools for using the Linux live CD to hack, repair, and enjoy our PC: includes Knoppix on DVD (2. ed.).* O'Reilly, pp. I–XXV, 1–391 (cit. on p. 14).

Marinilli, M. (2002). *Java Deployment with JNLP and WebStart.* Kaleidoscope Series. Sams. URL: {`http://books.google.de/books?id=lbvUD6LUyV8C`} (cit. on p. 6).

Nüttgens, M. and E. Tesei (2000). *Open source - Marktmodelle und Netzwerke.* Veröffentlichungen des Instituts für Wirtschaftsinformatik. Iwi. URL: {http://books.google.de/books?id=bBMPPwAACAAJ} (cit. on p. 10).

OpenJEVis.org (2013[a]). *JEADFWeb. JEVis Automatic Data Fetch via Webprotocol.* URL: {http://openjevis.org/projects/jeadfweb} (cit. on p. 9).

— (February/2013[b]). *JEVis Installation Manual.* Available online. visited on March 1. 2013. URL: {http://www.openjevis.org/documents/12} (cit. on p. 23).

Open Source Initiative (2012). *The Open Source Definition.* Available online. URL: {http://opensource.org/osd} (cit. on p. 10).

Remastersys.com (January/2013). *Remastersys. A Unique Linux Backup to Live Media Tool for Debian and Ubuntu.* Available online, visited on February 18. 2013 (cit. on p. 27).

Sally, G. (2009). *Pro Linux Embedded Systems.* The Expert's Voice in Linux. Apress. URL: {http://books.google.de/books?id=3pBP5Y_fLDkC} (cit. on p. 49).

Vaughan-Nichols, Steven J. (2013). *Linux servers keep growing, Windows and Unix keep shrinking.* URL: {http://www.zdnet.com/blog/open-source/linux-servers-keep-growing-windows-and-unix-keep-shrinking/10616} (cit. on p. 24).

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Ort, Datum Unterschrift