

UNIVERSITY OF GRONINGEN  
Faculty of Mathematics and Natural Sciences  
Department of Computing Science  
Software Architecture



## System Architecture

Home Power Save

---

### GROUP 3

Bakker, Erik  
Jillings, Marcel  
Konterman, Gerjan  
Manteuffel, Christian  
Pieters, Harrie  
Suharto, Edy

1.0  
Groningen, November 14, 2011

## Authors

Name	Acronym	E-Mail
Bakker, Erik	eb	erikbakker20@gmail.com
Jillings, Marcel	mj	m.jillings@student.rug.nl
Konterman, Gerjan	gk	gkonterman@gmail.com
Manteuffel, Christian	cm	cm@notagain.de
Pieters, Harrie	hp	harriepieters@gmail.com
Suharto, Edy	es	edy.suharto@gmail.com

## Revision History

Version	Author	Date	Description
0.1	cm	13/09/11	Set up structure based on document template
0.2r	cm	15/09/11	First draft of the business vision
	mj	15/09/11	First draft of the business rationale, target audience, system evolution
	hp	15/09/11	First draft of the analysis - design alternatives
	gk	15/09/11	First draft domain model, roadmap
	eb	15/09/11	First draft costs/competitors
0.3r	cm	17/09/11	Titlepage logo, business vision, use cases
	mj	18/09/11	Changed business rationale and target audience according to first review
		20/09/11	New product description, improvements in all chapters
		22/09/11	Added more to risk assessment and assumptions
	eb	25/09/11	Product description sum up to text
0.4r			First draft of function requirements
	gk	25/09/11	First draft of function requirements
		27/09/11	Improvements to business and domain model
			Changed roadmap
	mj	25/09/11	Changed target audience, evolution requirements, risk assessment and assumptions according to first review
		26/09/11	First draft of technological roadmap and changes to evolution requirements
		27/09/11	Changed target audience and key drivers
	gk	27/09/11	Changed business model and domain model
	hp	27/09/11	Changed systemcontext, customer requirements and design alternatives
	es	27/09/11	Improvements to stakeholders
0.5r	mj	02/10/11	Changed target audience, requirements, requirements order and use-cases
	gk	02/10/11	Changed requirements
	hp	03/10/11	Changed design alternatives, review changes and first draft hardware architecture
		04/10/11	Draft version hardware architecture, sales and renting, review changes

0.6r	mj	04/10/11	First draft of hardware architecture Socket Meter, changed assumptions
		05/10/11	First draft of hardware architecture main device, Socket Meter and servers, changed functional requirements and assumptions
	eb	05/10/11	Requirements
	mj	10/10/11	Changed hardware overview and hardware architecture - Socket Meter according to review
	eb	10/10/11	Added Architectural vision and changed financial model
	hp	10/10/11	Changed consistency, hardware and design Alternatives
	mj	11/10/11	Second draft of hardware architecture - server, changed competitors
0.7r	gk	11/10/11	Added Initial models
	es	11/10/11	Improvement in stakeholders, draft verification
	eb	11/10/11	Improved requirements
	mj	17/10/11	Changed hardware architecture - server, according to review
		18/10/11	Changed target audience, hardware architecture decisions, and hardware architecture - hps, socket meter, server, according to review
0.8r	es	18/10/11	First draft process view
	eb	18/10/11	Architectural vision.
	hp	20/10/11	Finalized High-level Design Decisions.
	gk	23/10/11	Reworked Initial models/elaborated models to system context
0.9r	cm	23/10/11	Finalized Use-cases, Requirements First draft Software Architecture Improved Arch. Vision
	mj	23/10/11	Changed hardware architecture according to review, cleaned up financial model and some todos in the document
	es	25/10/11	Improvement on System Architecture verification
	mj	25/10/11	Draft for requirement verification + draft for deployment view
	eb	25/10/11	Changed Logical view
		26/10/11	Improvements to deployment view
	mj	30/10/11	Chapter introductions.
		31/10/11	Chapter introductions + draft for system evolution + review notes.
		01/11/11	Draft for requirements validation + improved system evolution.
	eb	01/11/11	Improved logical view + added database schema
1.0	es	01/11/11	Improvement on system verification
	hp	03/11/11	Improvement monitoring, reporting added policies component.
	gk	01/11/11	Added process view
	mj	13/11/11	Improvements throughout the document + changes throughout the document
	hp	11/11/11	Added web-interface, updated decisions, updated financial model
		12/11/11	Added MVC to the web-interface

	13/11/11	Added Architectural Evaluation, updated system evolution and policies, web-interface components.
es	13/11/11	Improvement on system verification
eb	13/11/11	Improved database schema + improved logical view

---

Legend: r = reviewed

# Contents

<b>Contents</b>	<b>I</b>
<b>List of Figures</b>	<b>III</b>
<b>List of Tables</b>	<b>IV</b>
<b>List of Listings</b>	<b>V</b>
<b>Glossary</b>	<b>VI</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 System context</b>	<b>1</b>
2.1 Context . . . . .	1
2.2 Home Power Save . . . . .	1
<b>3 Business Information</b>	<b>2</b>
3.1 Business Vision . . . . .	2
3.2 Business Rationale . . . . .	4
3.3 Product Description . . . . .	4
3.4 Target Audience . . . . .	5
3.5 System Evolution . . . . .	5
3.6 Business and Domain Model . . . . .	6
3.7 Roadmap . . . . .	7
3.8 Financial Model . . . . .	7
3.8.1 Software Architecture - consultancy costs . . . . .	8
3.8.2 Development costs . . . . .	8
3.8.3 Hardware costs . . . . .	8
3.8.4 Product costs . . . . .	8
3.8.5 End-user profit . . . . .	9
3.9 Competitors . . . . .	9
3.9.1 Black & Decker Power Monitor . . . . .	9
3.9.2 Black & Decker Power Cost Monitor WiFi Edition . . . . .	9
3.9.3 Plugwise Home Start . . . . .	10
<b>4 Requirements</b>	<b>11</b>
4.1 Architectural Vision . . . . .	11
4.2 Stakeholders . . . . .	12
4.3 Key Drivers . . . . .	13
4.4 High-Level Requirements . . . . .	14
4.5 Stories and Use-Cases . . . . .	16
4.5.1 Description . . . . .	16
4.6 Functional Requirements . . . . .	22
4.7 Technical Requirements . . . . .	23
4.8 Commercial Non-Functional Requirements . . . . .	24
4.9 Technical non-functional Requirements . . . . .	24
4.9.1 Reliability . . . . .	24
4.9.2 Usability . . . . .	25
4.9.3 Adaptability . . . . .	25
4.9.4 Recoverability . . . . .	25

4.9.5	Security . . . . .	26
4.9.6	Compliance . . . . .	26
4.10	Risk Assessment . . . . .	26
<b>5</b>	<b>Analysis</b>	<b>27</b>
5.1	Assumptions . . . . .	27
5.2	Technology Roadmaps . . . . .	27
5.3	High-level Design Decisions . . . . .	29
<b>6</b>	<b>System Architecture</b>	<b>35</b>
6.1	System context . . . . .	35
6.1.1	Diagram . . . . .	35
6.1.2	Users and Roles . . . . .	36
6.1.3	External Systems . . . . .	36
6.1.4	Channels and Information Flows . . . . .	37
6.1.5	Alternatives . . . . .	38
6.2	Verification . . . . .	39
6.2.1	Availability . . . . .	39
6.2.2	Time to Market . . . . .	39
6.2.3	Update . . . . .	40
6.2.4	Device Communication . . . . .	40
6.2.5	Cost and Return of Investment . . . . .	40
<b>7</b>	<b>Hardware architecture</b>	<b>41</b>
7.1	Hardware Design Decisions . . . . .	41
7.2	Hardware Overview . . . . .	46
7.3	Main Device . . . . .	46
7.3.1	Specifications . . . . .	46
7.3.2	Storage . . . . .	46
7.3.3	Hardware Block Diagram . . . . .	47
7.4	Socket Meter . . . . .	47
7.4.1	Specifications . . . . .	47
7.4.2	Storage . . . . .	48
7.5	Server . . . . .	49
7.5.1	Specifications . . . . .	49
7.5.2	Service Level Agreement . . . . .	51
<b>8</b>	<b>Software architecture</b>	<b>52</b>
8.1	Architectural Significant Drivers . . . . .	52
8.2	Logical View . . . . .	52
8.2.1	Primary Presentation . . . . .	52
8.2.2	Element Catalog . . . . .	54
8.2.3	Variability Guide . . . . .	61
8.3	Data View . . . . .	62
8.3.1	Events . . . . .	62
8.3.2	Database Schema . . . . .	62
8.4	Process View . . . . .	64
8.4.1	Monitoring diagram . . . . .	64
8.4.2	Element Catalog . . . . .	66
8.4.3	Policies . . . . .	68
8.5	Deployment View . . . . .	69

8.5.1	Deployment Diagram . . . . .	69
8.5.2	Artifacts . . . . .	69
8.6	Software Design Decisions . . . . .	71
<b>9</b>	<b>Architecture Verification</b>	<b>75</b>
9.1	Requirements verification . . . . .	75
9.1.1	Functional Requirements . . . . .	75
9.1.2	Technical Requirements . . . . .	77
9.1.3	Commercial Requirements . . . . .	78
9.1.4	Technical Non-Functional Requirements . . . . .	79
9.2	Architecture Evaluation . . . . .	81
9.2.1	Architectural Approaches . . . . .	81
9.2.2	Utility Tree . . . . .	82
9.2.3	Low-Level Scenarios . . . . .	83
<b>10</b>	<b>System evolution</b>	<b>85</b>
10.1	Mobile Devices . . . . .	85
10.2	Expansion and Localization . . . . .	85
10.3	Business Device . . . . .	85
10.4	Gas and Water . . . . .	85
10.5	Automated Control . . . . .	86
10.6	Integrated Device . . . . .	86
10.7	Energy Storage and Production . . . . .	86
10.8	HPS versions . . . . .	86
	<b>References</b>	<b>87</b>
	<b>Appendix</b>	<b>91</b>
	<b>A1 Example: Executive report</b>	<b>91</b>
	<b>A2 Decision</b>	<b>92</b>
A2.1	Arguments . . . . .	92
A2.2	Deployment View . . . . .	95
A2.3	Chronological View . . . . .	97
A2.4	Relation View . . . . .	98
	<b>A3 MessageBus Events</b>	<b>99</b>
A3.1	Monitoring . . . . .	99
A3.2	Pricing . . . . .	99
A3.3	Budgeting . . . . .	100
	<b>A4 Risk Assessment</b>	<b>100</b>
	<b>A5 Service Level Agreement</b>	<b>104</b>
	<b>A6 Mock-up Prototype</b>	<b>105</b>
	<b>A7 Time Tracking</b>	<b>106</b>

## List of Figures

1	Business Model . . . . .	6
2	Domain Model . . . . .	7
3	HPS deployed in a two-story house. . . . .	11
4	Operation modes of the HPS . . . . .	15
5	Use case diagram . . . . .	16
6	Technology roadmap of the HPS device. . . . .	27
7	Technology roadmap of the Smart Grid based on [42]. . . . .	28
8	System context diagram . . . . .	35
9	Hardware overview. . . . .	46
10	Hardware block diagram of the GuruPlug Server. . . . .	48
11	The HPS Socket Meter, based on the Plugwise Circle [35]. . . . .	48
12	Layered-decomposition of the Software . . . . .	53
13	System Decomposition . . . . .	53
14	Device Management . . . . .	55
15	Update component . . . . .	58
16	Update component . . . . .	59
17	ESH Update Server component . . . . .	60
18	HPS Database . . . . .	62
19	ESH Database . . . . .	63
20	Activity diagram monitoring process . . . . .	65
21	Process Device Communication . . . . .	66
22	Publish messages via MessageBus . . . . .	67
23	Responding to messages . . . . .	67
24	Policy response to a budget trigger. . . . .	68
25	Quality attribute utility tree . . . . .	83
26	Webgrid for decision 13 . . . . .	92
27	Webgrid for decision 14 . . . . .	92
28	Webgrid for decision 15 . . . . .	92
29	Webgrid for decision 16 . . . . .	93
30	Webgrid for decision 17 . . . . .	93
31	Webgrid for decision 18 . . . . .	93
32	Webgrid for decision 20 . . . . .	93
33	Webgrid for decision 21 . . . . .	94
34	Webgrid for decision 22 . . . . .	94
35	Webgrid for decision 23 . . . . .	94
36	Webgrid for decision 24 . . . . .	94
37	Deployment View . . . . .	96
38	Chronological View . . . . .	97
39	Relationship View . . . . .	98
40	Login . . . . .	105
41	Dashboard / Startpage . . . . .	105
42	Context-sensitive help . . . . .	105
43	System overview . . . . .	105
44	Enable/Disable devices . . . . .	105
45	Add device . . . . .	105



## List of Tables

2	SWOT analysis . . . . .	3
3	Development costs . . . . .	8
4	Matrix of stakeholder concerns . . . . .	12
5	UC-1.1 - Configure a Device . . . . .	16
6	UC-1.2 - Monitor a Device . . . . .	17
7	UC-2.1 - Request Prices . . . . .	18
8	UC-3.1 - Generate Report . . . . .	19
9	UC-4.1 - Create Policy . . . . .	19
10	UC-4.2 - Execute Policy . . . . .	20
11	UC-6.1 - Update System . . . . .	21
12	Risk severity based on the impact and likelihood of a risk. . . . .	26
13	Decision - Local Unit . . . . .	29
14	Decision - ZigBee Smart Energy v2 . . . . .	30
15	Decision - Linux . . . . .	31
16	Decision - Local Data Storage . . . . .	32
17	Decision - Web-interface . . . . .	33
18	Decision - Push Updates . . . . .	34
19	System Architecture Availability. . . . .	39
20	Decision - GuruPlug Server . . . . .	41
21	Decision - SD card . . . . .	41
22	Decision - Plugwise Circle Socket Meter . . . . .	42
23	Decision - Outsourced update server . . . . .	43
24	Decision - Debian - Linux Distribution . . . . .	45
25	Specifications of the main HPS device. . . . .	47
26	Technical specifications of the Socket Meter. . . . .	49
27	Artifact descriptions for the HPS device. . . . .	70
28	Artifact descriptions for the ESH server. . . . .	71
29	Decision - SQLite . . . . .	71
30	Decision - Message Bus . . . . .	71
31	Decision - Web-server . . . . .	72
32	Decision - Programming Language . . . . .	73
33	Decision - Firewall . . . . .	73
34	Functional requirements verification. . . . .	77
35	Technical requirements verification. . . . .	78
36	Commercial non-functional requirements verification. . . . .	79
37	Technical non-functional requirements verification. . . . .	81
38	Events of the Monitoring component . . . . .	99
39	Events of the Pricing component . . . . .	99
40	Events of the Budgeting component . . . . .	100
41	Service Level Agreement requirements for the ESH server. . . . .	104

## Listings

1	Interface MessageDispatch . . . . .	54
2	Interface MessageListener . . . . .	54
3	Interface ITableGateway . . . . .	55
4	Interface DeviceManagement . . . . .	55
5	Interface DeviceLookup . . . . .	56

6	ESH Update Server interface . . . . .	59
7	Push receiver interface . . . . .	59
8	ISystemFacade interface . . . . .	59
9	Interface Register . . . . .	60
10	Protocol Event . . . . .	62

# Glossary

## **ESH**

European Smart Homes, the company which has commissioned the architecture for HPS. 2, 3, 6

## **ESH update server**

A internet server that deploys new updates for the HPS devices. 4

## **HAN**

Home Area Network, enable users to remotely connect to and control automated digital devices in a house. 5, 10

## **HAS**

Home Automation System. 3, 5

## **HPS**

HomePowerSave, the product that will be developed. 2–10

## **Prosumer**

An customer that not only consumes energy but also produces it. 2

## **Realtime**

Measure the power usage and production at least every 8 seconds. 7

## **Realtime Pricing**

The Utility announces hourly electricity prices through the SmartMeter. This allows the consumer to adopt the behavior in realtime. 2, 3, 7

## **Smart Device**

A device that can be controlled by a protocol. Controlled means that it can be started/stopped/paused or dimmed (reduced energy consumption). 2–4

## **Smart Grid**

A system which includes a variety of operational and energy measures, including smart meters, smart appliances, renewable energy resources, and energy efficiency resources [16]. 2, 3, 5

## **Smart Meter**

A electrical meter that records consumption of electric energy and reports that to the Utility on a regular basis. 2–6, 8

## **Socket Meter**

A small meter that is plugged between the power socket and the device. The Socket Meter send the energy consumption to the HPS via ZigBee. (cf. Smart Device) 2–4, 6, 7

## **TimeTariffInterval**

Contains the price and costs of electricity for one or more periods of time. 4

## **Utility**

Operates and maintains the grid. 3

# 1 Introduction

We are an independent software architecture consultancy, being commissioned by European Smart Homes (ESH) to elaborate a system architecture for their new product HomePowerSave. This documents describes the architecture of HomePowerSave.

## 2 System context

In this section the system context is outlined. This section starts with an introduction of the company. Subsequently, the context of the system is outlined. This section ends with a description about the new product.

### 2.1 Context

Since the beginning of the 21 century the energy market is seeing some significant changes. From a passive consumer market people become more and more aware of the environment and their energy consumption. At the same time the energy market itself is seeing the birth of the Prosumer, where consumer they not only buy energy but also produce and sell energy. The main result of these changes is the birth of the Smart Grid.

**Smart Grid definition** - Smart grid is a type of electrical grid which attempts to predict and intelligently respond to the behavior and actions of all electric power users connected to it - suppliers, consumers and those that do both – in order to efficiently deliver reliable, economic, and sustainable electricity services. [53]

The Smart Grid allows people to be more aware and gain better understanding of the energy they use. This allows the consumer to take control of their energy consumption.

### 2.2 Home Power Save

With the emergence of the Smart Grid in Europe, European Smart Homes (ESH) is extending their offered systems with a new product, called HomePowerSave (HPS), which optimizes energy efficiency.

HPS is a device, which can interact with a Smart Meter, Smart Devices in order to optimize the energy management of houses. For instance, it reads data about energy consumption from the Smart Meter, it starts or stops ZigBee enabled Smart Devices and it tracks the energy produced by the energy generators.

On the one hand, optimizing the energy management involves that the system uses the Realtime Pricing feature to reduce the energy bill. On the other hand, and optimized energy management would focus on reducing the energy consumption by the house in total (cf. 3.3).

With the HPS the end-user has insights into the actual energy consumption and production of the house<sup>1</sup>. Moreover, the end-user has the possibility to view a detailed real-time estimation of the upcoming energy bill, reflecting the monetary impact of certain devices on the energy bill<sup>2</sup>.

---

<sup>1</sup>The HPS will periodically gather information from compatible devices about their consumption and generate a report.

<sup>2</sup>Additional hardware and configuration is required per device that is not ZigBee enabled. e.g. a Socket Meter that is plugged between the power socket and the actual device.

### 3 Business Information

The following section emphasizes on describing the business environment of ESH and the HPS. The architectural relevant business information is also illustrated.

At first the business vision and rationale are presented, including the unique selling point of the HPS as well as a SWOT-analysis. Subsequently, the main functionalities of the HPS are described with an emphasize on providing an high-level overview of the product being developed. Afterwards, the evolution and roadmap of the HPS are presented.

The section finishes with an illustration of the financial model and an identification of potential competitors.

#### 3.1 Business Vision

The Smart Grid will change the way energy will be consumed, produced and even the way its paid. The ability of getting paid for disposing unused energy into to the electrical grid and the possibility to quickly adapt energy consumption to price changes in the market, is a real opportunity for each house owner.

However, new devices are required to use the new functionality of the Smart Grid and bring the advantages to the house owner, like the HomePowerSave.

ESH aims to provide a system that has three main goals:

- Reduce the end-user's energy bill
- Provide the end-user with detailed information about his energy consumption and production
- A real-time view on the monthly energy bill

The HPS project and accordingly ESH, have been assessed based on internal factors like strengths and weaknesses but also on external factors, which have been grouped into opportunities and threads.

The result of this SWOT analysis is illustrated in Table 2.

Strengths	Weaknesses
<ul style="list-style-type: none"> <li>• ESH has experience with HAS.</li> <li>• Real time billing is a upcoming hot-topic.</li> <li>• Reduces the end-user's energy bill, this makes marketing easier.</li> </ul>	<ul style="list-style-type: none"> <li>• Low-experience in the electricity market.</li> <li>• Hardware is not produced, instead HPS relies on COTS<sup>3</sup> devices.</li> <li>• Limited possibility to connect to heterogeneous Smart Devices. Most devices or HAS use an own or derived protocol, that require additional implementation effort.</li> <li>• Requires technical aware end-users.</li> </ul>
Opportunities	Threats
<ul style="list-style-type: none"> <li>• The Smart Grid is expanding and is already implemented in certain areas.</li> <li>• Not many competitors (cf. 3.9).</li> <li>• Green technology is booming.</li> <li>• Attracts environmental-aware customers.</li> </ul>	<ul style="list-style-type: none"> <li>• Competitors could enter the market sooner.</li> <li>• Competitors could offer cheaper products.</li> <li>• Interface incompatibilities. The HPS relies on external interfaces/devices that evolve and change. HPS will become incompatible if it is not adopted.</li> </ul>

Table 2: SWOT analysis

The unique selling point of our system can be summarized as follows:

1. The possibility to take advantage of Realtime Pricing by **policy-based device controlling**. The system lets the end-user decide how it should behave based on policies.
2. Getting **overview of the energy consumption** and the **price paid for electricity** at any moment.
3. A **verbose system** that informs the user about events, like price-changes, unusual high energy consumption or if the bill limit is exceeded.

This is not the only advantage compared to competing systems. In order to convince the user of the technology, the product will also be:

1. User-friendly, e.g. the web-interface should be clear and easy to understand.
2. Reasonably-priced.

---

<sup>3</sup>Commercial of the shelves

### 3.2 Business Rationale

European Smart Homes will create a new system for the ever changing needs of home energy aligning. With the emergence of Smart Grids in Europe, new opportunities are created to revolutionize home energy aligning, optimizing the energy consumption, reducing energy costs and improving upon the user-friendliness of the system.

ESH is determined to open a new market for this system and to extend their current market share in home automation systems.

ESH is confident that their relatively feasible technical solution aligns with the business vision. Furthermore, ESH is confident that there is a need and potential for this system. By offering the system at an attractive price and developing an user-friendly system, the system will be attractive to customers. An increasing need for these systems, combined with the increasing national and international coverage of the Smart Grid, makes the market very likely to continue to grow for a long enough period to not only cover the return on investment period, but also being able to make a profit.

### 3.3 Product Description

ESH offers its customers a system called HPS. The HPS is a system that offers customers a way to save money on the usage of electricity. The HPS will request the current electricity prices from the SmartMeter, utilizing the Realtime Pricing-feature of the Smart Grid.

The HPS physically consists of two types of hardware. The first type are ZigBee enabled devices. These can be either Smart Devices, like Washing Machine or simple electricity meters which can be plugged into a power socket. The later device will be called Socket Meter.

The Socket Meter can start or stop the electricity going through it. This way the attached devices(e.g. a lamp) can be turned on or off. The Socket Meter tracks the energy consumption off the attached devices.

The second device is the main processing unit (from now on called HPS). The HPS is a single device. It tracks the energy consumption of the Socket Meters and Smart Devices. This way it can report the energy consumption per device in contrast to the overall consumption of the house. It has a detailed usage analysis for Socket Meter enabled devices. The reports can be read by the user using a web-interface or by receiving an e-mail. The HPS gives advises on the user's energy consumption, e.g. it tries to identifies devices that have an unusual high energy consumption compared to other days.

The web-interface shows a report of the consumed electricity (cf. Appendix A1). It outlines energy consumption of individual devices.

The HPS can interoperate with home automations devices so called Smart Devices, which are supported by the protocol "ZigBee: Smart Energy" or "ZigBee: Home Automation". The devices that are part of the home automation systems will be controlled. The home automation system itself will not be controlled. Because of the heterogenous nature of the Smart Devices, each requires an own vendor specific driver. HPS will be shipped with a core set of drivers, for example drivers for Miele. New drivers will be developed and provided as update for existing users.

The HPS displays a real-time estimation of the daily,monthly and annual bill. The bill displays,

per Socket Meter or Smart Devices, the share of the device on the energy bill. The user can configure a price limit in total or for each device. HPS will inform the user in case the price exceeds the limit. The user will be informed via e-mail or the web-interface.

The HPS allows the user to define and configure policies. Policies define, how the HPS is allowed to control devices in order to save energy. For example, if the price of electricity is high, light can be dimmed, the heater reduced or the charging of the car can be postponed until the price is low

The HPS requires an internet connection to get updates. Data like energy consumption and bill information will not be exposed to the internet.

### **3.4 Target Audience**

The target customer of the HPS device has a high acceptance of new technologies and is willing to automate devices in their home. The target customer is a home owner who resides in The Netherlands. The HPS device is suitable for all ages and the target customer also wants to contribute to the reduction of the global climate change. The target customer is aware about the possibilities of a reduction of electricity costs and is willing to spend €300 for the HPS device.

### **3.5 System Evolution**

As soon as the HPS is widely accepted by users in the target market, the next step involves to emerge the HPS to new markets in Europe and the USA.

Furthermore, the goal is to develop an advanced product version of the HPS for the business sector. ESH have planned to deliver the product to businesses in Q1 2014.

Home automation systems that do not support ZigBee will be included. E.g. X10, C-Bus. This way more customers are able to buy and use the HPS.

Additionally, the number of supported devices will be constantly increased by releasing new drivers for Smart Devices.

Next to that the support of the HPS should extend to the controlling of energy generators and the ability to control energy storage products.



### 3.6 Business and Domain Model

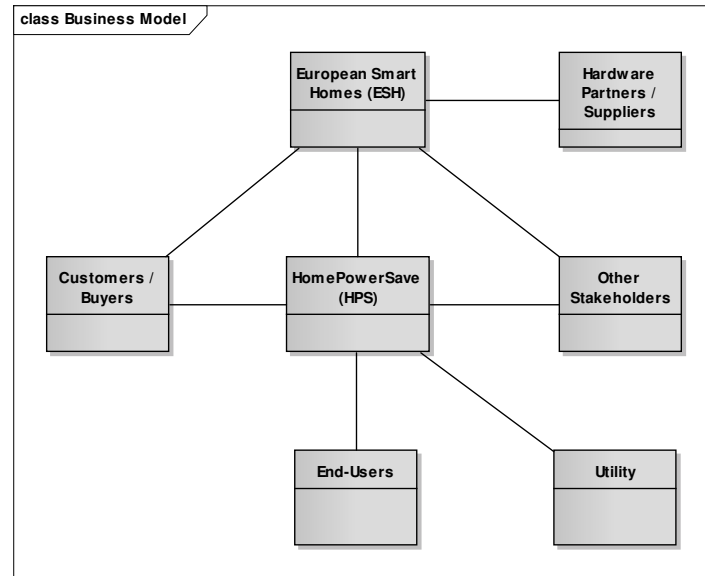


Figure 1: Business Model

The business model, as illustrated in Figure 1, names the main entities that play a role in the environment of the HPS. A line between two entities represent sthat there is some kind of a relation between them.

ESH has relations with several hardware partners/suppliers to manufacture the product. It also has relations with all the other stakeholders (cf. 4.2) that have concerns about the HPS that ESH develops.

HPS has a relation with the customer that buys (and most likely uses) the HPS and also with the end-user who is eventually using it. HPS has an indirect connection with the Utility company through the Smart Meter. The utility company provides the latest energy prices through the Smart Meter.

In Figure 2 is the domain model depicted of the HPS is modelled in UML. It consists of the main entities/components that are connected in some way to the HPS system. The interfaces (protocol/transport) that connects two components are outlined next to the edges for convenience.

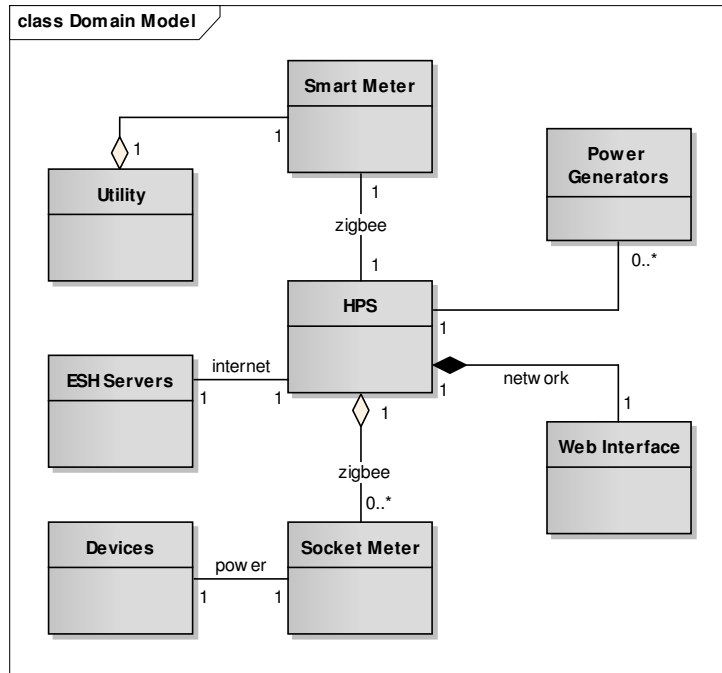


Figure 2: Domain Model

### 3.7 Roadmap

ESH want to launch the product as soon as possible. The first quarter of 2013 is desired. It is estimated that the development time of such a mature system, with all the components/interfaces/protocols and the software, takes approximately one year.

In West-Europe as well as in North-America several regions already switched to the Smart Grid, hence ESH can start in these countries. However, as a first step it is decided to start in The Netherlands, ESH is already well-known brand and has working distribution system in The Netherlands.

The roadmap anticipates to start delivering the product first to home owners (households). After a successful market launch it is planned to extend the target to the business sector. ESH has planned to deliver the product to businesses in Q1 2014.

#### Overview

Period	Area	Description
2013 Q1	Netherlands	Launch HPS for households
2014 Q1	Netherlands	Launch HPS for business
2015 Q1	Europe	Launch HPS for households
2015 Q4	Europe	Launch HPS for business

### 3.8 Financial Model

In this chapter the financial model are discussed. This chapter explains in depth about the salary, development costs, hardware costs, production costs and end-user profit.

Table 3: Development costs

Description	man hours
Get electric usage from Smart Meter	160
Firmware update framework	120
Get prices from SmartMeter	200
Communication with ZigBee devices	2560
Web-interface (front-end)	1280
Configure GuruPlug	1920
Connection to database & database configure	320
Testing & debugging	480
Release build	160
Overhead & Hours worked on other not mentioned/forgotten parts	960
<b>Total hours</b>	<b>8160</b>
<b>Total cost</b>	<b>€408.000,00</b>

### 3.8.1 Software Architecture - consultancy costs

Every member of the team will cost €100 per hour. The software architecture team exist of six members. Each member will spend 10 weeks for research and making the software architecture document. Each member will spend 16 hours per week. The total cost of making the software architecture document is:

$$\text{€}100,00 * 16 * 10 * 6 = \text{€}96.000,00$$

### 3.8.2 Development costs

The costs for developing the whole system will be covered with the profit from other divisions of ESH. ESH already has customers with home automation systems. Hence, ESH can offer these customers the HPS system.

The development of the system requires a team working full-time on the project. The team-size shall not be lower than four persons. The time needed to make the product ready for release will take 8160 hours. The development team will work 40 hours a week on the project. A team member has a pay check of €50,- per hour. According to this, the total development costs will be **€408.000,00**. The details are illustrated in Table 3.

### 3.8.3 Hardware costs

The hardware needed for the complete HPS system are a GuruPlug server plus [22], a SD-card of 4 GB and two Socket Meters. The price for the GuruPlug server plus is €94,92 per device. The 4GB SD-card costs €10,-. The costs of a Socket Meter are €36.65 [34].

The total costs for the complete HPS system is:

$$\text{€}94,92 + \text{€}10 + (2 * \text{€}36.65) = \text{€}178,22$$

### 3.8.4 Product costs

The product includes the development costs per device, the costs for making the architecture document per device, marketing costs €15,00, distribution €5,00, packaging €10,00, overhead €35,00 and maintenance/guarantee are an additional €30,00. The total costs per device are shown in the figure below:

Number of units	Development costs per device	Architecture document	Total costs per product
10.000	6.40	19.2	303.82
20.000	3.20	9.6	291.02
30.000	1.07	3.2	282.49
50.000	0.64	1.92	280.78
100.000	0.32	0.96	279.50

### 3.8.5 End-user profit

In order to calculate the end-user profit, a sample household has to be chosen. In this case, a family household consisting of four persons . An average usage of such a household is 4800KwH [14].

The hardware consumption is 5V/3A (max), giving us a consumption of 15 watts equaling 131,4 kWh a year. The end-user can expect the system, if used properly, to save 10% on the total energy consumption. This will return a minimum gain of **480 kWh - 131,4 kWh = 384,60 kWh**.

Since the average price per KwH, at the moment of writing, is around €0,25 the savings per year will be €96,10. The consumer will break-even in around three years, since the energy prices are rising and the energy savings could be higher.

## 3.9 Competitors

There are several competitors to the HPS device, each with slightly different unique selling points and specifications.

### 3.9.1 Black & Decker Power Monitor

The Black & Decker Power Monitor [45] comes in two parts, one part that attaches to your meter and the other part that goes somewhere in your house. The monitor itself reads your meter and transmits the information to the base inside of your house. The features of this model are:

- Displays electricity use in your home, minute by minute, in dollars or kW.
- Studies show families that use the Power Monitor to change their behavior and save energy can reduce their electricity bill by up to 20%.
- Easy to install weatherproof sensor requires no wiring and works with almost all electricity meters.
- Shows month-to-date bill and predicted monthly bill, helping households stick to their energy budgets.
- Also shows day, time, and outside temperature (wherever the sensor is placed).

The price of this device is \$99.99.

### 3.9.2 Black & Decker Power Cost Monitor WiFi Edition

This power monitor [52] comes in two parts, just like the Black & Decker power monitor. The advantage of the monitor is that it can be connected to the internet. This way your data is accessible through the internet.

The price of this device is \$191.00.

### 3.9.3 Plugwise Home Start

The Plugwise Home Start system will bring your energy consumption under control. It offers detailed insight into your energy consumption, enabling you to intelligently switch your devices on or off. The features of this model are:

- *Circle's* for legacy devices (e.g. fridge, washing machine, television), which measures the energy consumption or production of the connected device, and is positioned between the plug of the device and the power socket.
- The software, called *Source*, can calculate your totals over time and show them as Cost, Energy or CO<sub>2</sub> savings.
- The *Circle* can switch appliances according to switching schemes that you customize in *Source*.
- The *Circle* is also equipped with a standby killer: when a connected appliance switches to standby mode, the Circle can entirely interrupt power to prevent unwanted energy usage.

The price of this device including two *Circles* is €99.95.

## 4 Requirements

In this section the requirements of the HPS are outlined. It starts with the architectural vision of the HPS, followed by the stakeholders and their concerns. Subsequently, the key drivers are presented, which are derived from the concerns of the stakeholders. The section ends with use-cases and requirements, followed by a risk assessment.

### 4.1 Architectural Vision

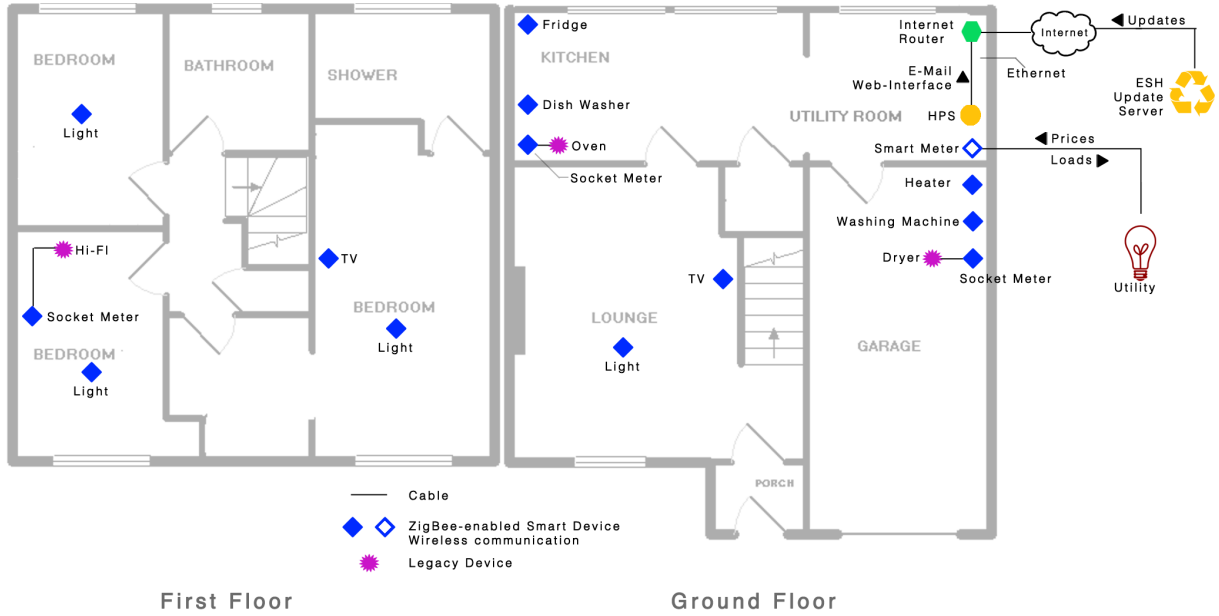


Figure 3: HPS deployed in a two-story house.

The environment of the HPS (circle) and the entities the HPS interacts with are illustrated in Figure 3.

The HPS is the central and main device in the environment. Usually, it will be placed in the utility room next to the Smart Meter (bordered triangle) and the Internet Router (hexagon).

The HPS is connected to the Internet Router via Ethernet to get access to the Internet. Internet access is necessary in order to receive Software-Updates from the ESH Update server (arrow circle) and send E-Mails to the end-user. The strength of HPS is the possibility to quickly adapt changes in the field of Smart Energy, hence it is crucial that it provides an elaborated upgrade mechanism.

The current electricity prices are wirelessly requested from the Smart Meter via ZigBee. The Smart Meter itself receives the price updates directly from the Utility, most likely via digital power line communication. However, this is the concern of the utility and not of the HPS.

The HPS has the ability to communicate with every device in the household that supports the wireless ZigBee protocol (filled triangle). The type of communication depends on the type of the device. Some devices provide the ability for interaction e.g. a light provides the functionality to switch it on/off or dimm it. However, all Smart Devices shall support the possibility to report their load consumption to the HPS. Those devices that do not support ZigBee can be integrated into the HPS by special Smart Devices, so called Socket Meters. For instance, as shown in Figure 3, the Oven in the Kitchen, which is a device that does not support ZigBee but can be still

monitored and controlled via a Socket Meter. Socket Meters are adapters between the Power Socket and the Power Cable of the device. The Socket Meter report the load consumption to the HPS, as every other ZigBee device does.

The HPS uses the load information from the Smart Devices and the Smart Meter and combines them with the Pricing information from the Smart Meter to generate estimations of the electricity bill and to generate reports of the overall energy consumption of the house. The end-user of HPS is able to access and to configure the HPS via a web-interface.

## 4.2 Stakeholders

**Product owner (ESH)** is concerned about the profitability, affordability, reliability and usability of HPS. He provides the budget and invest in the development and production of HPS. He also affords the marketing of HPS and is concerned about the cost-benefit impact of HPS. The product-owner needs a useful and reliable system in order to compete in the market and to draw profit.

**Customers** are concerned about usability, profitability, reliability and adaptability. They want a useful system that is reliable and also adaptable to changing environments, like new devices. They do not want to buy a new system, if they install a new Smart Device.

**End users** are concerned about product usability, reliability and adaptability. They benefit from reduced electricity bills, since the HPS aims to reduce energy consumption. They want useful product which is also reliable and adaptable towards the changing of tools and environments.

**Developer** are concerned about usability, testability, adaptability and maintainability. They develop the system, which includes analysis, design, implementation and testing of the HPS system.

In Table 4 the stakeholder concern matrix is illustrated<sup>4</sup>.

Table 4: Matrix of stakeholder concerns

(By priority) Stakeholder	Concerns						
	Usability	Reliability	Adaptability	Profitability	Affordability	Maintainability	Testability
Product owner / ESH	10	30		30	30		
Customer	20	30	30	20			
End user	50	30	20				
Developer	10		30			30	30
Total	90	90	80	50	30	30	30

<sup>4</sup>Each stakeholder is given 100 points in total that need to be distributed among the concerns.

### 4.3 Key Drivers

The following key-drivers are deduced from the list of stakeholders and their concerns (cf. Table 4). They reflect the most important concerns of the highest prioritized stakeholders.

**Usability** The usability of the HPS is an unique selling point as it will give an advantage over the competitors. Usability improves customer loyalty, because switching to another product with a lower usability is unlikely. If the product is not usable, less customers are attracted.

**Reliability** Reliability is very important for the HPS , since wrong decisions/suggestions/calculations will be made when the HPS is unreliable. Selling an unreliable system will cost ESH a lot of money for maintenance, support. Additionally, if the HPS is unreliable, the customer will lose confidence in the product and probably buy a competitor's product.

**Adaptability** Adaptability is an important driver for HPS as the environment of the energy market is rapidly changing. For example new protocols and standards emerge rapidly. In the future ESH also plans to emerge to new markets and to support a wider range of standards, which requires the HPS to be adaptable.



## 4.4 High-Level Requirements

The following requirements represent the system's high-level functionality. Each high-level requirement will be refined in detail by various specific and technical requirements.

### Monitoring

- |             |             |  |
|-------------|-------------|--|
| <b>HL-1</b> | <b>Must</b> | <p>The system is able to monitor the energy consumption of Socket Meter or ZigBee-enabled Smart Devices. It monitors the devices by periodically requesting the energy consumption. Additionally, it receives the total energy consumption from the Smart Meter.</p> <p>The data will be stored persistently in a way that it can be processed later to generate reports or calculate the bill.</p> <p>The system has a list of devices that should be monitored. This list can be configured by the user in the user-interface.</p> <p>The user should have the possibility to export the raw load measurements and billing information. The exported data can be used to create own statistics or reports.</p> |
|-------------|-------------|--|

### Billing

- |             |             |   |
|-------------|-------------|---|
| <b>HL-2</b> | <b>Must</b> | <p>The system is able to estimate the bill using the realtime pricing feature and the monitoring information (cf. HL-1).</p> <p>The billing system fetches the energy prices from the internet and stores them persistently, so that can be associated with the amount of energy consumed during a certain period of time. In addition, this data can be represented in the report. The user has the ability to see the upcoming monthly bill and the previous bills in the web-interface. The bill displays the amount of energy used and produced and the share of each device on the bill.</p> |
|-------------|-------------|---|

### Reporting

- |             |             |   |
|-------------|-------------|---|
| <b>HL-3</b> | <b>Must</b> | <p>The system should generate and display reports. Reports can include tables, charts, graphs and textual data. The report may be generated on request.</p> <p>The system shall be able to generate different kinds of report like summaries of the daily energy consumption and costs or detailed view on the energy consumption and costs of a particular device over a period of time.</p> <p>The user can select the type of report in the user-interface. The system should be able to automatically e-mail reports to the user.</p> |
|-------------|-------------|---|

### Policies

- |             |             |   |
|-------------|-------------|---|
| <b>HL-4</b> | <b>Must</b> | <p>The user can configure policies to let the HPS control ZigBee devices. A policy contains a trigger and actions. The trigger specifies the conditions that have to be matched in order to execute the actions of a policy. The policy should contain at least the following triggers: Electricity price based, Time-based and Budget-based. The action part of a policy define device-specific control-operations like start or stop or increase/decrease the energy consumption. The policies can be edited using the web-interface of the HPS. The option to configure policies allows the user to influence the way the system operates.</p> |
|-------------|-------------|---|

## Web-interface

**HL-5**     **Must**     HPS should provide a web-interface that allows the user to configure, operate and control the HPS. The web-interface shall be secured by a login.

## Update

**HL-6**     **Must**     Since the system is expected to grow in functionality and supported devices, it shall provide the possibility to be updated. The update process should be done automatically.

It is important that the device remains operational after the update.  
So the update is not allow to fail at any circumstances.

## Operations modes

**HL-7**     **Optional**     The software should provide two modes of operation: normal and maintenance mode. The maintenance mode is needed to give the maintenance personnel a low-level access to the system.

During maintenance mode the access to the network shall be blocked, it shall be possible to access the device via serial connection.

The maintenance mode shall be triggered in different ways. It shall be possible to enter maintenance mode programmatically during updates, via web-interface and it may be possible to enter maintenance mode due to an external event e.g. Button-pressed or USB-Key connected.

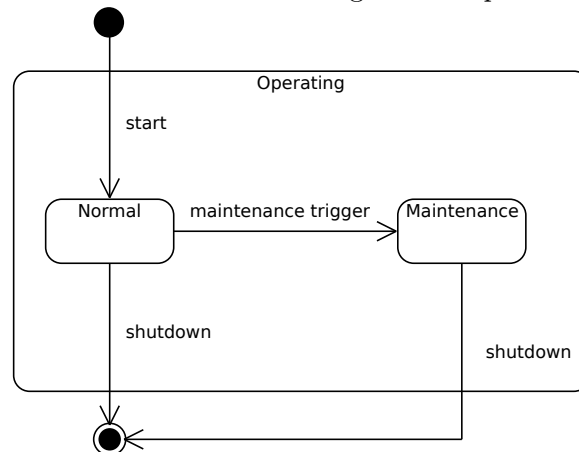


Figure 4: Operation modes of the HPS

## 4.5 Stories and Use-Cases

In Figure 5 is the use-case diagram illustrated, which provides an overview of the most important use-cases and their actors. The first number of each use-case refers to the corresponding high-level requirement.

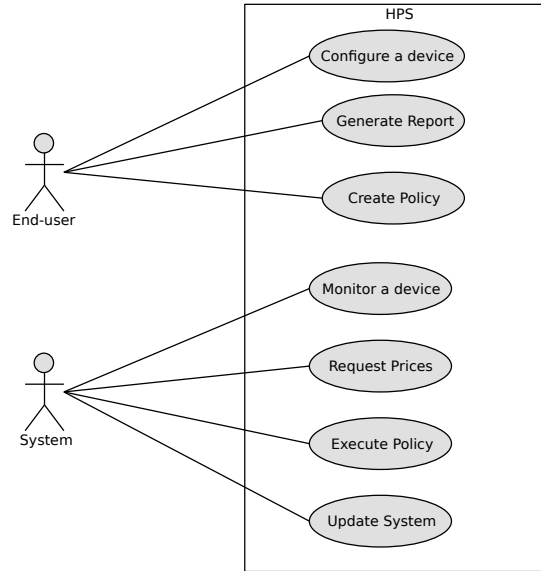


Figure 5: Use case diagram

### 4.5.1 Description

The following sections shows the architectural significant use cases. The numbering of the uses refers to the High-level requirement e.g. UC-6.x refers to HL-6.

Table 5: UC-1.1 - Configure a Device

#### UC-1.1 - Configure a Device

<b>Primary actor</b>	End-user
<b>Goal</b>	Install a new Socket Meter and start monitoring its energy consumption.
<b>Preconditions</b>	
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1. The user plugs the Socket Meter into the power socket.</li> <li>2. The user plugs the device into the Socket Meter</li> <li>3. The user opens the web-interface</li> <li>4. The user navigates to <i>new devices</i></li> <li>5. The user see a list of devices that were discovered by HPS but not associated yet.</li> <li>6. The user selects the device that he wants to install and selects add.</li> <li>7. The user saves the new settings</li> </ol>

<b>Extensions</b>	
<b>Postcondition</b>	
<b>Related Requirements</b>	HL-1, SR-1.1, SR-1.2, SR-1.4, HL-5

Table 6: UC-1.2 - Monitor a Device

<b>UC-1.2 - Monitor a Device</b>	
<b>Primary actor</b>	System
<b>Goal</b>	Install a new Socket Meter and start monitoring its energy consumption.
<b>Preconditions</b>	
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1. This use case starts when the user-interface displays the page for <i>new devices</i>.</li> <li>2. The system performs a ZigBee broadcast and lists unknown Socket Meter.</li> <li>3. The system is provided with the id of the devices that should be configured,</li> <li>4. The system persists the changes and starts to monitor the device</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>3.a The name and type could not be auto-discovered. <ol style="list-style-type: none"> <li>3.a.1 The system asks for a name and type. The flow continues with step 4.</li> </ol> </li> </ol>
<b>Postcondition</b>	
<b>Related Requirements</b>	HL-1, SR-1.1, SR-1.2, TR-1.1, TR-1.2, TR-1.3

Table 7: UC-2.1 - Request Prices

UC-2.1 - Request Prices	
<b>Primary actor</b>	System
<b>Goal</b>	Request energy prices from Smart Meter.
<b>Preconditions</b>	The HPS and the Smart Meter are part of the same HAN. The Smart-Meter is Smart Energy v2 compliant and implements the pricing function set. This use case is triggered, when the system receives an event to update energy prices or on its first run.
<b>Main success scenario</b>	
	<ol style="list-style-type: none"> <li>1. The Pricing-component requests the SmartMeter-Handle from the DeviceManager</li> <li>2. The Pricing component uses the handle to request the current TimeTariffInterval from the SmartMeter via ZigBee.</li> <li>3. The system processes the price and priceLevels and stores them for further processing.</li> <li>4. Depending on the future trend of the priceLevel, the Pricing component generates events to inform the Policy component.</li> </ol>
<b>Extensions</b>	
	<ol style="list-style-type: none"> <li>3.a No prices are returned <ol style="list-style-type: none"> <li>3.a.2 The system keeps track, for which timespan no prices are available and indicates these periods as errors so that they can be either excluded or marked as period with issues.</li> <li>3.a.3 The system tries to update these periods later. After 24hours these periods are marked as unresolvable issues. The use-case stops.</li> </ol> </li> <li>3.b The Smart Meter does not respond. <ol style="list-style-type: none"> <li>3.b.1 Inform the user about the error. The flow continues with 3.a.2</li> </ol> </li> <li>5.a The <i>priceLevel</i> is not supported by the Smart Meter. <ol style="list-style-type: none"> <li>5.a.1 The system uses heuristics to identify low-prices. The flow continues with Step 3.</li> </ol> </li> </ol>
<b>Postcondition</b>	The HPS uses the new prices for calculation.
<b>Related Requirements</b>	HL-2, SR2.1

Table 8: UC-3.1 - Generate Report

UC-3.1 - Generate Report	
<b>Primary actor</b>	End-user
<b>Goal</b>	The end-user wants to see the daily executive report.
<b>Preconditions</b>	The HPS has gathered data, for at least one day.
<b>Main success scenario</b>	
	<ol style="list-style-type: none"> <li>1. The user selects <i>reports</i></li> <li>2. The user selects the <i>daily executive report</i> from the list of available reports</li> <li>3. A loading-bar is displayed that indicates the progress of the report generation.</li> <li>4. The user can see the report in the browser and download it as PDF</li> </ol>
<b>Extensions</b>	
	<ol style="list-style-type: none"> <li>4.a The generation of the report fails. <ol style="list-style-type: none"> <li>4.a.1 A message is displayed that the report could not be generated. The error will be sent to ESH via E-mail. The use case stops.</li> </ol> </li> </ol>
<b>Postcondition</b>	The generated report is deleted.
<b>Related Requirements</b>	HL-3, SR-3.1, SR-3.2, SR-3.3, NF-2.7, NF-2.6

Table 9: UC-4.1 - Create Policy

UC-4.1 - Create Policy	
<b>Primary actor</b>	End-user
<b>Goal</b>	The end-user wants to create a policy to control the behavior of the washing-machine.
<b>Preconditions</b>	A washing-machine has been configured as device and is supported by HPS.
<b>Main success scenario</b>	
	<ol style="list-style-type: none"> <li>1. The user selects <i>new policy</i></li> <li>2. The user sets a price-based trigger for <i>low price</i></li> <li>3. The user defines the action <i>start washing program</i></li> <li>4. The user specifies that he will be notified by mail.</li> <li>5. The user saves the policy</li> </ol>

<b>Extensions</b>	
	<p>2.a The user wants that the washing-machine shall be finished at 10 pm at any case.</p> <p>2.a.1 The user specifies an exception for the trigger, that the policy shall be triggered before 10pm. The flow continues with Step 3.</p>
<b>Postcondition</b>	The policy will be executed if triggered
<b>Related Requirements</b>	HL-4, SR-4.1, SR-4.2, SR-4.3

Table 10: UC-4.2 - Execute Policy

<b>UC-4.2 - Execute Policy</b>	
<b>Primary actor</b>	System
<b>Goal</b>	The system triggers the policy of a washing-machine.
<b>Preconditions</b>	A policy has been defined that specifies that the washing machine shall run if the price is low (cf. UC-6.1).
<b>Main success scenario</b>	
	<ol style="list-style-type: none"> <li>1. The system checks if the price-based trigger of the washing-machine policy is fulfilled.</li> <li>2. The system checks that the washing-machine is ready-to-run and that the user has specified the washing-program.</li> <li>3. The system sends a start command to the washing machine. The use-case is finished.</li> </ol>
<b>Extensions</b>	
	<p>1.a The price-based trigger is not fulfilled.</p> <p>1.a.1 The System checks if the time threshold of the trigger is exceeded. (Washing machine should start 10pm latest) The use-case continues with Step 2.</p> <p>2.a The washing-machine is not ready or no program has been scheduled.</p> <p>2.a.1 The system skips the rest of the use-case. The policy will be executed again in the next iteration. The use-case stops.</p>
<b>Postcondition</b>	
<b>Related Requirements</b>	HL-4, SR-4.1, SR-4.2, SR-4.3

Table 11: UC-6.1 - Update System

<b>UC-6.1 - Update System</b>	
<b>Primary actor</b>	System
<b>Goal</b>	Self-update
<b>Preconditions</b>	HPS has a working Internet connection
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1. The system listens for incoming update notifications from the ESH update server.</li> <li>2. The system downloads the update from the ESH update server and performs a checksum</li> <li>3. The system creates a backup of the user data, configurations and the previous software version, in a way that the old version of the software can be easily restored. The backup is stored locally.</li> <li>4. The system enters maintenance mode and installs the update</li> <li>5. The system performs a self-check and informs the user of the successful update</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>1.a The system is not able to listen because of various reasons (e.g. firewall or UPnP incompatible router) <ol style="list-style-type: none"> <li>1.a.1 The system falls back to the pull-mechanism. It asks for new updates every hour. The flow continues at step 2.</li> </ol> </li> <li>4.a The update fails due to various reasons. <ol style="list-style-type: none"> <li>4.a.1 The System loads the backup.</li> <li>4.a.2 The System installs the previous version of HPS</li> <li>4.a.3 The System imports the user data and configuration files. The use case stops.</li> </ol> </li> <li>5.a The self-check fails <ol style="list-style-type: none"> <li>5.a.1 The System loads the backup.</li> <li>5.a.2 The System installs the previous version of HPS</li> <li>5.a.3 The System imports the user data and configuration files. The use case stops.</li> </ol> </li> </ol>
<b>Postcondition</b>	The system is in an consistent state.
<b>Related Requirements</b>	HL-6, TR-6.1, SR-6.1, SR-6.2, SR-6.4, SR-6.5



## 4.6 Functional Requirements

In this chapter a more detailed representation is given of the highlevel requirements. SR-1.\* reverse to HL-1. SR-2.\* reverse to HL-2 and so on.

Several functional requirements are based on global requirements. The next functional requirements are global requirements.

<b>SR-1.1</b>	<b>Must</b>	The system shall communicate with Socket Meters and Smart Devices.
<b>SR-1.2</b>	<b>Must</b>	The system shall monitor energy consumption of connected Socket Meters and Smart Devices periodically.
<b>SR-1.3</b>	<b>Must</b>	The system shall persist the monitored energy consumption per device.
<b>SR-1.4</b>	<b>Must</b>	The web-interface shall provide the possibility to add/edit/delete new devices.
<b>SR-1.5</b>	<b>Optional</b>	The web-interface shall provide a possibility to enable/disable monitoring of a particular device.
<b>SR-2.1</b>	<b>Must</b>	The system shall fetch electricity prices from the Smart Meter and persist it, so that it can be associated with the amount of energy used to calculate the bill.
<b>SR-2.2</b>	<b>Must</b>	The web-interface should display a real-time estimation of the electricity bill. This means that the user can see a live-view on the amount of money spend on electricity in the past 24 hours.
<b>SR-2.3</b>	<b>Must</b>	The system shall list the amount of energy consumed in total and per device on the energy bill. The prices will also be displayed on the bill.
<b>SR-2.4</b>	<b>Must</b>	The system shall calculate a daily, monthly and annual bill of the stored data
<b>SR-2.5</b>	<b>Optional</b>	The web-interface should be able to display energy bills of the previous month/days/years.
<b>SR-3.1</b>	<b>Must</b>	The web-interface shall provide a possibility to create reports.
<b>SR-3.2</b>	<b>Must</b>	The web-interface shall provide the possibility to generate reports of energy consumption and costs. This includes a daily/weekly/monthly executive report and a detailed energy consumption report per device.
<b>SR-3.3</b>	<b>Must</b>	The system shall be able to generate charts and graphs.
<b>SR-3.4</b>	<b>Must</b>	The system shall be able to send reports automatically per email. This also includes a view in the web-interface, which lets the user specify the reports that should be received automatically.
<b>SR-4.1</b>	<b>Must</b>	The system shall provide a possibility to create/edit/remove policies from the web-interface.
<b>SR-4.2</b>	<b>Must</b>	The system shall be able to provide policies based on the types of device. A washing-machine offers different actions than a light.
<b>SR-4.3</b>	<b>Must</b>	The system shall be able to execute policies based on triggers.

<b>SR-5.1</b>	<b>Must</b>	The web-interface shall require authentication and authorization.
<b>SR-5.2</b>	<b>Must</b>	The web-interface should provide a possibility to change the username and password.
<b>SR-6.1</b>	<b>Must</b>	The system shall be able to perform an automatic update of the HPS software without requiring user-interaction.
<b>SR-6.2</b>	<b>Must</b>	The system shall inform the user prior to an update and after the success/failure of an update.
<b>SR-6.3</b>	<b>Must</b>	The system shall be able to be connected to the internet.
<b>SR-6.4</b>	<b>Must</b>	The system should create a full backup of all user data (monitor data, policies, settings, bills, etc.) and configurations files before performing an update. The backup needs to be created in such a way that the software can be easily restored to the previous version, in case an update fails.
<b>SR-6.5</b>	<b>Must</b>	The system should restore the backup automatically in case an update fails. It is important that the HPS remains operational.
<b>SR-6.6</b>	<b>Must</b>	The updates shall be pushed to the system from one central server.
<b>SR-7.1</b>	Optional	The web-interface shall provide a possibility to export raw load measurements The user can specify the data and the timespan..
<b>SR-7.2</b>	Optional	The web-interface shall provide a possibility to download reports and bills.

## 4.7 Technical Requirements

In this chapter a more technical representation is given of the functional requirements. TR-1.\* reverse to SR-1. TR-2.\* reverse to SR-2 and so on.

<b>TR-1.1</b>	<b>Must</b>	The system shall communicate over ZigBee Smart Energy with the Smart Devices and the Smart Meter.
<b>TR-1.2</b>	<b>Must</b>	The system shall request the energy-consumption of each device every 300 seconds in a round-robin manner.
<b>TR-1.3</b>	<b>Must</b>	The system is limited to maximally monitor 50 devices at the same time.
<b>TR-1.4</b>	<b>Must</b>	The system should communicate over ZigBee Home Automation with the Smart Devices.
<b>TR-3.1</b>	<b>Must</b>	The reports should be downloadable in PDF format.
<b>TR-4.1</b>	<b>Must</b>	The system and web-interface shall provide Price-based, Time-based and Budget-based triggers.
<b>TR-5.1</b>	<b>Must</b>	The web-interface shall be compatible with the W3C XHTML standard for browser compatibility.
<b>TR-6.1</b>	<b>Must</b>	The HPS shall provide a RJ-45 100Mbit Ethernet interface to connect to the local network.
<b>TR-7.1</b>	<b>Must</b>	The raw load measurements should be exported in CSV-format. Including a timestamp and the energy consumption at the given timestamp per device or total
<b>TR-7.2</b>	<b>Must</b>	The system shall provide a operation and a maintenance mode. During maintenance mode the internet connection is not available.

<b>TR-7.3</b>	Optional	The system shall be able to enter maintenance mode based on external events (events that can be issued without running software on the machine)
<b>TR-7.4</b>	Optional	The user-interface shall provide a possibility to put the system into maintenance mode.

## 4.8 Commercial Non-Functional Requirements

In this chapter are the commercial non-functional requirements outlined.

<b>CF-1</b>	<b>Must</b>	The hardware of the product should be able to be placed at a location that it will not bother the end-user, and be of a neutral color either white, grey or black.
<b>CF-2</b>	<b>Must</b>	The production cost of the HPS unit including two Socket Meters should not exceed 200,00 Euro. The HPS itself should cost 120,00 Euro or less, two Socket Meters should be 50,00 Euro or less, leaving the rest of the budget, 50,00 Euro to development of the software.
<b>CF-3</b>	<b>Must</b>	The price of the HPS should not exceed 330,00 Euro per HPS. This way the product adoption will be high, and ESH will make a healthy profit margin per device.
<b>CF-4</b>	<b>Must</b>	The device should have proper utilization manuals in the Dutch and English language.
<b>CF-5</b>	<b>Must</b>	The user-interfaces should be tested during the development of the product, in a number of scenarios to at least 8 test subjects. The subjects or testers will be selected using Hall Intercept Testing. The 8 users are based on 'Five users is enough' by Jakob Nielsen which states that 8 users will detect 85% of all the usability problems [54]. The formula is where p is the probability of one subject identifying a specific problem and n the number of subjects (or test sessions). $U = 1 - (1 - p)^n$
<b>CF-6</b>	<b>Must</b>	The user-interface of the HPS should be available in the native language and be extendable with other languages
<b>CF-7</b>	<b>Must</b>	The hardware should on average work 5 years and a minimal of 2 years.

## 4.9 Technical non-functional Requirements

In this chapter are the technical non-functional requirements outlined. For this chapter are reliability, usability, adaptability, recoverability, security and compliance outlined.

### 4.9.1 Reliability

<b>NF-1.1</b>	<b>Must</b>	The system should be available 99.9%. $\alpha = \frac{\text{mean time to failure}}{\text{mean time to failure} + \text{mean time to repair}} = \frac{3 \text{ years}}{3 \text{ years} + 1 \text{ day}}^5$
<b>NF-1.2</b>	<b>Must</b>	The down-time caused by the installation of an update shall not be greater than 10 minutes.
<b>NF-1.3</b>	<b>Must</b>	The replacement of an old software version of HPS by a new version shall not cause loss of information.

<sup>5</sup>We assume that we have to send a technical support to the house.

<b>NF-1.4</b>	<b>Must</b>	The database shall be consistent at any time.
---------------	-------------	---

#### 4.9.2 Usability

<b>NF-2.1</b>	<b>Must</b>	The End-user wants to learn how to use the system such as creating policies, configuring devices and displaying the current energy bill. The system provides a context sensitive help. The End-user is able to follow the instructions and complete the task within five minutes.
<b>NF-2.2</b>	<b>Must</b>	The End-User wants to use the user-interface. The average time to display the user-interface to the user should be 1000 milliseconds.
<b>NF-2.3</b>	<b>Must</b>	The End-User wants to use the user-interface. The maximum time to display the user-interface to the user should be 5000 milliseconds.
<b>NF-2.4</b>	<b>Must</b>	Actions which cannot be undone should ask for confirmation
<b>NF-2.5</b>	<b>Must</b>	Error messages should explain how to recover from the error.
<b>NF-2.6</b>	<b>Must</b>	The interface actions and elements should be consistent.
<b>NF-2.7</b>	<b>Must</b>	The user-interface shall provide adequate feedback to the user or operations in form of a waiting indicator as described in [28].
<b>NF-2.8</b>	<b>Must</b>	The system shall be able to deliver messages (e-mails) in near real-time.

#### 4.9.3 Adaptability

<b>NF-3.1</b>	<b>Must</b>	The developer wished to add support for new HAS devices. During design time he makes modification without affecting other functionality. The number of changing classes shall not be higher than two.
<b>NF-3.2</b>	<b>Must</b>	The system shall be able to understand and manage self-descriptive interfaces/data in the context of the ZigBee Smart Energy 2.0 Application protocol.
<b>NF-3.3</b>	<b>Optional</b>	The system's SAI <sup>6</sup> shall not be lower than 0,45 in respect to environment changes.
<b>NF-3.4</b>	<b>Must</b>	The system's SAI shall not be lower than 0,75 in respect to protocol changes.

#### 4.9.4 Recoverability

<b>NF-4.1</b>	<b>Must</b>	The user data shall be backed up daily on a local storage.
<b>NF-4.2</b>	<b>Optional</b>	In the case of a severe hardware failure of the HPS all data shall be recoverable and transferable to the new/replaced device.
<b>NF-4.3</b>	<b>Must</b>	The system does not receive load measurement responds from the external Smart Meter. The system should log the event and continue to operate in normal mode. There is no downtime.
<b>NF-4.4</b>	<b>Must</b>	The system is not able to communicate with a device (e.g. Socket Meter, ZigBee device). The system should reestablish the connection.

<sup>6</sup>Software Adaptability Index as described in [39]

#### 4.9.5 Security

<b>NF-5.1</b>	<b>Must</b>	The HPS should have a software firewall only allowing incoming connections from ESH.
<b>NF-5.2</b>	<b>Must</b>	The HPS should validate update-files by calculating a hash-checksum.
<b>NF-5.3</b>	<b>Must</b>	<p>Password representations shall be saved as cryptographic hash values. One of the following well-known algorithms shall be used to generate the hash values:</p> <ul style="list-style-type: none"> <li>• MD5</li> <li>• SHA2</li> <li>• Whirlpool</li> </ul>
<b>NF-5.4</b>	<b>Must</b>	Passwords shall never be shown on the screen in plaintext.
<b>NF-5.5</b>	<b>Must</b>	At run-time the access to all externally visible areas of HPS shall be restricted to authorized and authenticated users.
<b>NF-5.6</b>	Optional	Support open source security methods.
<b>NF-5.7</b>	Optional	Users shall not be permitted to have more than one simultaneous login to HPS

#### 4.9.6 Compliance

<b>NF-6.1</b>	<b>Must</b>	The HPS must be ZigBee Smart Energy v2 compliant [3].
<b>NF-6.2</b>	<b>Must</b>	The HPS user-interface must be W3C compliant. This includes valid Mark-up and stylesheets.
<b>NF-6.3</b>	<b>Must</b>	The HPS must be ZigBee Home Automation compliant [2].

#### 4.10 Risk Assessment

The risk severity in Appendix A4 is calculated according to the impact and likelihood possibilities in Table , also known as ILS <sup>7</sup>.

Impact	<i>High</i>	Medium	High	Critical
	<i>Medium</i>	Low	Medium	High
	<i>Low</i>	Very low	Low	Medium
		<i>Low</i>	<i>Medium</i>	<i>High</i>
Likelihood				

Table 12: Risk severity based on the impact and likelihood of a risk.

---

<sup>7</sup>Impact, Likelihood, Severity

## 5 Analysis

In this section is the analysis outlined. This section starts with assumptions made about (the environment of) the HPS. Subsequently, the technology roadmaps are outlined. The section ends by presenting the high-level design decisions.

### 5.1 Assumptions

There are several assumptions about (the environment of) the HPS:

- The Smart Meter can provide new prices every hour ahead.
- The customer has internet access and has a browser.
- The customer has a compatible Smart Meter installed, which provides the total power input and output.
- The Smart Meter can be connected by ZigBee.
- The Smart Meter relays the total energy consumption and energy production.
- All devices are required to connect to the HPS by ZigBee.
- The Smart Meter is able to store the energy data hourly for at least one year in case the HPS is offline.
- The utility publishes prices through the Smart Meters.
- The amount of software updates to the HPS will not exceed once a month.
- Wireless security is provided by ZigBee (e.g. Sniffing of data-packages or device-hijacking)

### 5.2 Technology Roadmaps

In Figure 6 the outline of the HPS device is described.

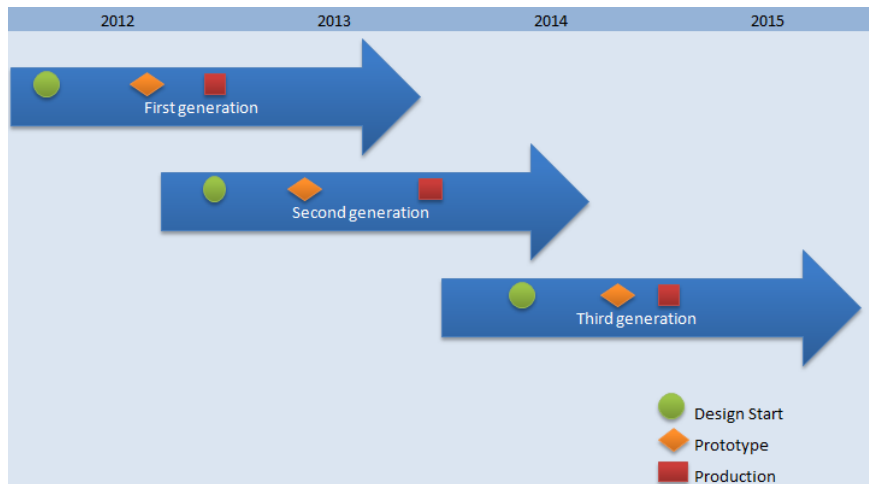


Figure 6: Technology roadmap of the HPS device.

The first generation of the HPS will include the features pointed out in Section 4. The second generation will bring cost reductions made possible by newer hardware [51], combined with compatibility for the Smart Grid in the US market. The third generation of the HPS brings more

cost reductions to the core components and features compatability with more HAS.

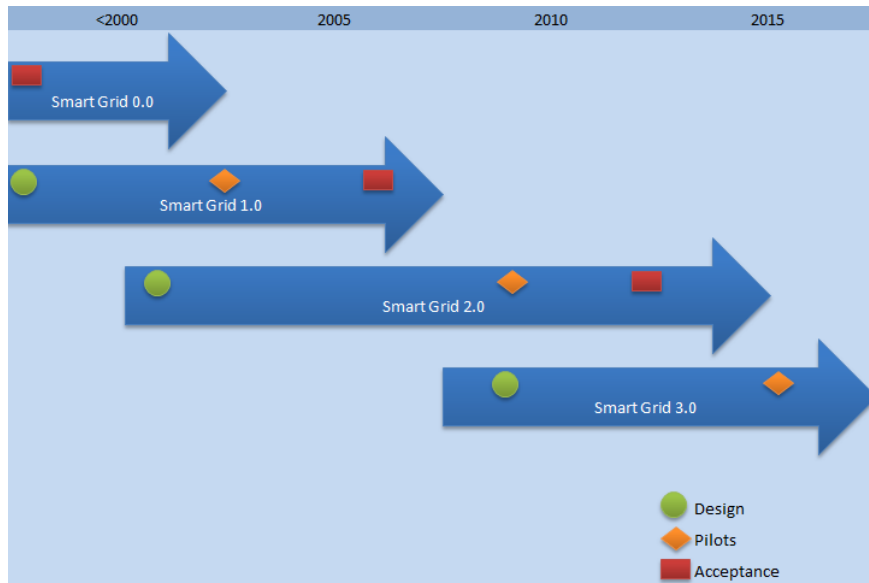


Figure 7: Technology roadmap of the Smart Grid based on [42].

The Smart Grid roadmap depicted in Figure 7 starts with the current power grid, which hasn't changed since the 1920's [47]. The development of the first version of the Smart Grid, version 1.0, was accepted for general use around 2005 [12]. Smart Grid version 2.0 started with pilot projects around the year 2010, with the finalized version of 2.0 expected to be accepted for general use in 2012 [42]. Smart Grid version 3.0 is still in the design stage, with pilot projects sceduled in the future after the design is near completion [13].

### 5.3 High-level Design Decisions

The high-level design decisions are based upon a process for architectural decisions and tool WebGrid 5. The alternatives are weight against the concerns, next the concerns are given weights based on the \$ 100 method.

Table 13: Decision - Local Unit

Name	Local Unit
Decision	13
Status	<i>Approved</i>
Problem/Issue	The HPS software needs to run on a hardware device that is able to communicate with the HAN.
Decision	The HPS will run on a local hardware unit, which is placed into each household.
Alternatives	<p><i>Thin Client/Online Service</i></p> <p>The user has only a small device which sends data over the internet to the ESH service. The ESH service is responsible for storing data and performs the calculations.</p> <p><i>Home PC</i></p> <p>The HPS software runs on the user's own computer.</p>
Arguments	<p>The local unit is safe, does not depend on the availability of the the internet and the user convenience is higher because it does require the Home PC. In contrast the Online Service is highly dependable on the internet and has high-operating costs that will be reflected in the price for the user. The main disadvantage of the Home PC is, that the PC is not always on and thus cannot measure loads or make use of the real-time pricing.</p> <p><b>Webgrid in Figure 26 in Appendix A2.1</b></p>



Table 14: Decision - ZigBee Smart Energy v2

<b>Name</b>	<b>ZigBee Smart Energy v2</b>
<b>Decision</b>	<b>14</b>
<b>Status</b>	<i>Approved</i>
<b>Problem/Issue</b>	The HPS needs to communicate and control devices in the HAN.
<b>Decision</b>	The HPS will use ZigBee Smart Energy v2 and ZigBee Home Automation to communicate and control devices in the HAN. ZigBee Home Automation will be used as an addition to support a larger number of devices, since Smart Energy is still in development.
<b>Alternatives</b>	<p><i>X-10</i> X-10 is a well-established home automation standard that works both over power-line and radio.</p> <p><i>UPB</i> UPB is an improvement of X-10 which increases reliability and works over power-line.</p> <p><i>Z-wave</i> Is a proprietary wireless protocol for home automation.</p>
<b>Arguments</b>	<p>The choice for the ZigBee protocol is based on the fact that it supports a lot of devices and has a high degree of interoperability not seen in the other protocols. Although X-10 is affordable, its lack of robustness, one-way communication, susceptible to interference and limited support are huge disadvantages. UPB has the disadvantage of the power-line-only communication and it also comes with high cost and low acceptance. Concerns of Z-wave are its proprietary technology, radio congestion with larger deployments, and low tolerance for failed, moved, or removed devices.</p> <p><b>Webgrid in Figure 27 in Appendix A2.1</b></p>

Table 15: Decision - Linux

<b>Name</b>	<b>Linux</b>
<b>Decision</b>	<b>15</b>
<b>Status</b>	<i>Approved</i>
<b>Problem/Issue</b>	The HPS device and server will need to run on a platform to run the developed software.
<b>Decision</b>	The HPS device and server will run on the Linux platform.
<b>Alternatives</b>	<p><i>Linux</i> Open Source, majority of distributions and software is free, reliable, secure, but lacks hardware compatibility, software availability, is hard to use, and has no/limited support.</p> <p><i>Windows</i> Easy to use, software availability, hardware compatibility, support, but license costs, majority of software is not free, less secure and reliable than Linux, not Open Source.</p> <p><i>Mac OS X</i> Unix-based, easy to use, but lacks hardware compatibility even more than Linux, license costs, cannot be upgraded/customized, and software availability.</p> <p><i>Android</i> Unix-based, designed for less powerful hardware, but has a feature overhead (e.g. call, contacts), and requires ARM CPU's unless unofficially ported.</p> <p><i>JavaOS</i> Low overhead, but requires Java SPARC hardware architecture.</p>
<b>Arguments</b>	<p>Linux is low cost and a ready to go platform to build the application up on. Also in terms of security and reliability, which are two very important factors in the success of the HPS device, comes Linux first. Even though the hardware compatibility of Linux isn't on the same level as Windows, the HPS device is supported by Linux out of the box (it ships with Linux [22]) and all modern hardware is supported through drivers by the Linux kernel [30], independent on the Linux distribution, so the server is also supported by Linux out of the box. Ease of use will not be a problem for ESH as the server is going to be managed by an external company and the HPS will only be accessible through the webinterface for the user. All things considered makes Linux the best choice as the operating system used on the HPS device and ESH server.</p> <p><b>Webgrid in Figure 28 in Appendix A2.1</b></p>

Table 16: Decision - Local Data Storage

Name	Local Data Storage
Decision	16
Status	<i>Approved</i>
Problem/Issue	The energy consumption needs to be stored somewhere to be able to report and inform the end-user.
Decision	The data will be stored locally on the HPS.
Alternatives	<p><i>Online Storage</i></p> <p>Storage online will allow for location independent accessibility, but will require a high investment, creates a security risk and completely depends on the internet.</p> <p><i>Mixed Storage</i></p> <p>One part of the data is stored on theHPS, the rest on ESH servers. This would improve reliability, although it will be the most expensive solution since it requires full storage capability on the local unit as well as a number of servers.</p>
Arguments	<p>To keep the costs low and the reliability high the best option is a local storage. ESH core competence are, at the current moment, HAS systems. Going for a part or full online service will require a high investment on the part of ESH. Also the security and privacy of the user will require a high cost to keep safe.</p> <p><b>Webgrid in Figure 29 in Appendix A2.1</b></p>

Table 17: Decision - Web-interface

Name	Web-interface
Decision	17
Status	<i>Approved</i>
Problem/Issue	The end-user needs a possibility to access and control the HPS
Decision	A web-interface will be used that can be called from LAN of the end-user. It is assumed that our target audience already owns a device that has a browser (e.g. PC, Laptop, Tablet, Mobile device (on Wi-Fi network), TV with browser, etc).
Alternatives	<p><i>Attached touchscreen</i> A touchscreen mounted on or close to the HPS device</p> <p><i>External screen/peripherals</i> A monitor that is connected to the HPS device but it uses mouse and keyboard as input device.</p> <p><i>Native app</i> A native app running on the PC of the end-user.</p> <p><i>Mobile app</i> A mobile app running on the the Mobile device of the end-user.</p>
Arguments	<p>A disadvantage of both the touchscreen and the external monitor is, that it adds cost to the product. It is highly possible that the HPS will be located in a utility room without direct access, it will be difficult to reach the device. Moreover, the device will be a lot larger in size. Both the mobile and native app have the disadvantage that they only support a certain group of devices or require a lot of effort to port the software to all kinds of platforms. In conclusion, the web-interface provides the best solution when it comes to costs and convenience. It also provides the same amount of usability as a native application.</p> <p><b>Webgrid in Figure 30 in Appendix A2.1</b></p>

Table 18: Decision - Push Updates

Name	Push Updates
Decision	18
Status	<i>Approved</i>
Problem/Issue	The HPS needs to be updated in order to fix bugs and adapt to new requirements
Decision	Updates will be rolled out via Push-notification from an external ESH server.
Alternatives	<p><i>Pull updates</i> The HPS device periodically checks for and install new updates</p> <p><i>Manual updates</i> The end-user has to issue the update manually from the web-interface.</p> <p><i>No online updates</i> The HPS can only be updated offline via USB or SD-Card.</p>
Arguments	<p>Pushing updates from the ESH server to the HPS devices has the advantage that the HPS devices are always up-to-date and the user does not has to care about updating HPS. This reduces maintenance costs. In contrast to the Pull and Manual Update mechanism, the ESH update server don't have to available all the time but during deployment phases. However, the pushing updates introduces potential security vulnerabilities because updates are requested over the internet.</p> <p><b>Webgrid in Figure 31 in Appendix A2.1</b></p>

## 6 System Architecture

This section outlines the system architecture. It starts with the system context and ends with the verification from the system context.

### 6.1 System Context

The system context, based on [26], is a fundamental artifact in the software architecture of a system. Developing the system context view is important, because this view is used as a mechanism to trace back to the business context, and downstream to the functional and operational architecture.

#### 6.1.1 Diagram

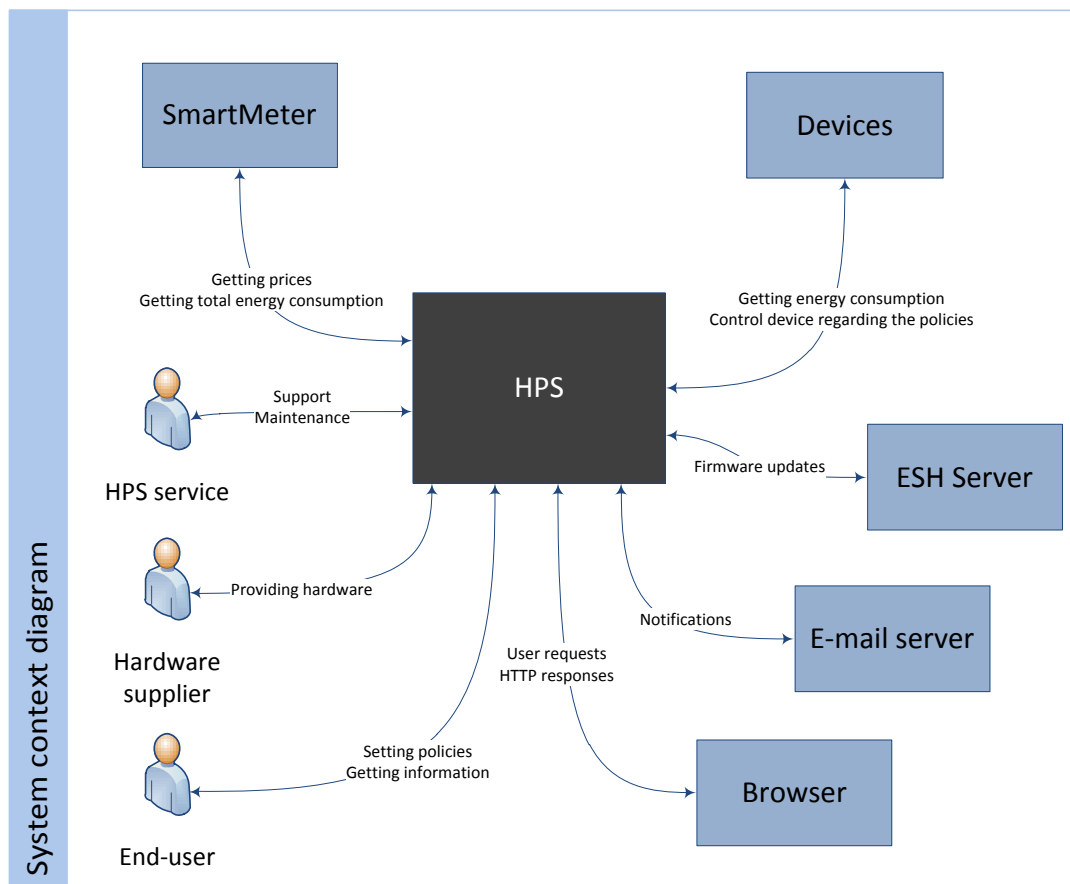


Figure 8: System context diagram

This system context diagram:

- Represents the HPS as a black box
- Depicts its interaction with external entities (systems and end users)
- Identifies the information and control flows between the system and external entities

External entities are not necessarily systems that are outside the system scope. The devices for example are in fact part of the system but are presented as external entity because ESH don't deliver them.

### 6.1.2 Users and Roles

**End-user** The end-user is the person that uses the HPS. The HPS will be installed in his/her house and can manage the HPS by using a browser. The end-user has to add devices to the HPS by connecting its ZigBee device or by adding a Socket Meter to the device. The goal is that the end-user benefits from the HPS by reducing the energy consumption. This is done by giving smart policies and getting detailed consumption information.

**HPS service** When the HPS is broken or doesn't work like expected there is a possibility to call the HPS service to solve this. When it's a problem that can not be solved by phone a mechanic has to come to the house where the HPS is installed.

**Hardware supplier** The hardware supplier provides the hardware for the system, think of parts of the HPS and the Socket Meters.

### 6.1.3 External Systems

**Smart Meter** The Smart Meter is an important external entity in our system. This is because the HPS relies on the Smart Meter to get prices one hour ahead (composed by the utility). With this price information the HPS can make decisions when to execute a policy or not. An other important aspect is that we get the total energy consumption from this Smart Meter. This is necessary because the HPS can not exactly know what the total energy consumption is from the house since it does not have the certainty that the end-user will connect *all* of his devices/electronics to the HPS. With the total energy consumption the HPS can give really nice statistics and detailed reports about the (relative) consumptions of devices.

**Devices** The devices can be split up in two categories: a Zigbee enabled Smart Device or a legacy device with a Socket Meter. Both devices can be added to HPS to at least monitor its energy consumption. As mentioned before the end-user has to connect its devices itself by configuring/identifying it through the web-interface using a browser. If it's a legacy device and the end-user wants to measure its power consumption he/she has to buy a Socket Meter and put it between the power socket and the legacy device. A Zigbee enabled Smart Device can also perform some smarter actions like dimming lights. For this smart devices the user can configure very detailed policies.

**ESH Server** The server that pushes firmware updates over the internet to the HPS.

**E-mail server** The E-mail server can send notification mails to the user if he wants to. The end-user can specify when he want to receive an e-mail, for instance every week a report of the energy consumption of a device. This can also be done via the web-interface using a browser.

**Browser** As mentioned before the browser is the (only) way to manage the HPS, that can be adding/removing a device, setting a policy, consulting a report, viewing the bill etc.

#### 6.1.4 Channels and Information Flows

Channel: HPS $\longleftrightarrow$ Smart Meter	
<b>Description</b>	The HPS gets the energy prices from the Smart Meter
<b>Connection</b>	Wireless, Wi-Fi 802.11b/g
<b>Protocol</b>	ZigBee, Smart Energy Profile 2.0 Application Protocol [3]
<b>Data volume</b>	Every hour

Channel: HPS $\longleftrightarrow$ Smart Meter	
<b>Description</b>	The HPS gets the total energy consumption from the Smart Meter
<b>Connection</b>	Wireless, Wi-Fi 802.11b/g
<b>Protocol</b>	ZigBee, Smart Energy Profile 2.0 Application Protocol
<b>Data volume</b>	On request

Channel: HPS $\longleftrightarrow$ Devices	
<b>Description</b>	The HPS gets the energy consumption from the connected device
<b>Connection</b>	Wireless, Wi-Fi 802.11b/g
<b>Protocol</b>	ZigBee, Smart Energy Profile 2.0 Application Protocol or Home Automation Profile [2]
<b>Data volume</b>	Every 300 seconds

Channel: HPS $\longleftarrow$ ESH Server	
<b>Description</b>	The ESH server pushes firmware updates to the HPS
<b>Connection</b>	Wired, internet
<b>Protocol</b>	TCP/IP
<b>Data volume</b>	Depends on updates

Channel: HPS $\Longrightarrow$ E-mail server	
<b>Description</b>	The HPS sends messages to the E-mail server that will care for sending the E-mail to the end-user
<b>Connection</b>	Wired, internet
<b>Protocol</b>	SMTP
<b>Data volume</b>	Depends on user policies

Channel: HPS $\longleftrightarrow$ Browser	
<b>Description</b>	The HPS has a web server that handles requests from a browser (used by a end-user)
<b>Connection</b>	Wired, internet
<b>Protocol</b>	HTTP over TCP/IP
<b>Data volume</b>	Depends on user



### 6.1.5 Alternatives

There are some alternatives in this system context that were considered.

#### **ZigBee**

ZigBee will be used a lot in the system to communicate with several components. That is directly the main reason why ZigBee is chosen, because it's high connectivity and adaptability. However, there are other protocols to communicate with the devices (cf. Table 14) that have been investigated. Nevertheless, ZigBee came out as the best option.

#### **E-mail**

The HPS has to send notifications when the end-user has configured to receive alerts or information about some devices. To do this e-mail is chosen because it's relatively easy to implement and it's widely used (everyone has e-mail, and nowadays a lot people receive it on his cell phone). Alternatives are SMS, some alert system built on the HPS or built a mobile notify application.

#### **External E-mail server**

As figure 8 shows, the HPS will use an external e-mail server to send the e-mails to the user. This means HPS isn't responsible for the actual sending of the e-mail but uses a third party (which will be probably the SMTP server of the ISP of ESH or another free SMTP server like Gmail). Mind that this is not a total new entity in the system but should be taken into account because the HPS does have a connection with it and has to know how to communicate with it. An alternative would be to built an own local SMTP server inside the HPS. Because of the extra developments costs to built an own, the possibility to have SPAM problems and the large availability of free SMTP servers the decision fell on external e-mail/smtp server.

#### **Getting prices**

To get the prices there are several options, getting the prices via internet directly from the utility, getting the prices from the ESH server or getting the prices from the Smart Meter using ZigBee. Because the wide usage of ZigBee and existing connection with the Smart Meter, getting prices from the Smart Meter is chosen.

## 6.2 Verification

The feasibility of the system architecture is verified below.

### 6.2.1 Availability

The possible availabilities of a few key components in the system are discussed and considered.

Let

MTBF: Mean Time To Failure

MTTR: Mean Time To Repair

The availability of the system is represented as  $\alpha$

$$\alpha = \frac{MTTF}{MTTF+MTTR}$$

The value of MTTF and MTTR are estimated, respectively.

Component	MTTF	MTTR	Availability ( $\alpha$ )
HPS Box	5 years	1 day	0.999452
Wifi Connection	3 years	1 day	0.998632
Internet Connection	3 year	1 day	0.998632
Mail Dispatch	3 years	1 day	0.998632
Web Interface	3 years	1 day	0.998632
Storage	5 years	1 day	0.999452

Table 19: System Architecture Availability.

Based on the above table, the system availability is calculated as the multiplication of all component availability. The result is 0.995264.

### 6.2.2 Time to Market

Wifi is a common communication mechanism nowadays. It is currently available in the market and ready for use in HPS. However, it needs time to set up the wifi within HPS system.

HPS can easily use the internet in order to enable HPS box communicate to ESH server. The time to set the connection up within HPS system is estimated 5 minutes.

Mail dispatch as an application for generating e-mail in HPS has not been ready yet. However, the SMTP protocol is available currently. Thus, it needs time to develop the application. The time is estimated as 1 week.

Web Interface as an application for providing billing information and for updating the policy has not been existed yet. Thus it needs time to develop that application. The time is approximately 4 weeks.

Storage chosen is SD Card which is available in the market recently. There is no time to wait for its production, but it needs time to configure and install the HPS software inside it.

The HPS box as the main hardware has not been ready yet. It needs time to produce and test before it is installed with HPS software system. The production deals with hardware supplier. The estimated the time to produce is 8 weeks. In addition, the time to develop the software system is approximated as 12 weeks.

Based on the above consideration, it is reasonable to develop HPS within 1 year since HPS shall be released in first quarter of 2013.

### **6.2.3 Update**

The update mechanism needs internet connection between HPS and ESH Server using HTTP/FTP protocol. From the connection point of view, it is possible to send the updated data through the internet as long as the connection is established and stable.

The next consideration is the amount of data sent from ESH server. The file consisting updated firmware is estimated 1 MB in size. While transferred through internet with bandwidth 10 Mbps, it needs approximately 0.8 seconds for the file to be completely sent. The storage of HPS, SD card, is capable to store 8 MB data. Thus, it is possible to save the updated firmware.

### **6.2.4 Device Communication**

HPS communicates with the devices through Wifi. A typical Wifi router using 802.11b or 802.11g, for instance, with a stock antenna might have a range upto 32 meter indoor. Therefore, it is possible to transfer data using Wifi inside a house, especially a small house. However, it needs certain installation configuration to make sure that all considered devices can be properly connected with HPS. The positions of devices against HPS box have to be in range of 32 meter. Regarding the data size, Wifi is enable to transfer data upto 11 Mbps. Thus it is possible to use Wifi to send data of energy consumption from devices.

### **6.2.5 Cost and Return of Investment**

As mentioned in Section 2.8 Financial Model, there are some cost to be considered. The salary for software architecture team is €96.000,00 for 10 weeks. The development cost is €408.000,00 for 204 weeks. The hardware cost is €178,22 per piece. For 10000 units, the cost per unit is €279,50.

The total investment is €504.178,22. During HPS development phase, there is no income for about one year. In the first quarter of 2013 is HPS launched. Then, the income is obtained from product selling. If the monthly income is about €10.000,00, within 4 years after product release the return of investment is recovered.

## 7 Hardware architecture

In this section is the hardware architecture outlined. This section starts with the hardware design decisions. Subsequently is the hardware overview outlined. After the hardware overview are the main device, socket meter and , process view and server information outlined.

### 7.1 Hardware Design Decisions

There are several decisions to be made before specifying the server hardware. Each decision is listed in a specific item.

Table 20: Decision - GuruPlug Server

<b>Name</b>	<b>GuruPlug Server</b>
<b>Decision</b>	<b>20</b>
<b>Status</b>	<i>To be reviewed</i>
<b>Problem/Issue</b>	The HPS software needs to have an off-the-shelf local hardware device.
<b>Decision</b>	The HPS will use the GuruPlug Server.
<b>Alternatives</b>	<p><i>Soekris net4501</i> Low-power and cheap computer (€143,00), but very low hardware specifications (133 MHz CPU, 64MB RAM) and WiFi requires the purchase of a separate module.</p> <p><i>Fit-PC2i</i> Small, powerful (1.6 GHz CPU, 1GB RAM), includes WiFi, but expensive (€233,00) and uses a lot of power on its own.</p> <p><i>Zotac ZBOX SD-ID12 Plus</i> Powerful (1.8 GHz CPU, 1GB RAM), 250GB space, relatively cheap (€175,00), includes WiFi, but uses a lot of power on its own.</p> <p><i>GuruPlug Server</i> Relatively powerful (1.2 GHz CPU, 512MB RAM), low-power, includes WiFi, cheap (€94,92), but lacks storage capacity (can be extended with MicroSD slot).</p>
<b>Arguments</b>	<p>The GuruPlug Server meets all our requirements, it sells at the lowest price between alternatives while still featuring decent hardware. Another advantage of the GuruPlug Server is its power consumption compared to the other hardware devices.</p> <p><b>Webgrid in Figure 32 in Appendix A2.1</b></p>

Table 21: Decision - SD card

<b>Name</b>	<b>SD card</b>
<b>Decision</b>	<b>21</b>
<b>Status</b>	<i>Approved</i>

<b>Problem/Issue</b>	The HPS device requires storage in order to save the energy prices and energy consumption and production of attached devices.
<b>Decision</b>	The HPS will use a SD card.
<b>Alternatives</b>	<p><i>Harddisk</i> Very cheap (250GB for 30 EUR) and fast, but also pretty large, noisy and uses more energy than alternatives.</p> <p><i>SSD</i> Quiet, fast, low-power, but expensive (32GB for 50 EUR), large and limited write and read cycles.</p> <p><i>USB stick</i> Quiet, small, Low-power, can be used to transfer data to another device, but expensive (16GB for 30 EUR), requires USB port, relatively slow, protrudes from the HPS device, and can get lost.</p> <p><i>SD card</i> Quiet, small, low-power, relatively cheap (16GB for 15 EUR), but relatively slow.</p>
<b>Arguments</b>	<p>The SD card is the best choice all round, it uses almost no power, can be placed inside the device and is relatively cheap. The SSD and harddisk share the common problem that they are rather large and thus cannot be placed inside the GuruPlug, resulting in an external case or increased box around the device which would result in additional costs. The USB stick is more similar to the SD card in terms of features but the price and protrusion from the devices makes it come second.</p> <p><b>Webgrid in Figure 33 in Appendix A2.1</b></p>

Table 22: Decision - Plugwise Circle Socket Meter

<b>Name</b>	<b>Plugwise Circle Socket Meter</b>
<b>Decision</b>	<b>22</b>
<b>Status</b>	<b><i>Approved</i></b>
<b>Problem/Issue</b>	The HPS needs to use Socket Meters in order to feature support for legacy devices.
<b>Decision</b>	The HPS will use the Plugwise Circle Socket Meter.

<b>Alternatives</b>	<p><i>ZigBee home automation wireless power meter socket [23]</i> Relatively cheap (40 USD each, excluding VAT), LCD screen, USB and ZigBee support, but uses wrong power socket plug and high shipping costs (20 USD each).</p> <p><i>Power Meter JNX-2000A [38]</i> Very cheap (20 EUR each), LCD screen, supports western (EU, US) power socket plugs, but no support for ZigBee or even wireless, and doubtful quality.</p> <p><i>Plugwise Circle Socket Meter [35]</i> Developed and assembled in The Netherlands, support in The Netherlands, relatively cheap (325 EUR for 10, including VAT, retail price), supports western (EU, US) power socket plugs, but has no LCD screen.</p>
---------------------	--

<b>Arguments</b>	<p>The Plugwise Circle Socket Meter is the only real option for a socket meter. The JNX-2000A has no support for ZigBee or even wireless, making it impossible to connect to the HPS. The ZigBee home automation wireless power meter socket contains all the requirements for the Socket Meter, but doesn't support European power socket plugs. Adapters can be made for the power socket plugs, but the price is still higher than the Plugwise Circle Socket Meter and its support would be in Hongkong instead of The Netherlands.</p> <p><b>Webgrid in Figure 34 in Appendix A2.1</b></p>
------------------	---

Table 23: Decision - Outsourced update server

<b>Name</b>	<b>Outsourced update server</b>
<b>Decision</b>	<b>23</b>
<b>Status</b>	<i>Approved</i>
<b>Problem/Issue</b>	The HPS requires a server to receive the software updates from.
<b>Decision</b>	The HPS will outsource the software update server to a third party.

## Alternatives

### *In-house*

Full control, full ownership, typically lower monthly costs, in-house support, but high initial expenditure, expensive to scale, and requires capable staff.

### *Cloud*

Predictable monthly costs, 24/7 support (on hardware problems only), but lack of control, and requires capable staff.

### *Outsourced/managed*

Predictable monthly costs, 24/7 support (on hardware and software), improves focus on core competencies, but lack of control, and requires full trust in external company's employees.

### *Colo*

More control than cloud and managed, full ownership of hardware and software, some external support (e.g. for reboots), typically lower monthly costs, but high initial expenditure, expensive to scale, and requires capable staff.

*Sources used: [9, 10, 15, 40]*

---

## Arguments

Cloud hosting and in-house hosting share a lot of similarities in advantages and disadvantages. The biggest disadvantage is that it requires capable staff to maintain the servers and since server hosting is not a core competency of ESH, it has to hire new staff which has to be available 24/7 in case of problems with the server. Another disadvantage is the high initial expenditure required, with cloud and outsourced/managed hosting there are predictable monthly costs, with typically a single setup fee. Cloud is a mixture between outsourced/managed hosting and in-house hosting and colo hosting. The server hardware is maintained by the external company, but the software maintenance is not, which still requires capable staff to be available 24/7 in case of problems with the server. With the outsourced/managed hosting both hardware and software is maintained by the external company and there are predictable monthly costs. This results in the outsourced/managed update server as the best possibility.

**Webgrid in Figure 35 in Appendix A2.1**

---

Table 24: Decision - Debian - Linux Distribution

Name	Debian - Linux Distribution
Decision	24
Status	<i>Approved</i>
Problem/Issue	The HPS device and ESH server need a Linux distribution to run the HPS software on.
Decision	The HPS device software will run on Debian and ESH server software will run on CentOS.
Alternatives	<p><i>Debian</i> Package availability, easy to upgrade, stable, easy to maintain, commercial support by third party available, but lacks on support length.</p> <p><i>CentOS</i> Support length, stable, low on resources, easy to maintain, performance, commercial support available, but lacks on package availability, and hard to do a live upgrade to a newer version.</p> <p><i>Ubuntu</i> Package availability, easy to upgrade, easy to maintain, but lacks on support length, and is developed originally for desktop purposes.</p> <p><i>Fedora</i> Up-to-date packages, commercial support by third party available, easy to maintain, but lacks on package availability, and hard to do a live upgrade to a newer version.</p> <p><i>Sources used: [21, 48]</i></p>
Arguments	<p>CentOS is chosen for its active development, commercial support, stability, easy maintenance and friendly environment for users and package maintainers. Debian is a close second, but due to less support (both in length and commercial availability) is it less suitable for the ESH server. Both Ubuntu and Fedora lack several important advantages compared to CentOS and Debian and are not considered as a fair alternative for the ESH server. The GuruPlug Server ships default with Debian [48] and since there is no ARM version of CentOS, it is not possible to install CentOS on the GuruPlug Server. Since Debian is already installed on the GuruPlug Server and Debian is the second choice, the HPS device will use Debian as the Linux distribution.</p> <p><b>Webgrid in Figure 36 in Appendix A2.1</b></p>



## 7.2 Hardware Overview

The total HPS system is formed by three home components, one main unit and two socket meters. Beside the home components is a server for the services the HPS requires. The server is placed at an external company's server facility. In Figure 9 a total overview of the hardware is depicted. The HPS and the Socket Meter are connected via ZigBee (red dashed line). The legacy device (e.g. fridge, television, washing machine) is connected to the Socket Meter via the CEE 7/4 power cable (black line). The HPS is connected to the ESH Server via internet. The ESH Server can push firmware updates to the HPS device through this internet connection. On the other hand, it's also possible for a ZigBee enabled device to connect to the HPS without using the Socket Meter. Some devices only feature Wi-Fi support and not come with separate module for ZigBee. However, the ZigBee protocol also supports HomePlug and Wi-Fi as transport mechanisms to allow customers and solution providers greater flexibility [33]. In this chapter we will take a closer look at the chosen hardware.

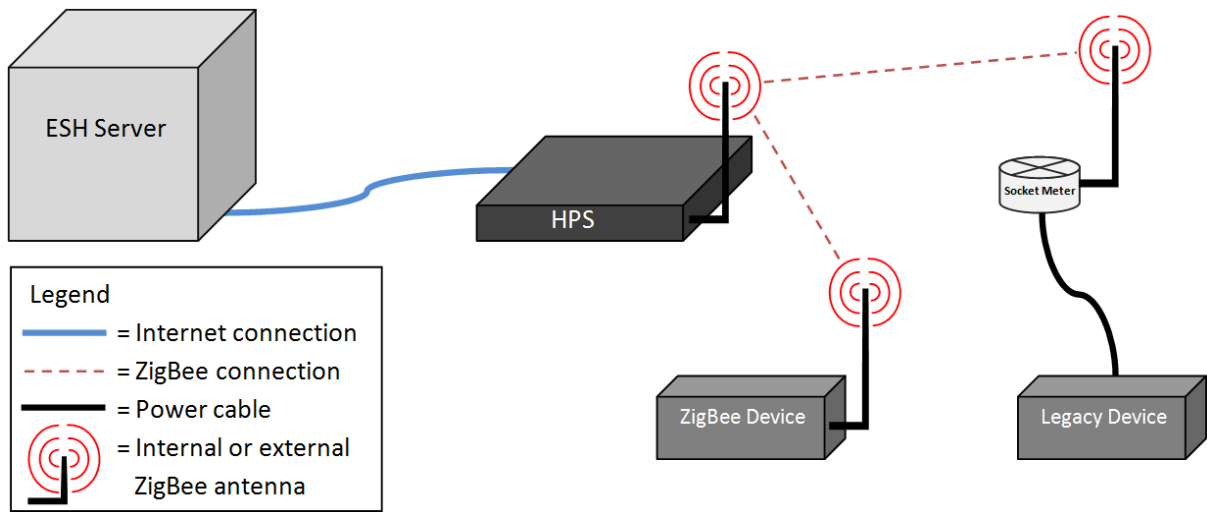


Figure 9: Hardware overview.

## 7.3 Main Device

The main device will be made up of the GuruPlug Server that already comes with a linux kernel added into the system.

### 7.3.1 Specifications

In Table 25 an overview is depicted of the hardware specifications of the main HPS device.

### 7.3.2 Storage

The HPS device fetches the energy consumption and production of each device, with a maximum of 50, every 300 seconds cf. 3.7. Each energy broadcast is 48 bit (cf. 7.4.2). 48 bit every 300 seconds is  $48 * 12 * 24 = 13.824$  bit a day. For a maximum of 50 devices to be stored for a year, a total of  $13.824 * 50 * 365 = 252.288.000$  bit. That equals a total of  $252.288.000 / 8 / 1024 = 30.796$  KByte. Each entry has to be linked to a device, and with an identifier of 32-bit per device (cf. 7.4.2), an additional  $(50 * 32) / 8 / 1024 = 0,19$  KByte is required to be stored. This means that a total of **30.796,19** KByte is required to store the energy data on

Component	Specifications
CPU	1.2 GHz operation, L1 Cache: 16K Instruction, 16K Data; L2 Cache: 256KB
Memory	DDR2 800MHz, 16-bit bus, 512MB 16bit DDR2 800MHz data rate.
Storage	NAND FLASH Controller, 8-bit bus, 512MB NAND FLASH: 4Gb, direct boot, 128-bit eFuse Memory.
Power	Power input: 100-240VAC/50-60Hz Max. 20W, DC Consumption: 5V/3.0A Max, High efficiency POL DC-DC converters
Peripherals	1x eSATA 2.0 port 3Gbps SATAII, 2x USB 2.0, 1x Internal MicroSD Socket for Optional Kernel System, 1x External MicroSD Socket and RTC w/Battery.
Connection	Wi-Fi 802.11b/g, Bluetooth and ethernet
Encryption	AES, DES and 3DES algorithms.
Authentication	SHA1 and MD5 algorithm.

Table 25: Specifications of the main HPS device.

for each year.

The HPS device also stores the energy prices of each hour (cf. Realtime Pricing), this can be stored in 64-bit. Each year will cost  $(64 * 24 * 365) / 8 / 1024 = 68$  KByte.

The maximum NAND capacity on the HPS device is 512 MByte [Table 25]. In order to store a backup of the HPS software before the update, the system should be able to store the 512 MByte first.

The HPS will store all events in log files. Storing a single entry is between 20 characters and 255 characters (maximum line length), with each character taking 7 bit (cf. ASCII size). In case a new entry is stored every second, is  $(7 * 255 * 3600 * 24 * 365) / 8 / 1024 / 1024 = 6.710,43$  MByte the total amount required for log files.

All values combined result in a total amount of  $512 + 6710,43 + 30,07 = 7252,50$  MByte. Combined with the requirement to generate and store reports on the HPS device, a 8GB SD card in the external MicroSD socket is sufficient to store all the data on.

### 7.3.3 Hardware Block Diagram

In Figure 10 is a hardware diagram depicted of the GuruPlug Server.

## 7.4 Socket Meter

The Socket Meter, depicted in Figure 11, is based on the Circle design by Plugwise [35]. With the Socket Meter the energy consumption of the connected device can be measured. The connected device can also be switched on or off. The Socket Meter stores the power consumption and power production data and transmits them to the HPS device, using the wireless ZigBee network.

### 7.4.1 Specifications

Unfortunately, there are no component specifications available of the Plugwise Circle [35]; there are only technical specifications available. The available technical specifications can be found in Table 26.

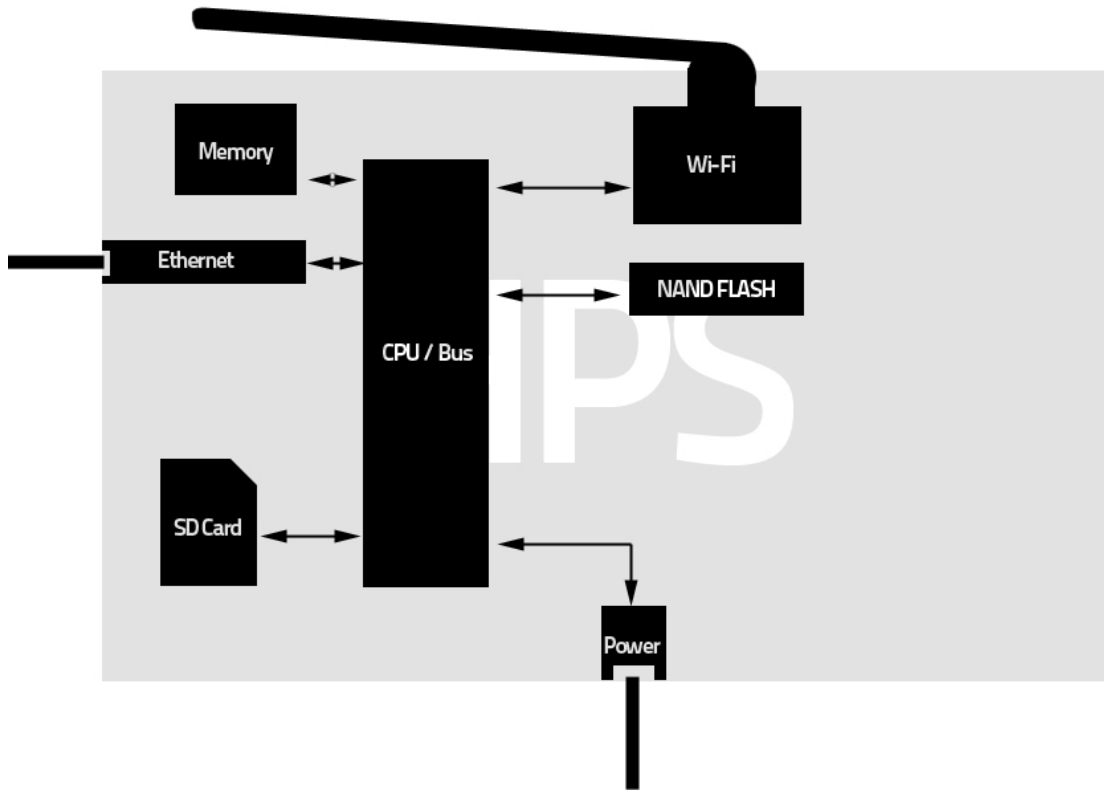


Figure 10: Hardware block diagram of the GuruPlug Server.



Figure 11: The HPS Socket Meter, based on the Plugwise Circle [35].

#### 7.4.2 Storage

The Socket Meters don't require the HPS to be connected at all times. The Socket Meter comes with 512 KByte storage capacity [35], in which it is able to store measurement data autonomously for more than a year, storing the average measurement with intervals of 1 hour [36]. Measurements are stored after that period, but earlier data are overwritten, starting with the oldest data. When the HPS is active, the Socket Meter is able to relay new statistics every 8 seconds [36]. This means that the minimum time to display new information in a Realtime overview is 8 seconds. The HPS is not only able to store the data from the Socket Meter every hour but also every 8 seconds.

Storing a single amount of power costs 24-bit, as 16-bit is not sufficient with just 65535 unique values. Since the Socket Meter has to store both the power consumption and the power production, each storage entry is:  $24 * 2 = 48$  bit. The HPS stores the data every hour, resulting in a total amount per day of:  $48 * 24 = 1152$  bit.

Component	Specifications
Connection	ZigBee. Baudrate 250 KBit/s, frequency 2400-2484 MHz, bandwidth 2.0 MHz. Power plug of type CEE 7/4.
Power	Power input: 85-253VAC / 50-60Hz. Power consumption: 0.3-1.1W, average of 0.55W.
Dimensions	81x72x44 mm.
Storage	512 KByte.
Measurement accuracy (current usage)	5% / 0,5 W.
Measurement accuracy (cumulative, 1 hour)	1% / 0,2 W.
Storage	512 KByte
Power plug	CEE 7/4 standard, unofficially known as "Type F". Compatible in The Netherlands.
Encryption	128-bit AES encryption.
Peripherals	None.

Table 26: Technical specifications of the Socket Meter.

The Socket Meter should be able to store the measurement data for at least a year, resulting in a yearly amount of:  $1536 * 365 = 420480$  bit, which results in  $420480 / 8 / 1024 = 51$  KByte, which is far below the storage capacity of 512 KByte. This means that the rest of the storage capacity on the Socket Meter can be used to store identification data (e.g. device 1) and also energy data for a longer period.

As the amount of data per measurement is so small (48 bit), the transfer rate requirement of the storage is very low. The device can be identified using a 32 bit integer, which means a total of 4.294.967.296 unique devices. The storage should be able to send the  $48 + 32 = 80$  bit out in 8 seconds, which results in a minimum transfer rate of 10 bits per second.

## 7.5 Server

The server will be used to send updates to the HPS devices. The server will be a *managed dedicated server*, which is a service that includes the hardware, software, and ongoing upkeep of a dedicated server by the external company. The external company will maintain, deploy and scale the server. There are a large number of requirements the server should abide by, so the external company has to abide to a certain Service Level Agreement. The requirements in the Service Level Agreement are described in this subsection.

### 7.5.1 Specifications

There are several specifications required for each ESH server. Each item is dedicated to a specification of the server.

#### 7.5.1.1 Operating System

The operating system of the ESH server will be CentOS (cf. 6.1). "CentOS exists to provide a free enterprise class computing platform to anyone who wishes to use it. CentOS is designed for people who need an enterprise class OS without the cost or support of the prominent North American Enterprise Linux vendor" [8].

### 7.5.1.2 Storage

The maximum NAND capacity on the HPS device is 512 MByte [Table 25], which includes the firmware and software for the HPS device. CentOS requires a minimum of 4 GByte storage space [7]. In order to maintain at least 100 updates, which is more than enough within the 5 year estimated lifespan of the server, on the ESH server, a total of **512 MByte \* 100 = 50 GByte** is required. This means that in total at least **4 GByte + 50 GByte = 54 GByte** is required on the ESH server.

### 7.5.1.3 Bandwidth

ESH plans to sell an estimated 100.000 devices [Section 2]. With a maximum of 512 MByte per update and a maximum of one update a month, the server should be able to handle at least **512 MByte \* 100.000 = 48.83 TByte** each month. With 4% TCP overhead [46] and TCP keep-alive packages of 100.000 devices every two hours of **60 Byte + 54 Byte [49] = 104 Byte**, which results in 1248 Byte per device each day on TCP keep-alive packages. For all devices would result in an additional bandwidth of **1238 Byte \* 100.000 = 0,12 GByte + 4% of 48.83 TByte = 1.95 TByte**. The total amount of bandwidth on the ESH server should be at least **48.84 TByte + 1.95 TByte = 50.78 TByte** each month.

### 7.5.1.4 Connectivity

As the server can serve up to 48.83 TByte each month, the server should be able to handle this with the connectivity. **50.78 TByte = 427.456.526 mbit**. Each month (30 days) contains a total of **1 \* 60 \* 60 \* 24 \* 30 = 2.592.000 seconds**. This means that the server should have at least **427.456.526 / 2.592.000 = 165 Mbit/s**. Since this isn't a valid connectivity (factor 10, e.g. 10 Mbit/s, 100 Mbit/s, 1 Gbit/s, and so on), the server should have 1 Gbit/s connectivity. In order to ensure that all clients get their update as quickly as possible, 10 Gbit/s would reduce the required time to push all updates massively, from 111 hours (**50.78 TByte = 417438 Gbit. 417438 / 3600 = 116 hours**) for 1 Gbit/s to 11.6 hours for 10 Gbit/s. With an increase in connectivity speed, the required speed of the RAM memory also increases as both need to be able to cope with the increased speed.

### 7.5.1.5 Memory

The software updates to the HPS device can be cached into the RAM memory of the server as there is no need to read the file in from the hard disk drive every push to a client. This means that the RAM memory should be able to have at least the size of the update (maximum of 512 MByte) and the speed of the connectivity used in the server (1 Gbit/s or 10 Gbit/s). All RAM memory with equal specifications of DDR-SDRAM PC-1600 or higher [50] has enough bandwidth (12.8 Gbit/s or higher) to match the connectivity.

### 7.5.1.6 Administration

The server will be administrated by the external company because ESH has no experience with server administration and because it is not their core competency. The advantage of this measurement is that no incompetent staff is able to access the server, which might result in problems on the server (e.g. security, reliability and performance). This means that all required changes to the server or the updates have to go through the external company first. The owners of ESH can access the server in case of emergency but the regular staff of ESH cannot access the server in any way and thus has to rely on the external company.

#### **7.5.1.7 Remote Access**

The server can only be accessed remotely by the owners of ESH. The regular staff is unable to access the server remotely and any required changes to the server have to go through the external company.

#### **7.5.1.8 Updates**

As the server is a *managed dedicated server*, all updates will be done by the external company. Critical updates requiring a system reboot have to be permitted by ESH before the installation and reboot, only updates which can be applied on-the-fly can be updated automatically by the external company.

### **7.5.2 Service Level Agreement**

In Appendix A5 are the Service Level Agreement requirements for the ESH server. This Service Level Agreement is adapted from the sample of Microsoft [44].

## 8 Software architecture

In this section is the software architecture outlined. This section starts with the architectural significant drivers. Subsequently are the architectural views outlined. After the architectural views are the logical view, data view, process view and deployment view outlined. This section ends with software design decisions.

### 8.1 Architectural Significant Drivers

The architectural significant driver for the software architecture are the key-drivers (cf. Section 4.3), the use-cases (cf. Section 4.5) and the following requirements (cf. Section 4).

SR-1.2, SR-2.1, SR-2.2, SR-3.1, SR-3.4, SR-4.2, SR-4.3, SR-6.1, SR-6.4, SR-6.5, SR-6.6, TR-1.1, TR-1.2, TR-4.1, TR-7.1, CF-6, NF-1.1, NF-1.4, NF-2.2, NF-3.1, NF-3.4, NF-5.1

### 8.2 Logical View

This view shows the structural elements, key abstractions and mechanisms that are used within HPS to realize the system's functionality. At first an overview of the components is provided. After that the main components are decomposed and desired in term of responsibilities and interfaces. In the end, the variability guide mentions parts of the software, which clarification is deferred until development/design phase.

#### 8.2.1 Primary Presentation

The Software Architecture of the HPS system has been structured by a layered-approach. The Layers Pattern has been chosen because it positively impacts exchangeability and maintainability of components. Furthermore, a general decomposition scheme has to be chosen in order to subdivide the work among the software development team.

The system has been structured according to the three-layers approach. The following coherent layers have been identified (cf. Figure 12):

**User-interface** This layer provides the central access-point for the end-user to the HPS System. It is responsible for delivering the web-interface to the browser, visualizing data and generating reports. The user-interface can access the system via the Application Facade.

**Application** The application layer does the main operation and calculations of the system. It manages the devices including the communication. It gathers load and price information as well as observing budget limits. It is also responsible for creating backups and updating the system.

**Data and OS Abstraction** Central data storage and abstraction for operations that require Operating System access.

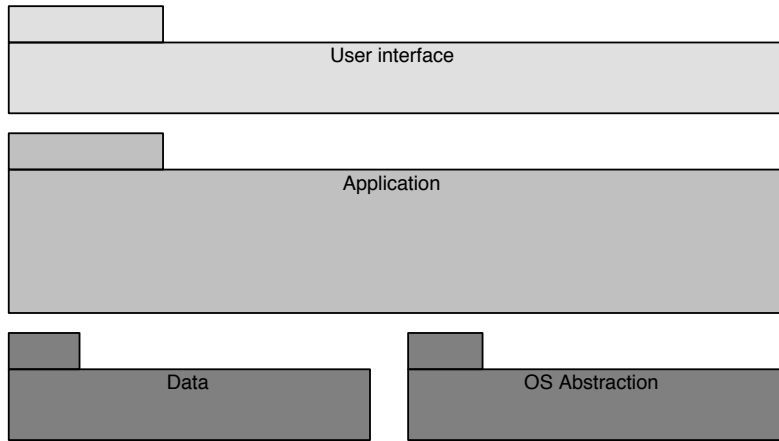


Figure 12: Layered-decomposition of the Software

The orchestration of the components within the three-tier architecture is illustrated in Figure 13. It shows that each layer is decoupled by an interface, e.g. the *IApplicationFacade*, *ISystemFacade* and *ITableGateway*.

A MessageBus is used in the Application Logic to decouple components and facilitate an adaptable and reliable architecture.

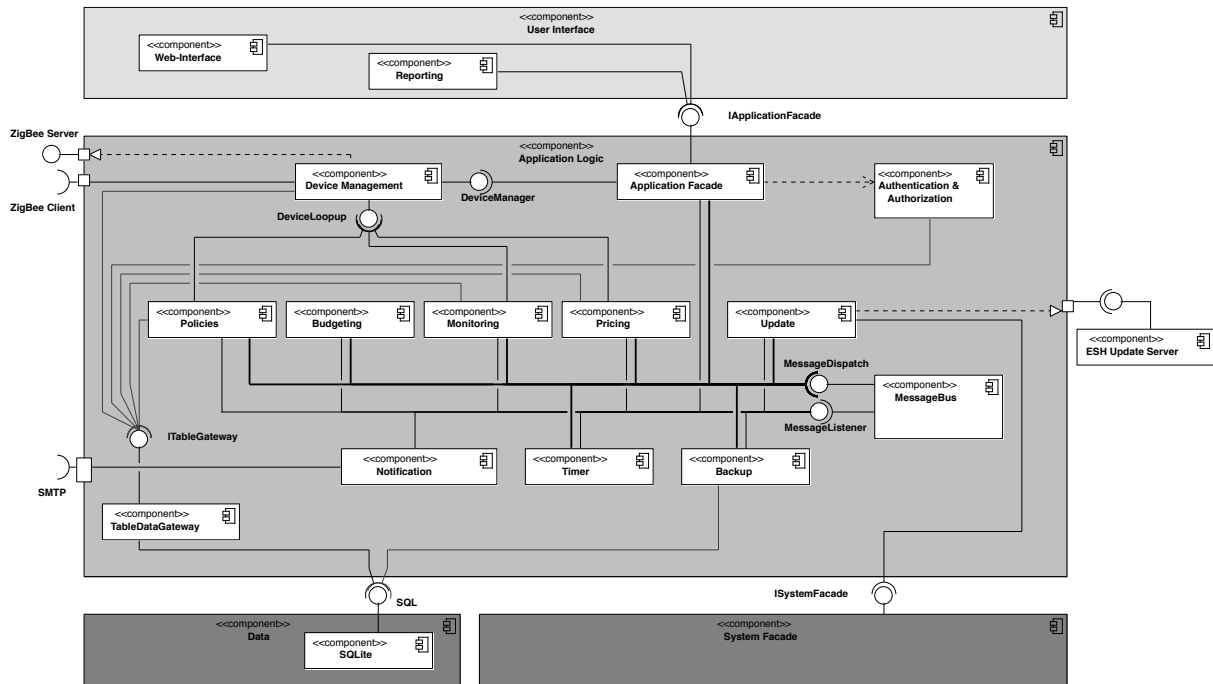


Figure 13: System Decomposition



### 8.2.2 Element Catalog

The Element Catalog describes the components mentioned in 13.

#### 8.2.2.1 MessageBus

Within the *Application Layer* the system uses a *MessageBus* [32] to decouple Components. One of the main driver of the System is the possibility to adapt to new environments. With the MessageBus new components can be easily integrated because all invocations are implicit.

Within the MessageBus, the *Publish/Subscribe*-Pattern [4] is used. Components can dispatch Messages and register themselves for all Messages or only for certain Message-types (cf. Listing 1). In order to receive Messages a component must realize the MessageListener Interface (cf. Listing 2).

In certain situations it is desirable to send a message to a particular component instead of broadcasting the message to all components. The sender field of an event can be used to reply with a directed message.

The MessageBus is also responsible for implementing *fault-detection*-mechanism by observing the messages that are sent over the Bus. The following types of faults are detected: Message that contain false content, error messages or the absence of expected messages.

Listing 1: Interface MessageDispatch

```
1 interface MessageDispatch {
2
3     dispatch( message: Message)
4     dispatchMessage( messageType: String)
5     dispatchDirectedMessage( receiver , response)
6
7     subscribe( listener : MessageListener)
8     subscribeToMessage( messageType: String , listener : MessageListener)
9 }
```

Listing 2: Interface MessageListener

```
1 interface MessageListener{
2
3     onMessage( message: Message)
4
5 }
```

#### 8.2.2.2 Database

The database will be a SQLite database. SQLite is a lightweight database that will be used to store the application data. The database will store the backup, the device management data, the policies, the energy usage data and the prices. The database will not be used to share data between the components. Instead, the MessageBus component is responsible for sharing data between components.

The Data View in Section 8.3 describes the database scheme.

The components can access the database through a *TableDataGateway* [17], which provides a programmatic interface to access the data.

To simplify and speed-up the backup, the Backup-component is the only one that should access the repository in a shared manner.

Listing 3: Interface ITableGateway

```

1 interface ITableGateway {
2     find( query : Query) : Row[]
3     delete( query : Query) : Row[]
4     update( query : Query, data)
5 }

```

### 8.2.2.3 Device Management

The DeviceManagement component is responsible for managing heterogeneous Smart Devices. This includes Socket Meters, Smart Devices and the Smart Meter. It provides interfaces for CRUD<sup>8</sup>-operations (cf. Listing 4) and lookup-operations (cf. Listing 5).

If a component lookups a device, the DeviceManagement returns an AbstractDeviceProxy, which provides an interface for the operations that can be executed on the device. The concrete communication is hidden by the *Proxy* [6].

A synchronous *Explicit Invocation* [4] is used for the concrete communication with the device. The communication itself will be coordinated by a *Broker* [6]. Depending on the device type (ZigBee Smart Energy or ZigBee Home Automation) a different Broker will be used. The Broker pattern hides the implementation details of the remote invocation of the Smart Devices services by encapsulating them into a layer other than the Proxy itself.

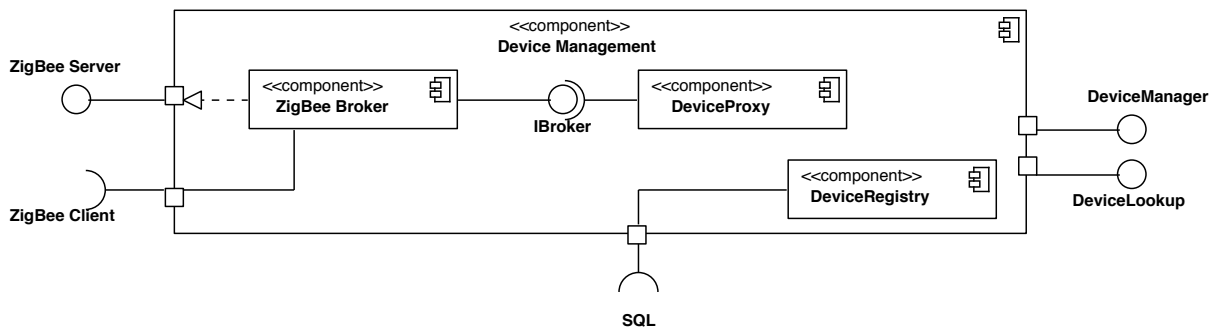


Figure 14: Device Management

Listing 4: Interface DeviceManagement

```

1 interface DeviceManagement {
2     getListOfDevices
3     findNewDevices
4     getDevice
5     addDevice
6     deleteDevice
7     editDevice
8 }

```

<sup>8</sup>Create Retrieve Update Delete

Listing 5: Interface DeviceLookup

```

1 interface DeviceLookup{
2     getAllDevice : AbstractDeviceProxy []
3     getDeviceClass : DeviceClass []
4     getDevicesByClass(class : DeviceClass) : AbstractDeviceProxy []
5     getSmartMeter : SmartMeterProxy
6 }

```

#### 8.2.2.4 Timer

The Timer is used to create periodic events within the system. A component can send the Timer a message, that it wants to receive a message at a particular time or after a certain period of time.

#### 8.2.2.5 Monitoring

The Monitoring component is responsible for gathering and persisting the consumption of the devices. It communicates with the *DeviceManagement* (cf. Figure 13) via the *DeviceLookup* interface, in order to get *DeviceProxies* (cf. Figure 21).

The DeviceProxies will be used to receive the information from the actual SmartDevices, the information will be stored into the shared repository using the *TableDataGateway* component

The actual Monitoring-process will be triggered via *TimerEvents* that are received via the *MessageBus*. Therefore the Monitoring component needs to dispatch *TimerRequestEvents* to the *MessageBus*.

In order to allow other components access to the data, the Monitoring component should dispatch or answer to Events. The Events are described in Appendix A3.

#### 8.2.2.6 Pricing

The Pricing component is responsible for gathering and persisting the energy prices. the information will be stored into the shared repository using the *TableDataGateway* pattern. It communicates with the *DeviceManagement* (cf. Figure 14) via the *DeviceLookup* interface, in order to get *SmartMeterProxy* (cf. Figure 21).

The SmartMeterProxy will be used to receive the information from the actual SmartMeter, the information will be stored into the shared repository using the *TableDataGateway* pattern.

The actual Pricing-process will be triggered via *TimerEvents* that are received via the *MessageBus*. Therefore the pricing component needs to dispatch *TimerRequestEvents* to the *MessageBus*.

To allow other components access to the pricing data, the pricing component replies to events as described in Appendix A3.

#### 8.2.2.7 Budgeting

The budgeting component is responsible for controlling the budget used in the policies. The general idea is that the end-user sets a budget for a certain task. The policies will request, on a real time basis, from the budgeting component if the budgeted tasks should be executed with this request it not only sends the request itself but also the running time. To return this data to the policies component, the budgeting component will request the current price from the pricing unit using the MessageBus interface, and then calculates if the budget is met.

#### **8.2.2.8 Policies**

The policies component is the center of the application logic, its responsibility is to enforce the policies that are set by the end-users.

So the first function of the policies component is the responsibility for persisting the policies in place, and allow for the end-user to update and extend these policies via the user-interface. The policies can be set per device type, since a washing machine has another list of functions than a light bulb. The options per device are retrieved using the DeviceLookup interface from the device manager component.

The second function is to execute and control these policies. The policies component is connected to its participating components using the MessageBus. These components, where can also be triggered on, are: Budgeting, Pricing, Monitoring, Notifications and Timer. The policies component will check every policy on triggers and then by contacting one of these components decide if it should take action.

Finally the policies component will take action either by sending a message over the MessageBus to the notification component to send a email to the end-user or using the same method contact the device management component to actively controlling smart appliances. In the later case the DeviceLookup Interface from the Device Management is used to contact the smart device and control it.

#### **8.2.2.9 Notification**

The Notification component is responsible for sending notifications via the SMTP interface. There are several templates available. It receives events from the MessageBus component. For example the Notification component receives via the MessageBus component an event from the Update component. The event contains a message that a new firmware update is installed. The Notification component selects the template corresponding to the firmware update and fill in the template with the data received with the event. Then the template is send via the SMTP interface to the email-address of the user.

#### **8.2.2.10 Web-Interface**

The web-interface component is responsible for the entire visualisation towards the end-user. To run the web-interface a web-server is implementend using Lighttpd (cf. Decision 31) that will run the html pages and the server side script. Next to the the visualisation the web-interface has the responsibility to send the options set by the user to the Application Facade component using the Application Facade Interface. The user can also change the username and password in the web-interface.

The web-interface is able to allow the end-user to add/edit/remove devices to the system. By calling the add device page from the web server the web-interface will show a list of available devices that are not yet connected to hps, again this list is recieved using the Application Facade Interface. The end-user is then able to add devices, delete devices from the hps and edit the names used.

Furthermore the web-interface is able to use the Reporting component to retrieve all the monitoring and price information from the devices. This will allow for visualisation (e.g. with charts and graphs, generated by the web-interface) of any kind of data from all of the devices or from a single device. This also allows for the user to get an accurate view of the bill (daily, /monthly/annually). Using a serverside module will allow for exporting the data to the format that the user wants, either a Comma Separated Values (Excel) or with another server side module

convert it to a pdf format. The user is also allowed to specify an interval in which it can generate reports to be sent to an emailaddress specified by the user.

It is also possible for the user to create, edit and/or remove policies from the web-interface, which uses the policies component for this purpose. To achieve this we use the Model-View-Controller architecture to construct the actual web-interface. The architecture or pattern isolates the domain logic from the user interface. This way the user interface will only be used to display and the domain logic will gather the data. This allows for concurrent development and easy extendability.

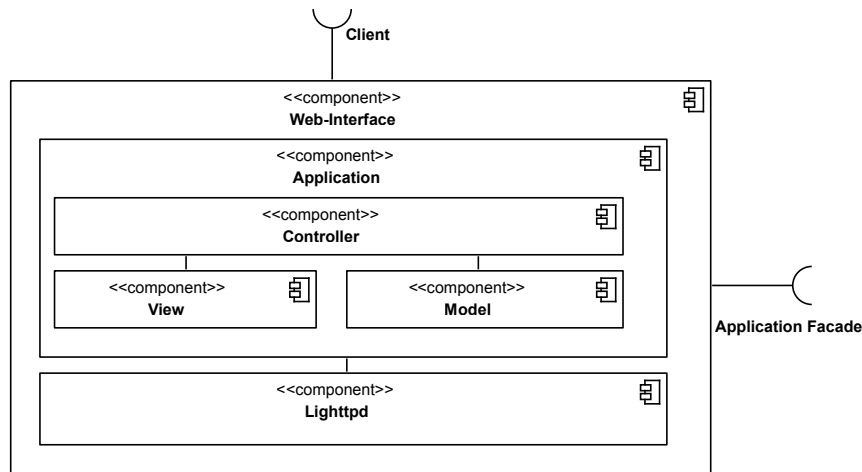


Figure 15: Update component

### 8.2.2.11 Reporting

The reporting unit is a supportive component for the Web-Interface. It can retrieve all the monitoring and pricing information by using the Application Facade Interface to connect to the facade component. This data is then transformed into usable raw data for the Web-Interface. This is done by combining all the data into day, week, month and year data views, and by combining pricing and monitoring to create a bill view. Reports can also be sent automatically on a specified interval (in a number days) with help of the Notification component to the emailaddress selected by the user (in the web-interface).

### 8.2.2.12 Update

When the HPS system starts up for the first time the Register service component tries to register itself at the ESH Update Server via the ESHUpdateServer interface. The data send to the ESHUpdateServer interface includes its ip-address, version number of the firmware, uid and the mac-address.

If there is a new update available the ESH Update Server component triggers the Push receiver component via the PushReceiver interface to get the new update. The system will make use of the notification component to rely to the user by email that a new update is available. The Push receiver component then gets the new update files via the FTP Client from the ESH Update

Server component. The connection between the ESH Update Server component and the FTP Client component is a FTP connection. Once the new update files are all downloaded the Push receiver component puts the system in maintenance mode.

Once the files are ready to be used and verified against the checksum the PushReceiver component uses the method updateProgram in the ISystemFacade interface to update the program. This interface is also responsible for creating a backup before the system starts the update, and in case the update fails, restores the backup for the system. At the end of the update will the update component use the notification component to rely to the user by email that the update has succeeded or failed.

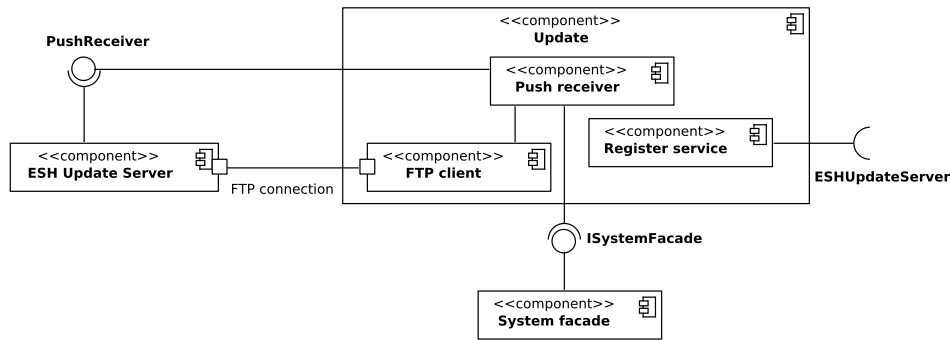


Figure 16: Update component

Listing 6: ESH Update Server interface

```

1  interface ESHUpdateServer{
2      String registerDevice( uid: String , mac: String , ip: String , version:
        String);
3      String unregisterDevice( uid: String);
4  }

```

Listing 7: Push receiver interface

```

1  interface PushReceiver{
2      String receiveUpdate( version: String , filename: String);
3  }

```

Listing 8: ISystemFacade interface

```

1  interface ISystemFacade{
2      String file( version: String , filename: String , file: File);
3      String updateProgram( filename: String , command: String);
4  }

```

### 8.2.2.13 ESH Update Server

The ESH Update Server component will push firmware updates to all the registered clients. Initially all new HPS systems register once via the register interface to the register service component. The data includes an ip-address, version number of the firmware, uid and mac-address. The register service component puts all the new registered HPS systems in the database via the SQL interface. Before performing a firmware update the Update service component asks the Monitoring component which HPS systems are online and ready to be updated. The Monitoring component gets a list of registered HPS systems from the Register service component. The Monitoring component checks via the HPS communication component every registered HPS

system if it is online and ready to receive an update. Every HPS system that is online and ready to receive an update responds with an uid, the current firmware version and the mac-address. The Update service component gets a list of available HPS systems from the Monitoring component to push a firmware upgrade via the HPS communication component to the HPS system.

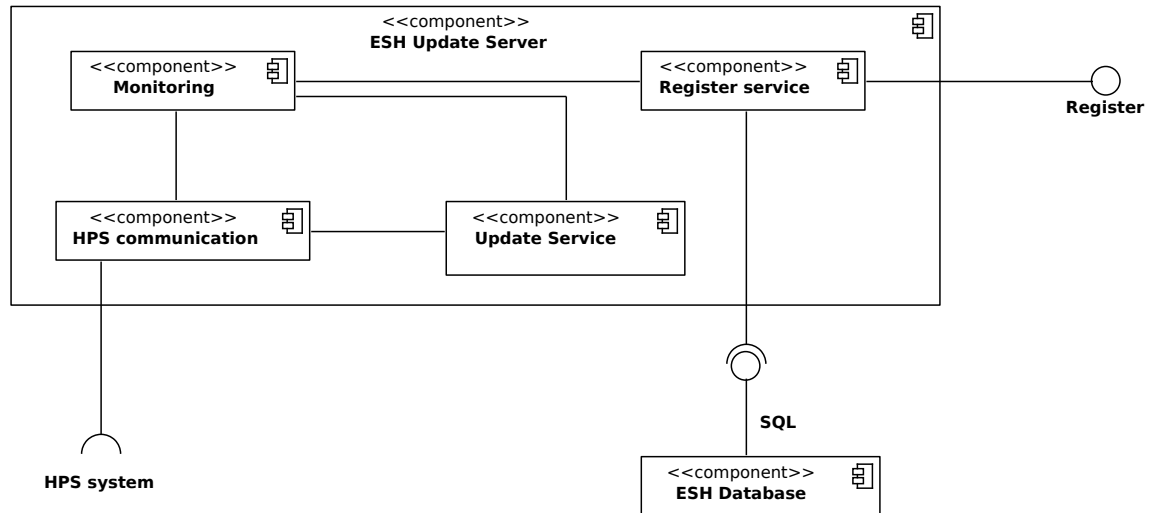


Figure 17: ESH Update Server component

Listing 9: Interface Register

```

1  interface Register{
2      String registerDevice( uid: String , mac: String , ip: String , version:
          String);
3      String unregisterDevice( uid: String);
4  }

```

#### 8.2.2.14 Backup

The backup will be stored locally on the HPS device. It will make a full backup of all user data (monitoring data, policies, settings, bills, reports, etc) and saves it on the SD-card. The backup of SQL data will be saved in a .sql file, whereas any other files will be backed up as the original copy from the HPS device.

#### 8.2.2.15 System Facade

The System Facade component is the underlying operating system. The Linux distro Debian is the operating system. This is the firmware that will be updated by the Update component. An event will be given from the Update component when a new Firmware update is available and ready to be installed. The default debian packaging format can be used. Before updating, the System Facade creates a backup of the current version. The backup includes the content of /etc, /var/lib/dpkg, /var/lib/aptitude/pkgstates and the output of dpkg --get-selections "\*". This will be archived in a tar.gz file. If something would go wrong these content can be pushed back.

#### 8.2.2.16 Authentication & Authorization

The responsibility of the *Authentication & Authorization*-component is to secure the external access to the system. It provides methods to check user credentials for validity, login as well as

logout. It also provides CRUD-methods to manage users.

The component ensures that all passwords are stored as cryptographic hash values.

It collaborates with the Application Facade and the TableDataGateway. The Application Facade uses the component to check if a call is valid or not, which means that the user is authenticated and authorized.

The *Authentication & Authorization*-component uses the *TableDataGateway* to store the Users.

### 8.2.3 Variability Guide

The described architecture is not complete, the following list gives an indication of those points within the architecture that are left unbound until a later stage of the project.

**MessageBus** The MessageBus can be used to easily integrate new components into the System.

An existing MessageBus component can be reused if appropriate. Additional effort has to be spent in order to elaborate a robust fault-detection mechanism that observes the messages.

**ZigBeeBroker** The ZigBee Smart Energy v2 is at the moment still a Draft, Therefore the Broker is modeled as Blackbox. However, if the HPS shall support other Protocols than ZigBee, they should be modeled as Brokers and implement the IBroker interface.

**Protocol Application Facade** The interface of the application facade depends a lot on the views that are realized within the user-interface. The design of the interface should be deferred until a concrete idea exists which data and format will be required by the interface.

**Localization** In the first phase of the project, the HPS is target for a market in the Netherlands. However, in later phases of the project the market will be extended, which requires localization and internationalization. For instance, support of different languages and cultural conventions like different text directions.

Within the system the *Notification* and *Web-Interface* are the components that need to be adapted. The usage of the *Model-View-Controller* within the Web-Interface helps to archive the goal. It is also desired to decouple concrete messages and logic within the *Notification*-component.



## 8.3 Data View

### 8.3.1 Events

The general message format is illustrated in Listing 10. Depending on the type of event the field message and payload are adjusted. Examples for concrete events can be found in the Appendix.

Listing 10: Protocol Event

```
1 interface Event{
2     type: integer(10)
3     sender: varchar(30)
4     timestamp: timestamp
5     message: varchar(255)
6     payload: varchar(255)
7 }
```

### 8.3.2 Database Schema

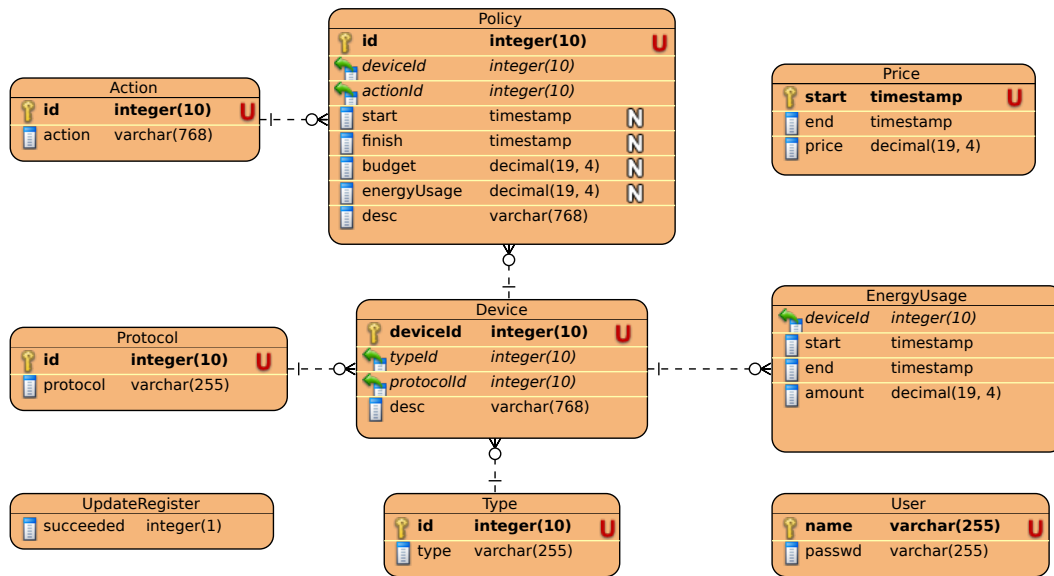


Figure 18: HPS Database

The above image displays the database structure of the HPS box. Every table will be discussed in this chapter. Starting with the Device table. The Device table is responsible for saving the connected devices which monitors the energy consumption. Every device has a unique mac-address which is in this table the private key. The type of the device is saved via the typeId. An example for a type can be wall-socket or washing-machine. The protocolId is used for which protocol the device is using. This can be for example ZigBee. Desc column is for a description of the device. Address is the ip-address of the device.

Every device is able to monitor its energy usage. This will be saved in the EnergyUsage table. The primary key is the mac address of the device. The energy usage will be saved with a start and end timestamp. This is the time in which the energy usage is monitored. Of course the amount of usage is saved in the amount column.

The Policy table is used to save the policies. Every policy has an unique id. This is also the private key. The mac column is the mac-address of the device for which the specific policy

is created. The triggerId can be for example a timetrigger. I.e. the washingmachine must be finished at 7 o'clock in the morning. For this example the finish column is set to a timestamp the next morning at 7 o'clock. If the washingmachine must be started at some given time then the start column should be filled in with a timestamp. There can also be a budget trigger. For example if the budget exceeds 20 euro then turn off the lights in the hallway to save money. A trigger can also be fired at exceeding energy usage. The amount of energy usage will be filled in in the energyUsage column. A description for the policy can be filled in in the desc column.

The energy price that is received via the smart meter will be inserted in the Price table. The starttime and the endtime of the give price will be saved. The price itself will be saved in the price column. For example: from the smart meter the price received at 12 o'clock is 0,15 euro for 1 kWh. The price changes at 23 o'clock to 0,11 euro for 1 kWh. The row to insert is start:12 o'clock(timestamp), end:23 o'clock (timestamp), price:0,15

For logging in to the user-interface of the HPS the log in credentials are saved in the User table. The username, password and email address are saved. The first time the HPS starts up it will try to register itself at the ESH update server. If it succeed the succeeded column will be set to 1.

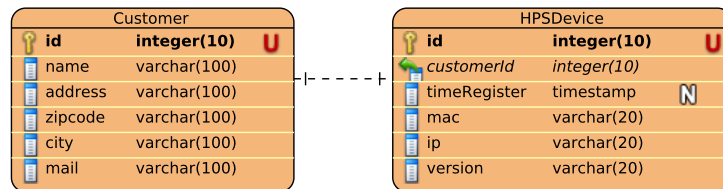


Figure 19: ESH Database

The database of the ESH update server is shown in the above image. The HPSDevice table is used for saving the HPS devices which are available for selling. Every HPS device has an unique id. Every HPS device will try to register itself at the ESH update server. Once the HPS device is registered the time when the registration took place will be inserted in the timeRegister column. The mac address of the HPS device can be retrieved from the mac column. The version number of the firmware on the HPS device will be saved in version. Via the ip address it is possible to push the firmware updates to the HPS device. Every HPS device is connected to a customer via the customerId.

The name, address, zipcode, city and mail address of the customer are saved in the customer table.

## 8.4 Process View

### 8.4.1 Monitoring diagram

In order to describe the software architecture from the process perspective, the process view is provided by modeling an activity diagram from the monitoring component, see Figure 20.

**8.4.1.1 General** The system decomposition consists of four components that will have a continuous process running. These components are Policies, Budgeting, Monitoring and Pricing. To get into detail the Monitoring process depicted to give some information and explanation how these processes will run.

**8.4.1.2 Activity Diagram** Figure 20 shows illustrates the monitoring process and the interaction with other process. In this diagram the Message Bus is illustrated as a partition/swimlane to highlight the abstraction that is created with the architecture.

The System process will start all processes that has to be started to get the HPS working. In this case the Monitoring process and the Timer process are started by the System process. *All the messages between the swimlanes are handled by the Message Bus, this is modeled in section 8.4.2.2.* The arrows via these Message Bus are deliberately interrupted and not placed as a straight line. That is because of the architecture, it is not a direct connection, it is surrendered to the Message Bus. This is the part where there might be some potential performance issues. Above the arrows that are going *to* the the Message Bus, the name of the destination process is stated. The arrow that are going *out* of the Message Bus, the source/sender process is stated.

There is also some error mechanism in the HPS and this activity diagram shows this. Every process has some fault detection to catch an unexpected interrupt. When this happens the process will handle this by trying to correctly stop itself (e.g. deallocate/free resources etc.) The system process has an extra functionality that if this process is interrupted, it sends messages through the Message Bus to all running processes that the process has to stop.

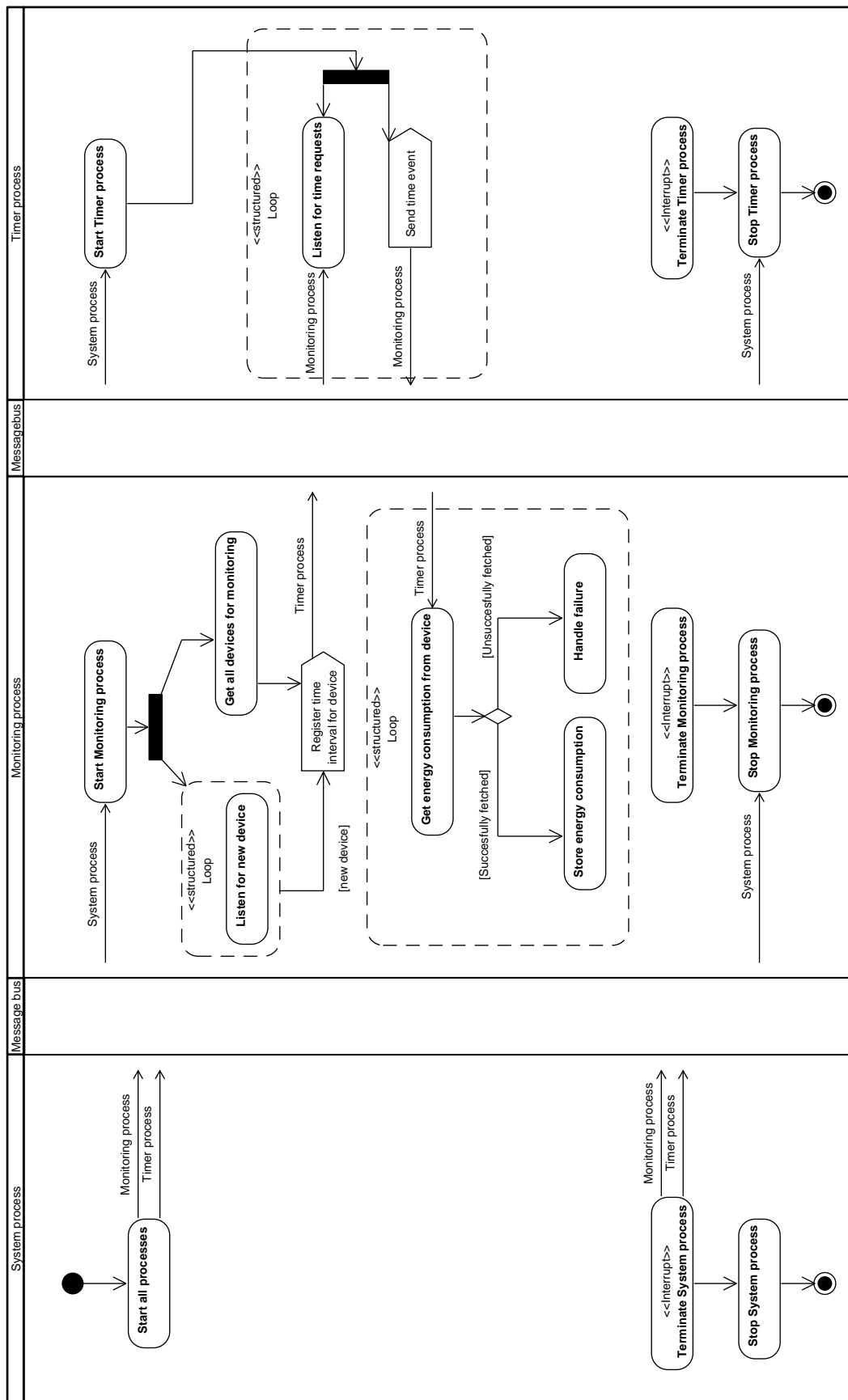


Figure 20: Activity diagram monitoring process

## 8.4.2 Element Catalog

### 8.4.2.1 Request Energy Consumption

The energy consumption is requested by the *Monitoring*-component from the Smart Meter. The Smart Meter is a ZigBee Smart Energy v2.0 device and implements the load-service as described by [3]. Compared to other ZigBee Application profiles, Smart Energy v2.0 adds an IP-based Layer to the ZigBee network stack. Furthermore, it uses HTTP to communicate with the REST-based services of the devices. Data of request and responses will be serialized as XML.

As illustrated in Figure 21 the *Monitoring* request the *DeviceProxy* for the Smart Meter from the *DeviceManagement*, which itself request the Proxy from its *DeviceRegistry*. The SmartMeter DeviceProxy provides methods to request the energy consumption. If this message is called, the invocation will be marshaled and routed to the actual Smart Meter by the *ZigBee Broker*. Since a synchronous Explicit Invocation is used the Monitoring has to wait until a response is send back or a *Timeout-Exception* is thrown.

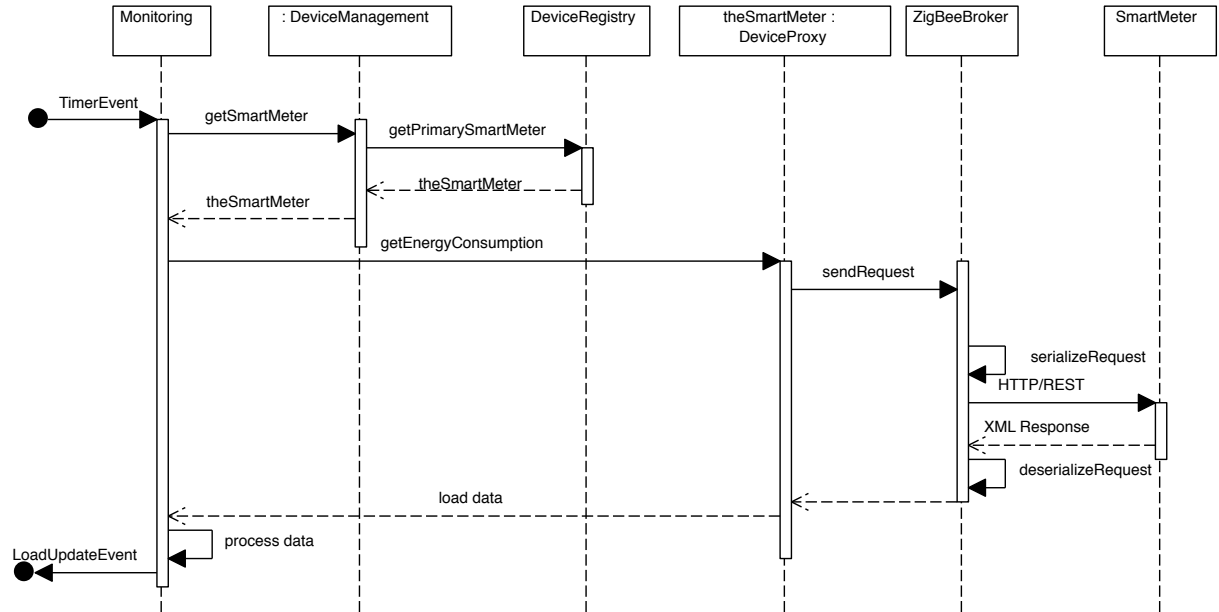


Figure 21: Process Device Communication

### 8.4.2.2 Message Bus

As illustrated in Figure 22 the *MessageBus* component is responsible for dispatching Message to all subscribed components.

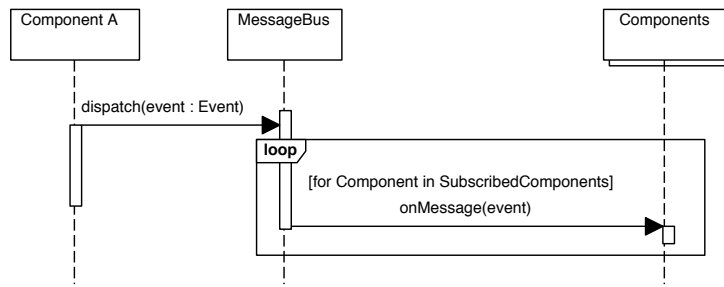


Figure 22: Publish messages via MessageBus

Even during a direct-reply Message, the *MessageBus* acts as an mediator to route the message to the correct component, to optimize communication and for fault-detection, as illustrated in Figure 23.

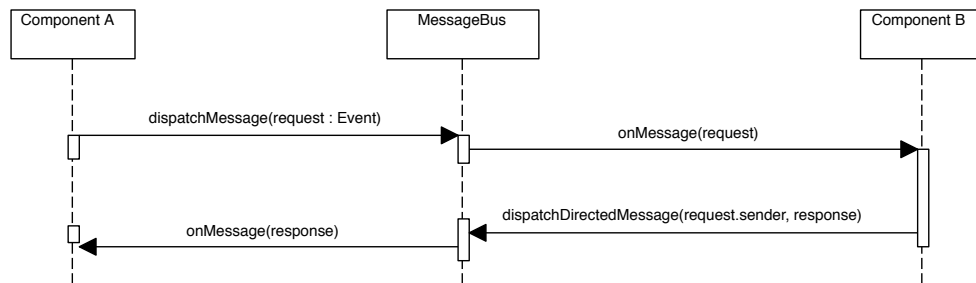


Figure 23: Responding to messages

### 8.4.3 Policies

The following process is a demonstration of the policy component that is handling a budget trigger.

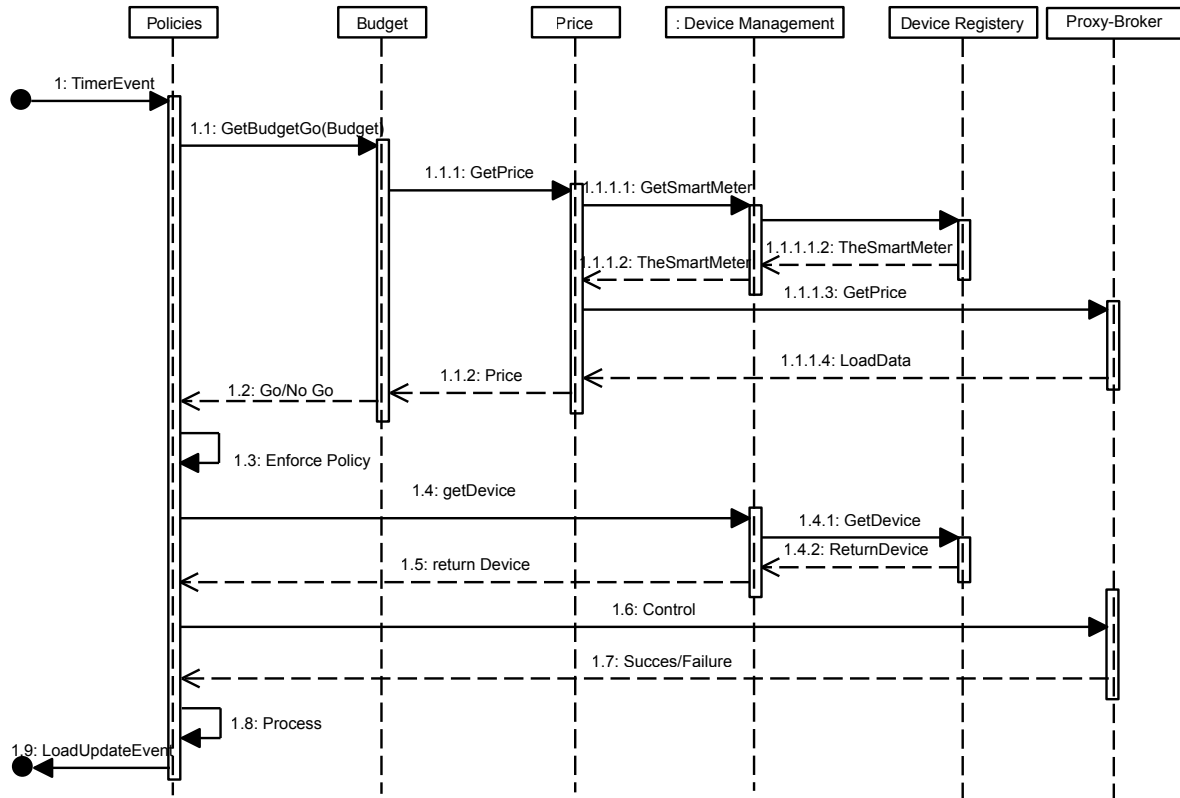


Figure 24: Policy response to a budget trigger.

The process start by a trigger from outside the policies component (TimerEvent). After the trigger is initiated the policy first needs to know if the event can run within the budget. Therefore it contacts the budget component to receive either a true or false. The budget component will use the pricing component to get the current price so it can calculate if the events get the go ahead. The pricing component will get the price via the smart meter (See 8.4.2.2 Request Energy Consumption for the proxy broker sequence).

After the go ahead is given it needs to get the device proxy of the device linked to the policy. This is done by contacting the device management using the device lookup interface.

When in contact the policies component will communicate to start its event. A response is communicated back from the device in order to tell the policies component that a action was actually performed or not. The policies component will then update its data set accordingly.

## 8.5 Deployment View

In a deployment view is shown how and where the system is deployed. The deployment view shows the relationships between the software and hardware components in the system and the physical distribution of the processing [25]. In this chapter will a deployment diagram be used represent the system. The artifacts in the deployment diagram will also be described in this chapter.

### 8.5.1 Deployment Diagram

In Figure 37 in the Appendix is the deployment diagram depicted of the HPS system.

The deployment view is separated in two parts, the *client home* which contains the HPS device and the router, and the *third party* which contains the ESH server. In the HPS and ESH server are all connections (e.g. associations, dependencies and manifestations) between artifacts within the application hidden to improve readability. All connections between the artifacts can be found in Figure 13 and 17. The hardware specifications of the HPS and ESH server are based on chapter 6.

#### 8.5.1.1 Client Home

The HPS consists of an *execution environment*, which is a type of node that represents a particular execution platform, such as an operating system [25]. The execution environment on the HPS is an operating system, namely Debian. The execution environment Debian consists besides the application *esh\_hps* of two other components, a *lighttpd* web server and a *SQLite* database. There is a connection from the webinterface in the *esh\_hps* application to the *lighttpd* web server in order to serve the websites to the user. The shared repository in the *esh\_hps* application on the other hand requires a connection to the *SQLite* database to store all required data in. The *client home* consists besides the HPS also about a router, which is used to connect the HPS to the internet, and thus to the ESH server.

#### 8.5.1.2 ESH Server

The ESH server also consists of an operating system execution environment, namely CentOS. The execution environment consists besides the application *esh\_server* of one other component, a *SQLite* database. The register in the *esh\_server* application has a connection to the *SQLite* database in order to store all required data. The ESH server has a direct connection to the internet, which is used to connect to the HPS devices.

### 8.5.2 Artifacts

The artifacts in Figure 37 are based on the system decomposition from Figure 13 and 17. Each component is represented in the deployment diagram as an artifact. A separation has been made between the HPS device and the ESH server.

#### 8.5.2.1 HPS

In Table 27 is a description depicted for each artifact of the HPS from Figure 37. The HPS device is split up in three layers, the *user-interface*, the *application* and the *data and OS abstraction*. In Figure 13 is a separation depicted where each artifact belongs to a specific layer. The artifacts are created around layers because it positively impacts the exchangeability and maintainability of these artifacts.



<b>Artifact</b>	<b>Description</b>
WebInterface	The web-interface is responsible for the entire visualisation towards the end-user.
Reporting	Reporting is responsible for the generation of user-generated reports (e.g. billing) towards the end-user.
DeviceManagement	DeviceManagement is responsible for managing heterogeneous Smart Devices. This includes Socket Meters, Smart Devices and the Smart Meter
AppFacade	The AppFacade is responsible for the communication between the user interface and the application logic.
Authentication	Authentication is responsible for the verification of the login of the end-user.
Policies	Policies is responsible for creating and executing policies for the end-user. The policies will be executed based on the configuration by the end-user.
Budgeting	Budgeting is responsible for controlling the budget used in the policies. The policies will request, on a real time bases, from the budgetting component if the budgeted tasks should be executed.
Monitoring	Monitoring is responsible for gathering and persisting the consumption of the devices. It communicates with DeviceManagement.
Pricing	Pricing is responsible for fetching the current prices from the Smart Meter which will be used by the other artifacts.
Update	Update is responsible for registering with the ESH server in order to receive updates for the HPS device. The updates will be transported via the FTP protocol.
MessageBus	The MessageBus is responsible for sharing data between artifacts.
Notification	Notification is responsible for sending notifications via the SMTP interface.
Timer	The Timer is responsible to create periodic events within the system. A component can send the Timer a message, that it wants to receive an event at a particular time or after a certain period of time.
Backup	Backup is responsible for creating (periodic) backups from the HPS device.
TableDataGateway	The TableDataGateway is responsible for providing a programmatic interface to access the data..
SystemFacade	The System Facade is the underlying operating system, namely Debian.

Table 27: Artifact descriptions for the HPS device.

### 8.5.2.2 ESH Server

In Table 28 is a description depicted for each artifact of the ESH server from Figure 37.

Artifact	Description
Monitoring	Monitoring is responsible for checking the availability of a HPS device before the update is sent.
Register	Registration is responsible for the initial registration for a new HPS device.
Communication	Communication is responsible for setting up a connection between the HPS device and the ESH server.
Update	Update is responsible for sending the updates to the HPS devices.

Table 28: Artifact descriptions for the ESH server.

## 8.6 Software Design Decisions

This section highlights some of the design decisions that have been made, especially those that are not mentioned in the previous views.

Table 29: Decision - SQLite

Name	SQLite
Decision	29
Status	<i>Approved</i>
Problem/Issue	Data needs to be stored persistently and accessed from various components.
Decision	The choice for data storage for the HPS device is SQLite.
Alternatives	<i>MySQL</i> Reliable and open-source RDBMS, which is often used for web-applications <i>MongoDB</i> NoSQL document database. High scalability.
Arguments	Because an SQLite database requires little or no administration, SQLite is a good choice for devices or services that must work unattended and without human support. SQLite is a good fit for use in cellphones, PDAs, set-top boxes, and/or appliances. It also works well as an embedded database in downloadable consumer applications. MySQL has been considered to be overweighted and resource consuming. MongoDB is not schema-based but data consistency is required.

Table 30: Decision - Message Bus

Name	Message Bus
Decision	30
Status	<i>Approved</i>

<b>Problem/Issue</b>	Components within the application logic are highly interconnected. In order to improve adaptability the components need to be decoupled.
<b>Decision</b>	Use a MessageBus to decouple components
<b>Alternatives</b>	<p><i>Blackboard</i></p> <p>Producer publish information on the blackboard. Consumer monitor the blackboard and use the information provided on the blackboard to solve their problem. Access to the blackboard is managed via a Moderator.</p>
<b>Arguments</b>	<p>The MessageBus has been favored over the Blackboard because it is perceived to be easier to implement. Furthermore, a MessageBus strongly improves modifiability. Each component has a single connection to the MessageBus instead of multiple connections to each of the other. Adding or removing a component is easier and has a lower impact on the existing system. Furthermore, it reduces application complexity in terms of interconnections between components. A clearly defined protocol is used. Last but not least, it provides the possibility to perform extensive fault-detection, since messages and thus the state of the system is observable.</p>

Table 31: Decision - Web-server

<b>Name</b>	<b>Web-server</b>
<b>Decision</b>	<b>31</b>
<b>Status</b>	<i>Approved</i>
<b>Problem/Issue</b>	A web-server has to be chosen to host the web-interface on.
<b>Decision</b>	The choice for the web-server is Lighttpd.
<b>Alternatives</b>	<p><i>Cherokee</i></p> <p>Good performance, large amount of features, easy to configure.</p> <p><i>Apache</i></p> <p>Average performance, large amount of features, easy to configure.</p> <p><i>nginx</i></p> <p>High performance, average amount of features, high concurrency, low memory usage, average to configure.</p> <p><i>Lighttpd</i></p> <p>Best performance, average amount of features, secure, flexible, low memory and CPU usage, average to configure.</p> <p><i>Sources used: [55]</i></p>

<b>Arguments</b>	The alternative chosen is Lighttpd. There is a lot of similarity between the results from Lighttpd and nginx, they both have excellent performance and low processor and memory footprint. The main reason Lighttpd is chosen is because it has slightly lower memory and CPU usage and is more secure than nginx. Another advantage of using Lighttpd is that the performance is slightly better than nginx. Apache and Cherokee both have a high memory and CPU usage and, given the importance due to hardware limitations, are not considered as an alternative for the Lighttpd web-server.
------------------	--

Table 32: Decision - Programming Language

<b>Name</b>	<b>Programming Language</b>
<b>Decision</b>	<b>32</b>
<b>Status</b>	<i>Approved</i>
<b>Problem/Issue</b>	A programming language has to be chosen to develop the HPS device software and ESH server software with.
<b>Decision</b>	The choice for the programming language is C++.
<b>Alternatives</b>	<p><i>C++</i> High performance, hard to code, huge availability of libraries, low memory and CPU usage, cross-platform</p> <p><i>Java</i> Mediocre performance, average to code, average availability of libraries, high memory and CPU usage, cross-platform</p> <p><i>Python</i> Low performance, easy to code, average availability of libraries, average memory and CPU usage, cross-platform</p> <p><i>Sources used: [41]</i></p>
<b>Arguments</b>	The chosen programming language to develop both the HPS device software and ESH server software in is C++. Even though it is harder to code in C++, it has less memory and CPU usage, making it a perfect match for the hardware specifications of the HPS device. The huge availability of libraries for C++ is a great advantage for C++ as it requires less code to be developed by ESH, which can result in earlier release, more features or a more bug-free product. The rest of the specifications (cross-platform and performance) of the languages are comparable, although the performance of C++ is a lot better than Java and Python, making more actions possible.

Table 33: Decision - Firewall

<b>Name</b>	<b>Firewall</b>
-------------	-----------------

<b>Decision</b>	<b>33</b>
<b>Status</b>	<i>Approved</i>
<b>Problem/Issue</b>	A firewall has to be installed on the HPS device to ensure only access to the ESH server and to limit access from intruders.
<b>Decision</b>	The chosen firewall for the HPS device is m0n0wall.
<b>Alternatives</b>	<p><i>ClarkConnect</i> High memory and CPU usage, large amount of features, hard to configure, requires a lot of disk space</p> <p><i>IPCop</i> Low memory and CPU usage, average amount of default features but huge community for adding features with plugins, hard to configure</p> <p><i>m0n0wall</i> Lowest memory and CPU usage, large amount of features, easy to configure, requires almost no disk space, hard to add features</p> <p><i>pfSense</i> Low memory and CPU usage, limited amount of default features, hard to configure, based on m0n0wall</p> <p><i>SmoothWall</i> Average memory and CPU usage, large amount of features, easy to configure</p> <p><i>Sources used: [18]</i></p>
<b>Arguments</b>	The firewall best suitable for the HPS is m0n0wall. This firewall is best adapted for the lower hardware specifications of the HPS device, as it has the lowest memory and CPU usage compared to the alternatives. Even though it is hard to add new features due to limited plugin support, it requires almost no disk space, and has a large amount of features and is easy to configure. The best alternative for m0n0wall is IPCop, because it also has a low memory and CPU usage (although higher) and has a decent amount of features available. pfSense is not considered a worthy alternative due to the very limited amount of features. ClarkConnect and SmoothWall require too much memory and CPU usage on the limited hardware specifications of the HPS device.

## 9 Architecture Verification

In this chapter is the architecture verified by requirement verification and architecture evaluation. In the first subsection are all requirements from chapter 3 checked to see if they are implemented in the system. The second subsection uses ATAM (Architecture Tradeoff Analysis Method) for the architecture evaluation. This chapter finishes with conclusions and improvements.

### 9.1 Requirements verification

This subsection is split up in the same way as the requirements are depicted in chapter 3. The verification of the requirements is performed by using traceability matrix. For the chapter traceability is in several cases the referenced point for the chapter described between round brackets to enhance the readability of the sections. The information between the round brackets refers to a subsection of the chapter.

#### 9.1.1 Functional Requirements

In Table 34 the verification is depicted of the functional requirements. Each ID and priority refers to the original requirement as found in chapter 4.6.

ID	Priority	Decision	Fulfilled	Chapter ability	Trace-	Remarks
SR-1.1	Must	14, 22	Yes	7.2, 7.3 (Specifica- tions), 7.4 (Specifica- tions)		Communication between the devices is done by ZigBee protocol
SR-1.2	Must	-	Yes	5.1, 7.4 (Storage), 8.2 (Monitoring)		Monitoring data is done on a regular interval, which will be stored on the HPS
SR-1.3	Must	16, 29	Yes	8.2 (Monitoring), 8.2 (Database)		Stored with help from TableDataGateway
SR-1.4	Must	17	Yes	8.2 (Web-Interface), 8.2 (Device Manage- ment)		Provided by the web- interface, which uses the device management component
SR-1.5	Optional	-	No	-		Implemented in the fu- ture
SR-2.1	Must	16, 29	Yes	4.1, 5.1, 8.2 (Pric- ing), 8.4 (Request Energy Consump- tion)		Prices are supplied to the Smart Meter by the Utility
SR-2.2	Must	17	Yes	8.2 (Reporting), 8.2 (Pricing), 8.2 (Web- Interface)		Provided by the web- interface
SR-2.3	Must	-	Yes	5.1, 8.2 (Reporting), 8.2 (Pricing), 8.2 (Monitoring)		On request of the user, done by the web-interface

SR-2.4	Must	-	Yes	8.2 (Reporting), 8.2 (Pricing)	Not automatically, customer has to generate the report manually
SR-2.5	Optional	17	Yes	8.2 (Reporting), 8.2 (Pricing), 8.2 (Web-Interface)	Only when storage space is sufficient
SR-3.1	Must	16, 17	Yes	8.2 (Reporting), 8.2 (Web-Interface)	Provided by the web-interface, requires user-interaction
SR-3.2	Must	17	Yes	8.2 (Reporting), 8.2 (Web-Interface), 8.2 (Billing), 8.2 (Device Management)	Provided by the web-interface, requires user-interaction
SR-3.3	Must	17	Yes	8.2 (Reporting), 8.2 (Web-Interface)	Provided by the web-interface, based on data from reporting
SR-3.4	Must	17	Yes	5.1, 8.2 (Reporting), 8.2 (Notification), 8.2 (Web-Interface)	Uses all three components in order to send reports automatically by email
SR-4.1	Must	17	Yes	5.1, 8.2 (Policies), 8.2 (Web-Interface)	Provided by the web-interface, requires user-interaction
SR-4.2	Must	-	Yes	8.2 (Policies)	The amount of different devices is limited to Zig-Bee's support
SR-4.3	Must	-	Yes	5.1, 8.2 (Policies), 8.2 (Budgeting), 8.2 (Monitoring)	Uses all three components to provide this requirement
SR-5.1	Must	17	Yes	8.2 (Authentication & Authorization), 8.2 (Database), 8.2 (TableData-Gateway), 8.2 (Web-Interface)	Uses all four components to secure external access from the system
SR-5.2	Must	17	Yes	8.2 (Web-Interface)	Provided by the web-interface, requires user-interaction
SR-6.1	Must	18	Yes	5.1, 7.5, 8.2 (Update), 8.2 (ESH Update Server)	Updates functionality is provided by internal modules and the external update server
SR-6.2	Must	18	Yes	5.1, 8.2 (Update), 8.2 (Notification)	User is informed by email

SR-6.3	Must	-	Yes	3.1, 4.1	Is an assumption about the customer.
SR-6.4	Must	16, 21	Yes	8.2 (Backup), 8.2 (Database), 8.2 (System Facade)	Backup functionality is provided using all these components
SR-6.5	Must	-	Yes	8.2 (Backup), 8.2 (Update)	The update components uses the backup component to initiate backup and/or restore
SR-6.6	Must	18	Yes	4.3, 5.1, 7.5, 8.2 (Update), 8.2 (ESH Update Server)	As specified in the hardware specifications, however, it can also be spread across multiple servers
SR-7.1	Optional	17	No	-	Implemented in the future
SR-7.2	Optional	17	Yes	8.2 (Web-Interface), 8.2 (Reporting)	Provided by the web-interface, requires user-interaction

Table 34: Functional requirements verification.

### 9.1.2 Technical Requirements

In Table 35 is the verification depicted of the technical requirements. Each ID and priority refers to the original requirement as found in chapter 4.7.

ID	Priority	Decision	Fulfilled	Chapter ability	Trace-	Remarks
TR-1.1	Must	14, 22	Yes	4.3, 7.3, 7.4		All devices communicate using the ZigBee protocol
TR-1.2	Must	-	Yes	8.2 (Timer), 8.2 (Monitoring)		To fulfill this should the timer be set to 300 seconds in system configuration
TR-1.3	Must	-	Yes	7.3 (Storage)		In case a limit is specified in the web-interface, other components have no strict limit
TR-1.4	Must	14	Yes	4.3, 8.2 (Device Management)		All devices communicate using the ZigBee protocol



TR-3.1	Must	-	Yes	8.2 (Reporting), 8.2 (Web-Interface)	Provided by the web-interface, requires user-interaction
TR-4.1	Must	-	Yes	3.4, 8.2 (Timer), 8.2 (Budgeting), 8.2 (Pricing), 8.2 (Web-Interface), 8.2 (Policies)	The triggers are part of the policies
TR-5.1	Must	17	Unknown	8.2 (Web-Interface)	Expected, not yet testable
TR-6.1	Must	20	Yes	7.3 (Specifications)	This is a default component in the hardware specifications
TR-7.1	Must	17	No	-	Implemented when SR-7.1 is
TR-7.2	Must	-	Yes	8.2 (System Facade)	The modes are triggered by the update event
TR-7.3	Optional	-	No	8.2 (System Facade)	Implemented in the future
TR-7.4	Optional	-	Unknown	8.2 (System Facade), 8.2 (Web-Interface)	To be decided by developer

Table 35: Technical requirements verification.

### 9.1.3 Commercial Requirements

In Table 36 is the verification depicted of the commercial non-functional requirements. Each ID and priority refers to the original requirement as found in chapter 4.7.

ID	Priority	Decision	Fulfilled	Chapter ability	Trace-	Remarks
CF-1	Must	20	No	7.1, 7.3		No dimensions and colors specified
CF-2	Must	20, 22	Yes	2.8 (Hardware costs)		No device exceeds threshold
CF-3	Must	20, 22	Unknown	2.8 (Product costs)		Production costs for the HPS set (including 2 socket meter) already exceeds €300. Final sale price not decided
CF-4	Must	-	No	-		No manual written

CF-5	Must	-	Unknown	-	Expected, not yet testable
CF-6	Must	17	Yes	8.2 (Web-Interface)	Fulfilled by a modular design
CF-7	Must	20, 22	Yes	5.2, 6	Only with a 99.5% availability

Table 36: Commercial non-functional requirements verification.

#### 9.1.4 Technical Non-Functional Requirements

In Table 37 is the verification depicted of the technical non-functional requirements. Each ID and priority refers to the original requirement as found in chapter 4.7.

ID	Priority	Decision	Fulfilled	Chapter ability	Trace-	Remarks
NF-1.1	Must	20, 22	No	5.2, 6		Only with a 99.5% availability
NF-1.2	Must	-	Unknown	8.2 (Update), 8.2 (System Facade)	8.2	Expected, not yet testable
NF-1.3	Must	-	Unknown	8.2 (Update), 8.2 (System Facade), 8.2 (Database), 8.2 (Backup)	8.2	Expected, not yet testable
NF-1.4	Must	29	Yes	8.2 (Database)		Fulfilled by the database design and implementation
NF-2.1	Must	17	Unknown	8.2 (Web-Interface), Manual		Expected, not yet testable
NF-2.2	Must	17	Unknown	8.2 (Web-Interface)		Expected, not yet testable
NF-2.3	Must	17	Unknown	8.2 (Web-Interface)		Expected, not yet testable
NF-2.4	Must	17	Unknown	8.2 (Web-Interface)		Expected, not yet testable
NF-2.5	Must	17	Unknown	8.2 (Web-Interface)		Expected, not yet testable
NF-2.6	Must	17	Unknown	8.2 (Web-Interface)		Expected, not yet testable
NF-2.7	Must	17	Unknown	8.2 (Web-Interface)		Expected, not yet testable

NF-2.8	Must	17	Unknown	8.2 (Web-Interface)	Expected, not yet testable
NF-3.1	Must	32	Yes	8 (Views)	Fulfilled by a modular design
NF-3.2	Must	14	Yes	8.2 (Device Management)	Fulfilled by the ZigBee module
NF-3.3	Optional	-	Unknown	7	Expected, not yet testable
NF-3.4	Must	-	Unknown	7	Expected, not yet testable
NF-4.1	Must	16, 21	Yes	8.2 (Backup), 8.2 (Database)	Backup should be set daily in the system configuration
NF-4.2	Optional	16, 21	Yes	8.2 (Backup), 8.2 (Database)	The SD card contains all data, and can also be transferred to another device
NF-4.3	Must	22, 29	Partly	7.4, 8.2 (Monitoring), 8.2 (Database)	No downtime, but no logging of the event
NF-4.4	Must	22, 14, 20	Yes	4.3, 7	Fulfilled by the ZigBee module
NF-5.1	Must	33	Yes	8.7 (Firewall)	Has to be specified in the firewall configuration
NF-5.2	Must	18	Yes	8.2 (Updates)	Provided by the update component
NF-5.3	Must	20	Partly	7.3	Only MD5 and SHA1 are supported (based on the hardware specifications)
NF-5.4	Must	17	Unknown	8.2 (Web-Interface)	Expected, not yet testable
NF-5.5	Must	17	Unknown	8.2 (Web-Interface), 8.2 (Authentication & Authorization)	Expected, not yet testable
NF-5.6	Optional	-	No	-	Implemented in the future
NF-5.7	Optional	17, 29	No	8.2 (Web-Interface), 8.2 (Authentication & Authorization)	Implemented in the future
NF-6.1	Must	14, 20	Yes	4.3, 7.3, 7.4	Fulfilled by design decision
NF-6.2	Must	17	Unknown	8.2 (Web-Interface)	Expected, not yet testable

NF-6.3	Must	14	Yes	5.3, 8.2 (Device Management)	Fulfilled by design decision
--------	------	----	-----	------------------------------	------------------------------

Table 37: Technical non-functional requirements verification.

## 9.2 Architecture Evaluation

To evaluate the architecture, as seen in the previous chapters, the ATAM (Architecture Trade-off Analysis Method) is used. This evaluation method is a structured approach to assessing how well a system is likely to meet its required, based on the characteristics of its architecture. Furthermore it identifies critical design decisions (architectural approaches), verifies them against key-drivers (low-level scenarios), considers multiple Quality Attributes & different views, discovers sensitivity points (Decisions for which a slight change makes a big difference in a Quality Attribute), discovers critical architectural tradeoffs (Decisions affecting more than one Quality Attribute) and discovers risks (Decisions that might create (future) problems for some Quality Attributes).

For the ATAM the following steps have been taken.

1. Present the ATAM: Moderator
2. Present business drivers: Business stakeholder
3. Present architecture: Architect
4. Identify architectural approaches: Architect
5. Generate quality attribute utility tree: Team
6. Analyze architectural approaches: Architect
7. Brainstorm & prioritize scenarios: Team
8. Analyze architectural approaches: Architect
9. Present results: Moderator

The results of the ATAM can be viewed in the remainder of the chapter. Not all steps are documented, the output are the: architectural approaches, utility tree, low-level scenarios, sensitivity points, architectural tradeoffs and risks and non-risks.

### 9.2.1 Architectural Approaches

This section is used to discuss the architectural approaches and styles used within the document. The results of listing the identifiable approaches with explanation is found below.

The goal, from the start, has been to build a product that will allow the end-user to save energy and therefor also money on the energy bill. The system does this by either informing the end-user for example by giving a estimate of the monthly bill, display total, per device energy consumption on any timeline and allow for the end-user to control the behaviour of smart devices by setting policies. The rational behind the decision is that the industry is currently only

exploring smart devices and the smart grid but had not come to any form of agreement and implementation is slow. Since the goal is to introduce the product in the year 2013 it must be a useful tool that really has value for consumers.

One of the most important details of the product is the communication between the smart devices and any other device requiring energy in the household. The decision is made to use ZigBee: Smart Energy as a protocol to allow for this communication. This because ZigBee supports a lot of devices and a high degree of interoperability not seen in the other protocols. Since the ZigBee protocol is used for device communication this is also the protocol to communicate with the smart meter.

To allow the user to control the HPS it is accessible via a web browser on the end-users devices. This to make the product more affordable to customers. The gathered data is stored on the HPS itself to keep costs down and prevent any security risks by going to a solution where a part or the full amount of data is stored on a web server / service.

For the architecture itself the layer approach is used to allow for a solid architecture that is reliable, extendible and the different layers can be developed concurrently.

The most important decisions are summarised below:

- Use policies to allow end-users to have maximum control over the device.
- Zigbee: Smart Energy to communicate between smart devices, socket meters and the smart meter.
- Web-interface to allow end-users to control the device.
- Data storing on the HPS locally.
- Three-layered approach in the software architecture.

### 9.2.2 Utility Tree

In this section the quality attribute utility tree is introduced. The quality factors that comprise the system "utility" are the key drivers of our system adaptability, reliability and usability (see Chapter 4.3 for details). These factors are further refined towards the requirements. To elicit the scenarios that will be explained more detailed the Quality Attributes Prioritization method is used (displayed in grey). This requirements in bold are written down more detailed into scenario's.

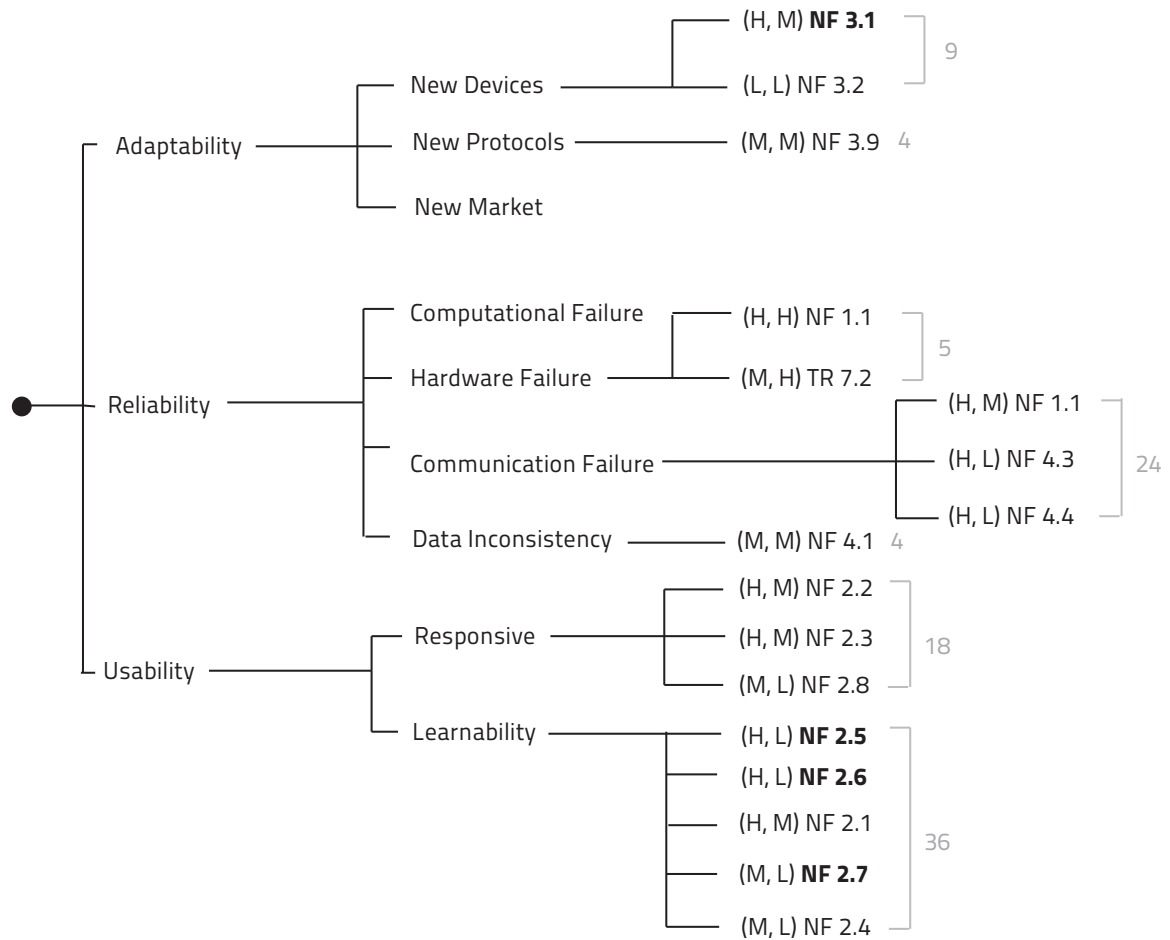
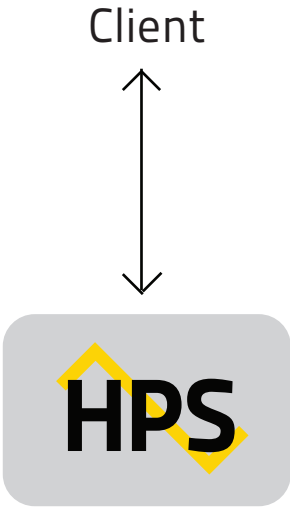


Figure 25: Quality attribute utility tree

Scenario: NF 2.5	The HPS has encountered a error, a message is displayed to the end-user explaining the error including how to solve it.
Scenario: NF 2.6	The entire interface is consistent and is experienced as a single environment.
Scenario: NF 2.7	The user recieve feedback from the system, when waiting for a report a waiting indicator is used.
Scenario: NF 3.1	The developer wished to add support for a range of new devices just entering the market.

### 9.2.3 Low-Level Scenarios

The purpose of this section is to identify sensitivity points and tradeoff points, risks and non risks to the quality attributes found in the utility tree. In this case one scenario is analysed for illustrative purposes.

Scenario #1	Detect a failure and inform the end-user.				
Q-Attribute(s)	Usability				
Environment	Normal operation				
Stimulus	User interaction with the HPS				
Response	The system will enter maintenance mode in order to protect the system, giving a error message to the end-user. This message will include a guide to recover from the error. Although in maintenance mode the HPS will try and resume normal operations depending on the severity of the error.				
DDs	Req.	Sensitiv.	Tradeoff	Risk	Non-risk
System	NF-1.1				NR1
Web-Interface	NF-2.5				NR2
Back-up	NF-4.1	S1	TO1	R1	
Flash Memory		S1	TO2		NR3
Reasoning	In the worst case scenario the system cannot recover from the system on its own, in this case a back-up can be restored.				
Arch. model	 <p>The diagram illustrates the architectural model. At the top, the word 'Client' is centered. Below it is a vertical double-headed arrow. At the bottom of the arrow is a grey rounded rectangle containing a yellow house icon and the letters 'HPS' in bold black font.</p>				
	S1: Back-up functionality adds more complexity to the system.				
	R1: Hardware failure does not allow for the back-up to be restored.				
	NR1: System is error free for 99,9% of the time.				
	NR2: The system shows an error message including feedback on the restoring process.				
	NR3: Storage is enough to store all the data to be able to return to a previous state of the machine.				
	TO1: Price vs Reliability - Back-up will require more storage but will give more reliability.				
	TO2: Price vs Capacity - 8gb is chosen to store all the back-up data.				
Conclusion	There is no fail safe if the hardware fails (fire, falling damage, short circuit, ect. Possibly a backup tool on the end-user pc could accommodate this to allow additional backup ability's.				

## 10 System evolution

The previous chapters presented a solution for the model of the HPS, as initially proposed by ESH (cf. Chapter 1). Even though the system, as it is proposed, has some great features, it has to keep evolving over time in order to preserve leadership in the market. The addition of new features will be the key factor in achieving this and it will make it possible for ESH to evolve the HPS into a bigger and better system.

### 10.1 Mobile Devices

Mobile devices are a great sales driver for customers to buy a certain system. With the mobile software, which the customer can install on their mobile phone with Android, iPhone or Windows Phone, is the customer able to access the HPS system from their mobile phone. The software will be released through the Android Market, Apple App Store and Windows Phone Marketplace. The software for the mobile devices will be released for free on all platforms, and enables the customer all possible actions which are possible through the web-interface, but adapted for mobile phone use. The decisions that are affected by this extension are: 17, 31 and 33 since a web-interface will not be enough, the web server will now also need to act as a service. The fact that the structure of the web server will need to be adjusted will make it difficult to implement.

### 10.2 Expansion and Localization

The HPS is planned to launch in Europe in 2015 Q1 (cf. chapter 2.7). This means that besides English and Dutch, a lot of other languages have to be supported by the HPS device. In the future, it is likely that the HPS will be launched also in other continents (e.g. U.S.A.), which might not only require new localizations, but also hardware changes (f.e. due to different power plug sockets or different electricity voltages). This will require the languages configuration to be extended with new languages, this is easy to implement. The decision that is affected by this extension is: 22 since the Plugwise Circle Socket Meter will not work for instance in the United Kingdom.

### 10.3 Business Device

The business sector is a large market for ESH and the HPS is scheduled for a business launch in The Netherlands in 2014 Q1 (cf. 2.7). In order to be able to offer the device for business it has to be adapted for business use due to major differences with households (e.g. working days from 9 to 5 in businesses, whereas households typically operate around those times). There is also a large difference in the number and types of devices between businesses and households (e.g. 50 personal computers in businesses, whereas households have one personal computer, one washing machine, etcetera). These differences require the system to evolve in a more specialized device for business use. The decisions that are affected by this extension are: 20 and 21, since the hardware required for a business will need to be a lot more powerful since the scale is a lot larger than a household. This will be reasonably difficult to implement since the software will also need to be extended.

### 10.4 Gas and Water

ZigBee is supported by a lot of different products, including gas and water meters [56]. For the evolution of the HPS system it is possible for the HPS device to monitor the amount of gas and water consumed and produced. It is also possible for the HPS device to set policies for gas and water in a similar way as electricity, in case there are off-the-shelf meters available which can control (e.g. enable flow, disable flow, limit flow) gas and water using ZigBee. The developing



team of ESH will need to write a driver for the gas and water so the implementation difficulty is medium.

## 10.5 Automated Control

In a future software update statistics could be gathered by ESH about the usage of the HPS and the policies. This enables ESH to create an automated policy algorithm, with different configurations (e.g. lowest price), for legacy devices and ZigBee enabled devices.

## 10.6 Integrated Device

A new software update by ESH will add the ability to include HAN into the HPS. ESH already has a lot of experience with HAN devices, making it the perfect solution in just one box. This integrated device enables the customer not only to have all current functionality of the HPS device, but it also enables the customer to control wireless home security, lights, HVAC, and home entertainment [24]. This will affect the software running on the HPS.

## 10.7 Energy Storage and Production

In the future it will be possible for the HPS device to interact (e.g. use, control, monitor) with energy storage devices (f.e. electric car, water pump) or energy production devices (f.e. gas generator, solar panel, wind turbine).

## 10.8 HPS versions

Version	Description
1.0	Current version.
1.1	Language extension to all european languages.
1.2	Allow apps on mobile devices to control the HPS even when not at home.
1.3	Extend the HPS to allow the control of Energy STorage and Production.
1.4	Extention to gas and water to the HPS possibilities.
2.0	Commercial application, support limitless ammounts of smart devices and socket meters. Allow for different groups and views to get a good overview.
2.1	Allow for a distributed HPS system within commercial applications, where multiple HPS units are installed to create a distributed version of the single device product.
3.0	Self learning HPS that will learn from end-users behaviour and behaviour of all users owning a HPS allowing for the end-user to let the HPS take control. This product probably will no longer be called the HPS and will be comprehensive including all HAS devices, heating, smart appliances, etc.

## References

- [1] ZIGBEE+HOMEPLUG JOINT WORKING GROUP: Smart Energy Profile Marketing Requirements Document. 2009 (Draft Revision 1.0). – Forschungsbericht
- [2] ZIGBEE ALLIANCE: ZigBee Home Automation Public Application Profile. Version: 2010. <http://www.zigbee.org/Standards/ZigBeeHomeAutomation/download.aspx> (1.1). – Forschungsbericht
- [3] ZIGBEE ALLIANCE AND HOMEPLUG POWERLINE ALLIANCE LIAISON: Smart Energy Profile 2.0 Public Application Protocol Specification. Version: July 2011. <http://zigbee.org/Standards/ZigBeeSmartEnergy/ZigBeeSmartEnergy20PublicApplicationProfile.aspx> (DRAFT). – Forschungsbericht
- [4] AVGERIOU, P. ; ZDUN, U. : Architectural Patterns Revisited - a Pattern Language. (2005)
- [5] BOUTIN, C. : *U.S., Europe Collaborating on Smart Grid Standards Development*. <http://www.nist.gov/smartgrid/grid-091311.cfm>. Version: September 2011, Abruf: 17.09.2011
- [6] BUSCHMANN, F. ; MEUNIER, R. ; ROHNERT, H. ; SOMMERLAD, P. ; STAL, M. : *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. 1. Wiley, 1996. – ISBN 0471958697
- [7] CENTOS: *Operating System Requirements*. [http://www.centos.org/docs/5/html/CDS/install/8.0/Installation\\_Guide-Support-Platforms.html](http://www.centos.org/docs/5/html/CDS/install/8.0/Installation_Guide-Support-Platforms.html). Version: October 2011, Abruf: 10.10.2011
- [8] CENTOS: *Purpose of CentOS*. <http://www.centos.org/modules/tinycontent/index.php?id=3>. Version: October 2011, Abruf: 10.10.2011
- [9] COBWEB: *In-house vs. Outsourcing Business Benefits*. <http://www.cobweb.com/learn-now/business-topics/inhouse-vs-outsourcing/business-benefits.aspx>. Version: October 2011, Abruf: 18.10.2011
- [10] COBWEB: *Outsource vs. In-House Faxing Solutions*. <http://www.secure-faxing.net/2011/03/outsource-vs-in-house-faxing-solutions.html>. Version: October 2011, Abruf: 18.10.2011
- [11] COMPUTERHOPE: *Linux vs. Windows*. <http://www.computerhope.com/issues/ch000575.htm>. Version: October 2011, Abruf: 17.10.2011
- [12] DOLIN, R. : *Smart Grid 2.0: Moving Beyond Meter-Centric Systems*. [http://www.smartgridnews.com/artman/publish/Business\\_Strategy\\_Resources/Smart-Grid-2-0-Beyond-Meters-and-onto-Intelligent-Energy-Management-2582.html](http://www.smartgridnews.com/artman/publish/Business_Strategy_Resources/Smart-Grid-2-0-Beyond-Meters-and-onto-Intelligent-Energy-Management-2582.html). Version: September 2011, Abruf: 26.09.2011
- [13] ENERGICS: *Smart Grid 1.0-3.0*. [http://www.energics.net/Smart\\_Grid\\_1.0-3.0.html](http://www.energics.net/Smart_Grid_1.0-3.0.html). Version: September 2011, Abruf: 26.09.2011
- [14] ENERGIE-VERGELIJKEN: *Energieprijzen vergelijken en overstappen*. <http://www.energie-vergelijken.nl/energie-vergelijken/energieprijzen-vergelijken>. Version: October 2011, Abruf: 23.10.2011
- [15] FALCON-SOFTWARE: *In-House vs. Outsourcing Web Hosting*. <http://blog.falcon-software.com/2010/08/24/in-house-vs-outsourcing-your-web-hosting/>. Version: October 2011, Abruf: 18.10.2011

- [16] FERC: *Demand Response*. <http://www.ferc.gov/legal/staff-reports/12-08-demand-response.pdf>. Version: October 2011, Abruf: 11.10.2011
- [17] FOWLER, M. : *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002. – ISBN 0321127420
- [18] FSCKIN: *Seven Different Linux/BSD Firewalls Reviewed*. <http://www.fsckin.com/2007/11/14/7-different-linuxbsd-firewalls-reviewed/>. Version: October 2011, Abruf: 25.10.2011
- [19] GAMMA, E. ; HELM, R. ; JOHNSON, R. ; VLISSIDES, J. : *Design Patterns: Elements of Reusable Object-Oriented Software*. 1. Addison-Wesley Professional, 1994. – ISBN 0201633612
- [20] GHEORGHIU, G. : *Agile Testing: Automated deployment systems: push vs. pull*. <http://agiletesting.blogspot.com/2010/03/automated-deployment-systems-push-vs.html>. Version: October 2011, Abruf: 17.10.2011
- [21] GO2LINUX: *Which Distro to choose? - Comparison -(Not a Debian vs Ubuntu vs Fedora vs Centos)*. <http://www.go2linux.org/debian-ubuntu-centos-fedora-comparison>. Version: October 2011, Abruf: 18.10.2011
- [22] GST: *OpenRD-Base*. <http://www.globalscaletechnologies.com/p-32-guruplug-server-plus.aspx>. Version: October 2011, Abruf: 23.10.2011
- [23] HKJIAHENG: *zigbee home automation wireless power meter socket*. <http://www.hkjiaheng.com/item/zigbee-home-automation-wireless/zigbee-home-automation-wireless/lid=15567979>. Version: October 2011, Abruf: 18.10.2011
- [24] HOMETOYS: *HomeToys eMagazine Article - Interview - Home Automation Networks*. <http://www.hometoys.com/ezone/11.02/murata/>. Version: October 2011, Abruf: 25.10.2011
- [25] IBM: *Deployment Diagrams*. <http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=%2Fcom.ibm.xttools.modeler.doc%2Ftopics%2Fcdepd.html>. Version: October 2011, Abruf: 25.10.2011
- [26] IBM: *Documenting software architecture, Part 2: Develop the system context*. <http://www.ibm.com/developerworks/library/ar-archdoc2/>. Version: October 2011, Abruf: 25.10.2011
- [27] IEEE (Hrsg.): *IEEE Std 830-1998 recommended practice for software requirements specifications*. IEEE, 1998
- [28] IT, U. : *Response Time Limits*. <http://www.useit.com/papers/responsetime.html>. Version: October 2011, Abruf: 25.10.2011
- [29] JOHN, J. S.: *Smart Grid 3.0: Getting There from Here*. <http://gigaom.com/cleantech/smart-grid-3-0-getting-there-from-here/>. Version: April 2011, Abruf: 17.09.2011
- [30] LINUX-DRIVERS: *Linux Hardware Compatability List*. <http://www.linux-drivers.org/>. Version: October 2011, Abruf: 23.10.2011
- [31] MANNION, M. ; KEEPECE, B. : SMART requirements. In: *ACM SIGSOFT Software Engineering Notes* 20 (1995), April, Nr. 2
- [32] MICROSOFT: *Message Bus*. <http://msdn.microsoft.com/en-us/library/ff647328.aspx>. Version: October 2011, Abruf: 25.10.2011

- [33] PCMAG: *ZigBee to Run Over Wi-Fi, Too*. <http://www.pcmag.com/article2/0,2817,2361520,00.asp#fbid=g0bXLTEbvuc>. Version: October 2011, Abruf: 23.10.2011
- [34] PLUGWISE: *Home Start Extension*. <http://www.pluginwise.com/nl/idplugtype-f/home-start-extension>. Version: October 2011, Abruf: 23.10.2011
- [35] PLUGWISE: *Plugwise Circle*. <http://www.pluginwise.com/idplugtype-f/circle>. Version: October 2011, Abruf: 04.10.2011
- [36] PLUGWISE: *Plugwise Circle*. <http://www.pluginwise.com/idplugtype-f/support/faq>. Version: October 2011, Abruf: 04.10.2011
- [37] SOMMERVILLE, I. : *Software Engineering: (Update) (8th Edition) (International Computer Science Series)*. 8. Addison Wesley, 2006. – ISBN 0321313798
- [38] STREET, T. : *power meter JNX-2000A(universal plug)GB/US/UK*. <http://www.free-business-opportunities.net/sell/show-87036.html>. Version: October 2011, Abruf: 18.10.2011
- [39] SUBRAMANIA, N. ; CHUNG, L. : *Metrics for Software Adaptability / Department of Computer Science University of Texas, Dallas*. – Forschungsbericht
- [40] TARGET, T. : *Determining cloud TCO vs. in-house IT costs*. <http://searchdatacenter.techtarget.com/tip/Determining-cloud-TCO-vs-in-house-IT-costs>. Version: October 2011, Abruf: 18.10.2011
- [41] TECHENCLAVE: *C++ vs Java vs Python vs Ruby*. <http://www.techenclave.com/programming/c-vs-java-vs-python-vs-66187.html>. Version: October 2011, Abruf: 25.10.2011
- [42] TRILLIANT: *The Evolution of the Smart Grid*. <http://www.trilliantinc.com/education/the-evolution-of-the-smart-grid/>. Version: September 2011, Abruf: 26.09.2011
- [43] VARIOUS: *AC power plugs and sockets*. [http://en.wikipedia.org/wiki/AC\\_power\\_plugs\\_and\\_sockets](http://en.wikipedia.org/wiki/AC_power_plugs_and_sockets). Version: October 2011, Abruf: 05.10.2011
- [44] VARIOUS: *Appendix B: Sample SLA*. <http://technet.microsoft.com/en-us/library/cc543293.aspx>. Version: October 2011, Abruf: 17.10.2011
- [45] VARIOUS: *Black and Decker Power Monitor*. <http://www.testfreaks.com/blog/review/black-and-decker-power-monitor/>. Version: October 2011, Abruf: 11.10.2011
- [46] VARIOUS: *Connection speeds compared*. <http://www.pixelbeat.org/speeds.html>. Version: October 2011, Abruf: 11.10.2011
- [47] VARIOUS: *Electrical grid*. [http://en.wikipedia.org/wiki/Electrical\\_grid](http://en.wikipedia.org/wiki/Electrical_grid). Version: September 2011, Abruf: 26.09.2011
- [48] VARIOUS: *Guruplug: Which distribution is/will be shipped?* <http://plugcomputer.org/pluginforum/index.php?topic=1432.0>. Version: October 2011, Abruf: 23.10.2011
- [49] VARIOUS: *Keepalive*. [http://en.wikipedia.org/wiki/Keepalive#TCP\\_keepalive](http://en.wikipedia.org/wiki/Keepalive#TCP_keepalive). Version: October 2011, Abruf: 11.10.2011
- [50] VARIOUS: *List of device bandwidths*. [http://en.wikipedia.org/wiki/List\\_of\\_device\\_bandwidths](http://en.wikipedia.org/wiki/List_of_device_bandwidths). Version: October 2011, Abruf: 10.10.2011

- [51] VARIOUS: *Moore's law*. [http://en.wikipedia.org/wiki/Moore's\\_law](http://en.wikipedia.org/wiki/Moore's_law). Version: September 2011, Abruf: 26.09.2011
- [52] VARIOUS: *Power Cost Monitor WiFi Edition*. [http://www.powercostmonitor.com/p8127/power\\_cost\\_monitor\\_wifi\\_edition.php](http://www.powercostmonitor.com/p8127/power_cost_monitor_wifi_edition.php). Version: October 2011, Abruf: 11.10.2011
- [53] VARIOUS: *Smart grid*. [http://en.wikipedia.org/wiki/Smart\\_grid](http://en.wikipedia.org/wiki/Smart_grid). Version: September 2011, Abruf: 27.09.2011
- [54] VARIOUS: *Usability testing*. [http://en.wikipedia.org/wiki/Usability\\_testing#How\\_many\\_users\\_to\\_test.3F](http://en.wikipedia.org/wiki/Usability_testing#How_many_users_to_test.3F). Version: October 2011, Abruf: 25.10.2011
- [55] WHISPERDALE: *Nginx vs Cherokee vs Apache vs Lighttpd*. <http://www.whisperdale.net/11-nginx-vs-cherokee-vs-apache-vs-lighttpd.html>. Version: October 2011, Abruf: 25.10.2011
- [56] ZIGBEE: *ZigBee Products Display*. <http://www.zigbee.org/Default.aspx?TabID=270&Ctrl=MyCompanyProducts>. Version: October 2011, Abruf: 25.10.2011

## Appendix

### A1 Example: Executive report

#### **Smart Monitoring** EXECUTIVE REPORT 26.09.2011

HELLO PIETER,

You did a good job today. Your total energy consumption has been lower than your average. You saved a total of **769 kw/h**. It seems that the sun was not shining that much today. Your solar panel could only generate **0,21 kw/h** but therefore your windmill produced **672 kw/h**.

**Peak:**  
The most energy has been consumed between **16:00 to 18:00** o'clock

**Low:**  
The least energy has been consumed between **00:00 to 02:00** o'clock

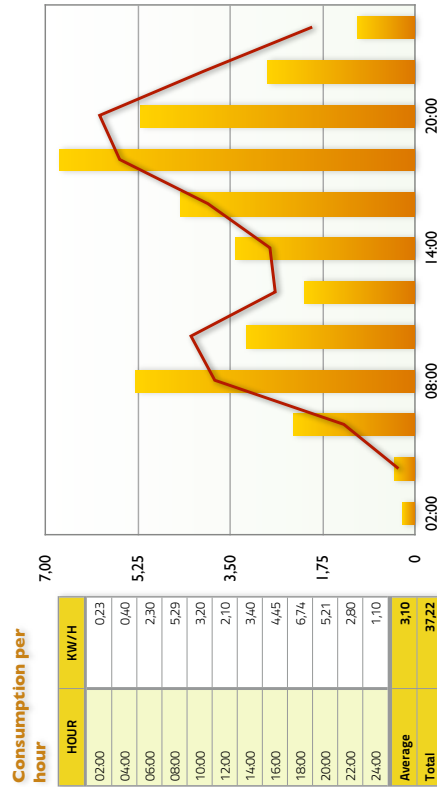
**Bill:**  
Your today's energy bill is estimated to be **9,02 Eur**.  
**! Define Smart Policies to make use of the realtime pricing of your utility**

**Earnings:**  
You earned today **1,68 Eur**.

Home Power Save saved you today 0,12Cent. In total Home Power saved 2340 Eur within 78 days.

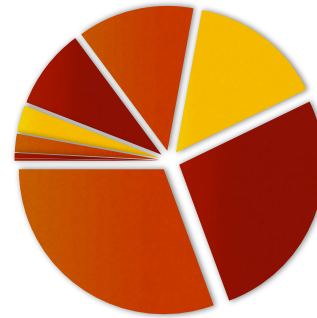
Yours,  
European Smart Homes

Available Reports:  
 • Detailed Consumption Report  
 • Energy Bill



**Consumption per device**

Device	Hour	KW/H
Washing Machine		0,10
Dryer		0,21
Light Living Room		0,76
Computer Harry		1,12
Media		3,34
Computer Pieter		4,85
Smart Socket #1		5,56
Cooking plate		9,89
Unmonitored		11,39
<b>Total</b>		<b>37,22</b>



## A2 Decision

### A2.1 Arguments

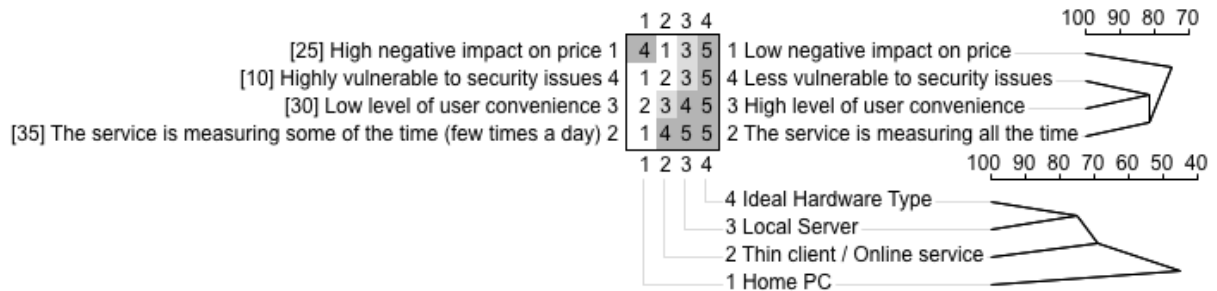


Figure 26: Webgrid for decision 13

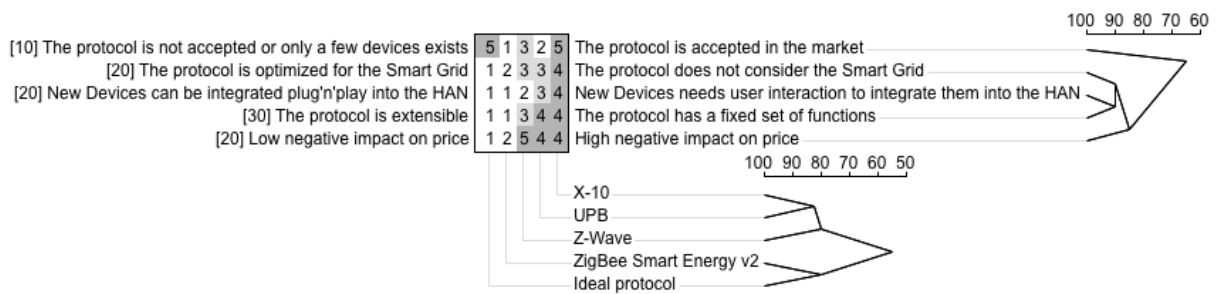


Figure 27: Webgrid for decision 14

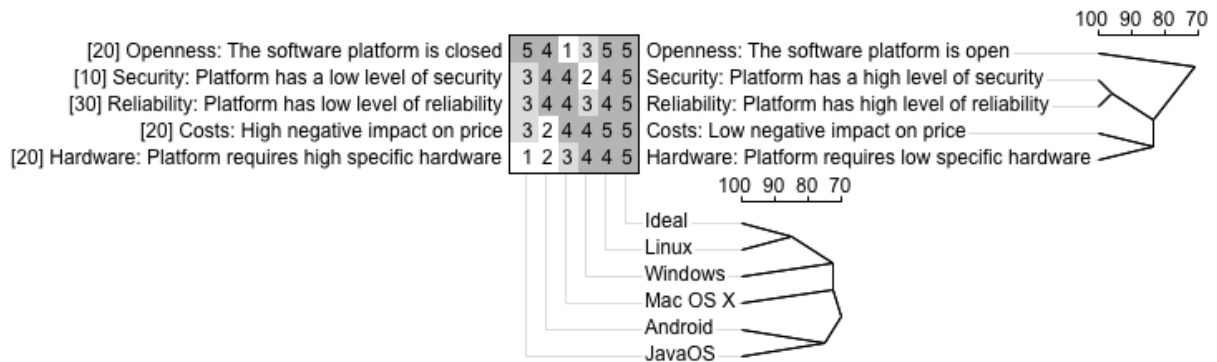


Figure 28: Webgrid for decision 15

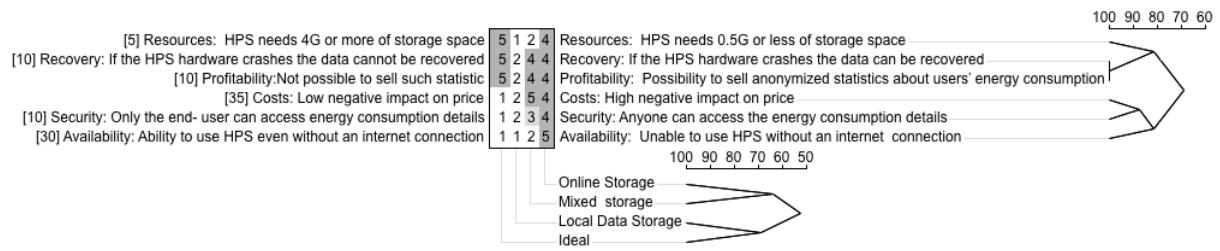


Figure 29: Webgrid for decision 16

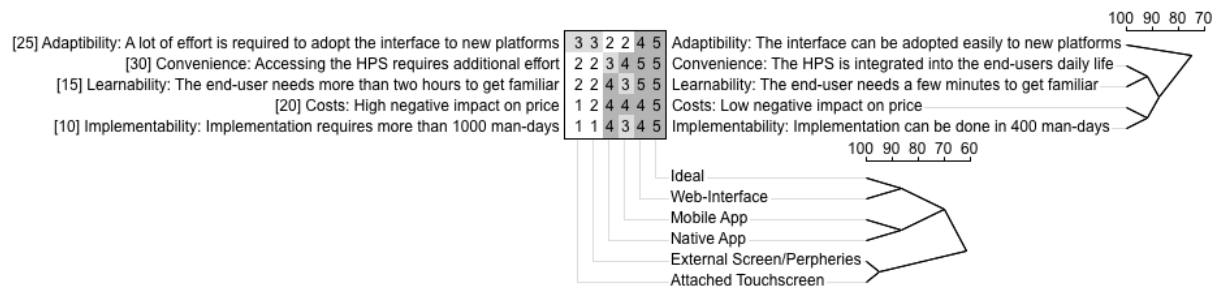


Figure 30: Webgrid for decision 17

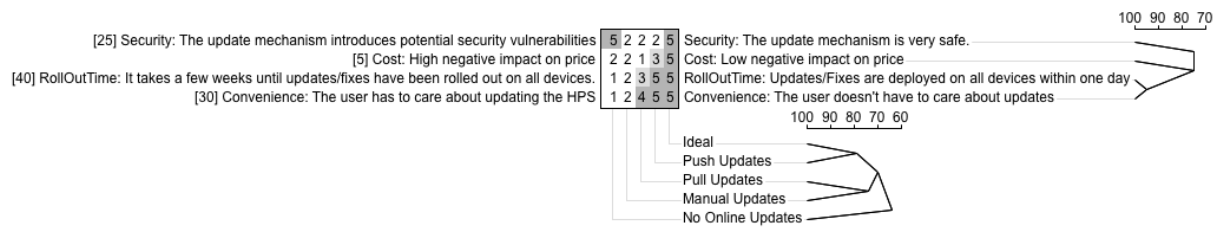


Figure 31: Webgrid for decision 18

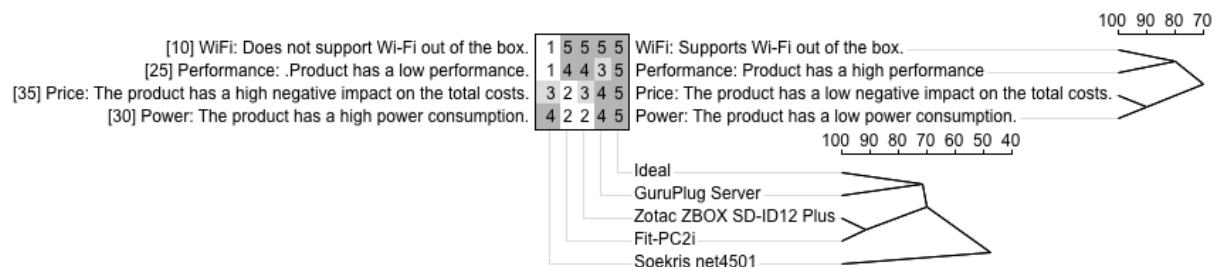


Figure 32: Webgrid for decision 20



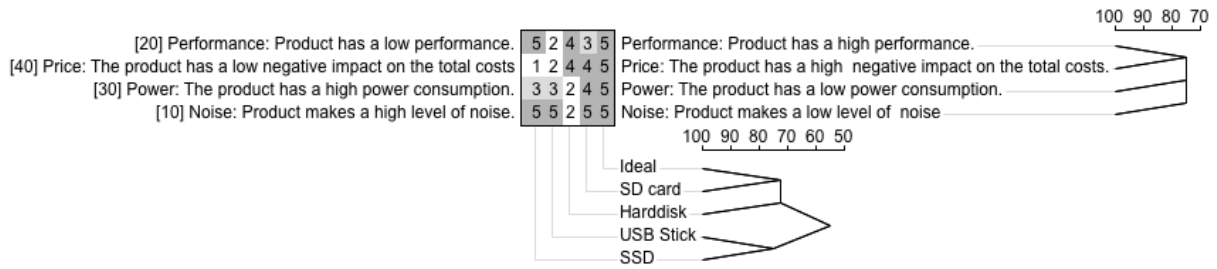


Figure 33: Webgrid for decision 21

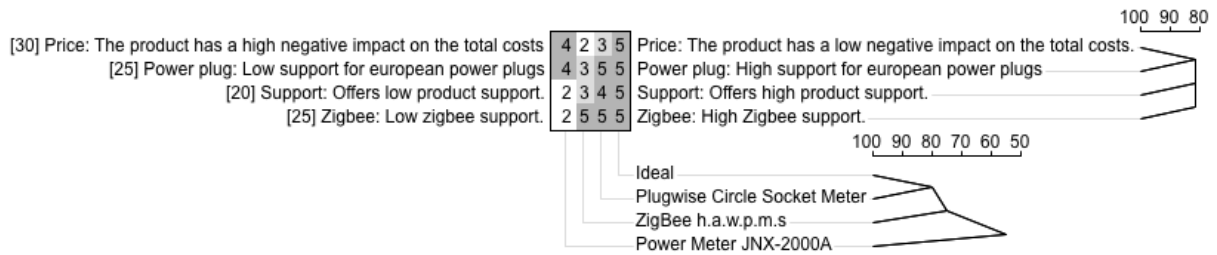


Figure 34: Webgrid for decision 22

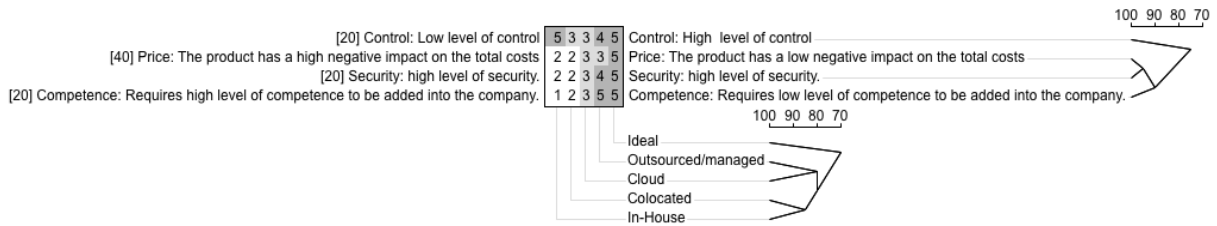


Figure 35: Webgrid for decision 23

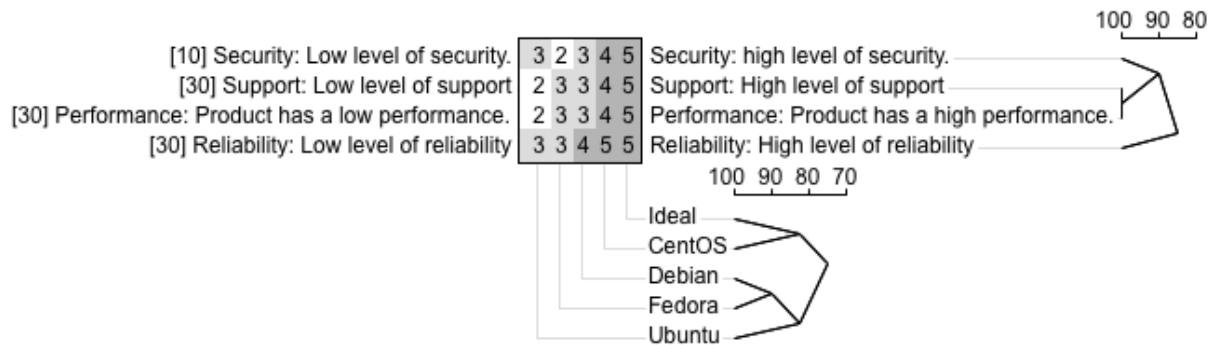


Figure 36: Webgrid for decision 24

## A2.2 Deployment View

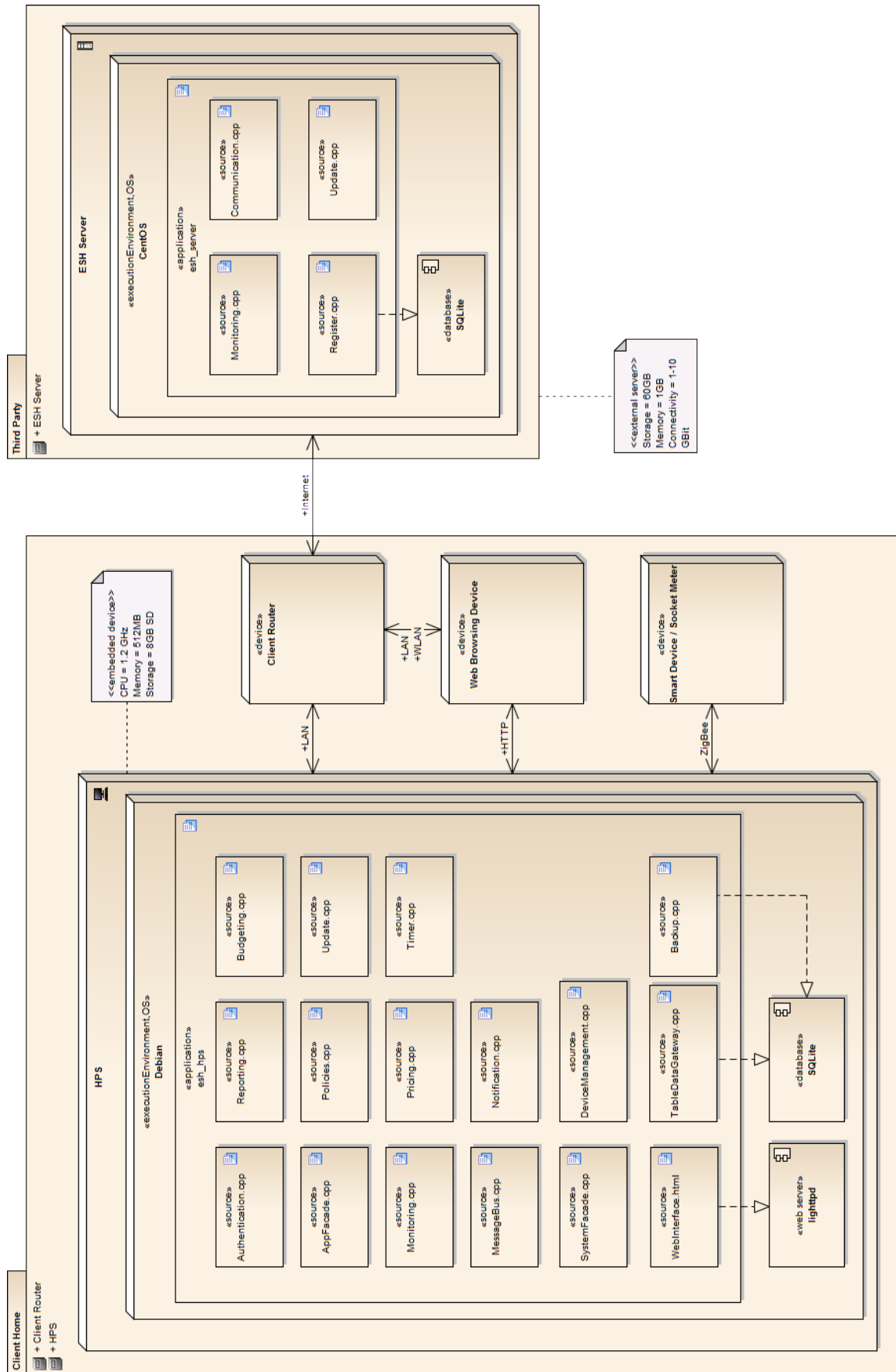


Figure 37: Deployment View

## A2.3 Chronological View

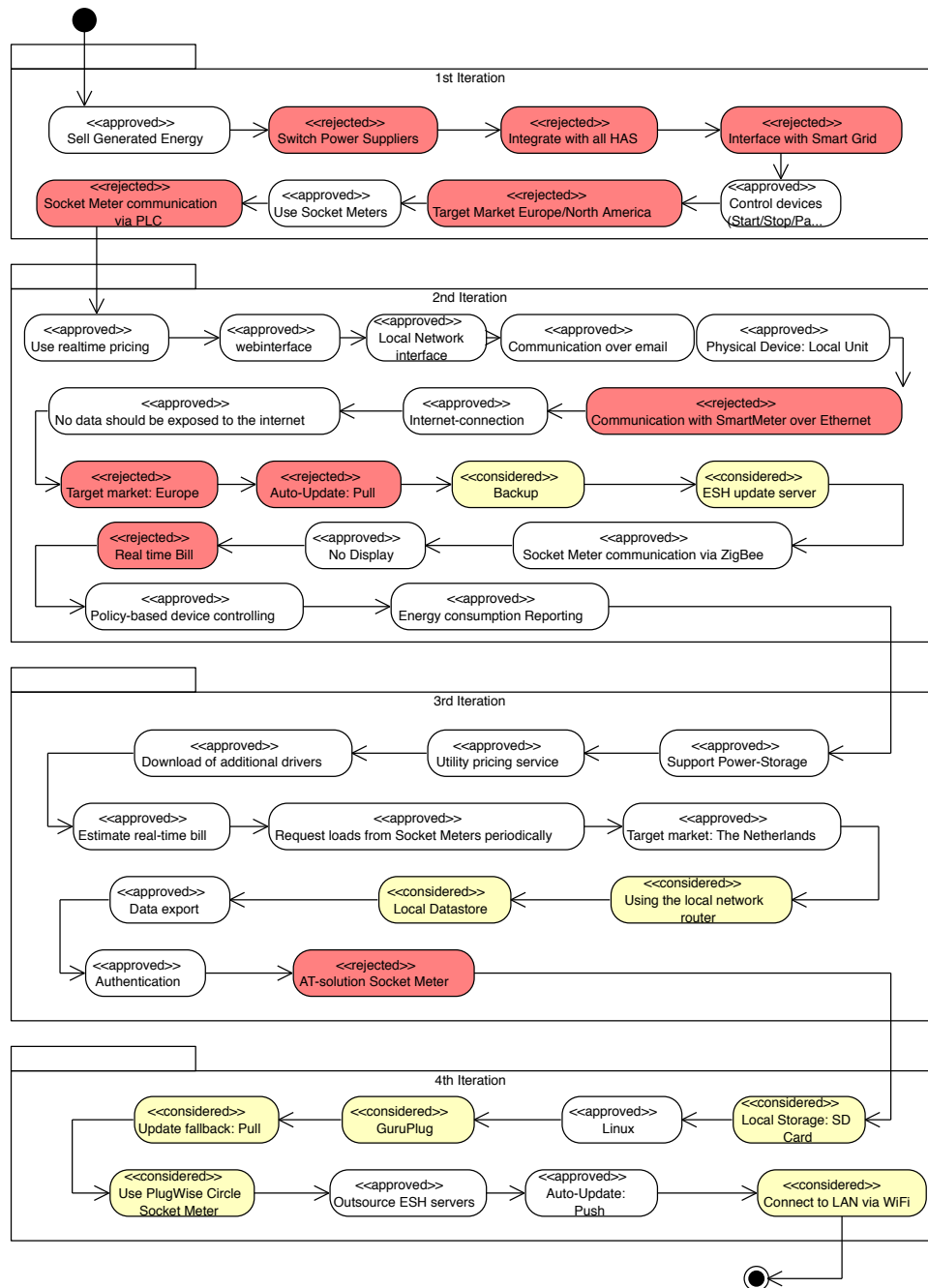


Figure 38: Chronological View

## A2.4 Relation View

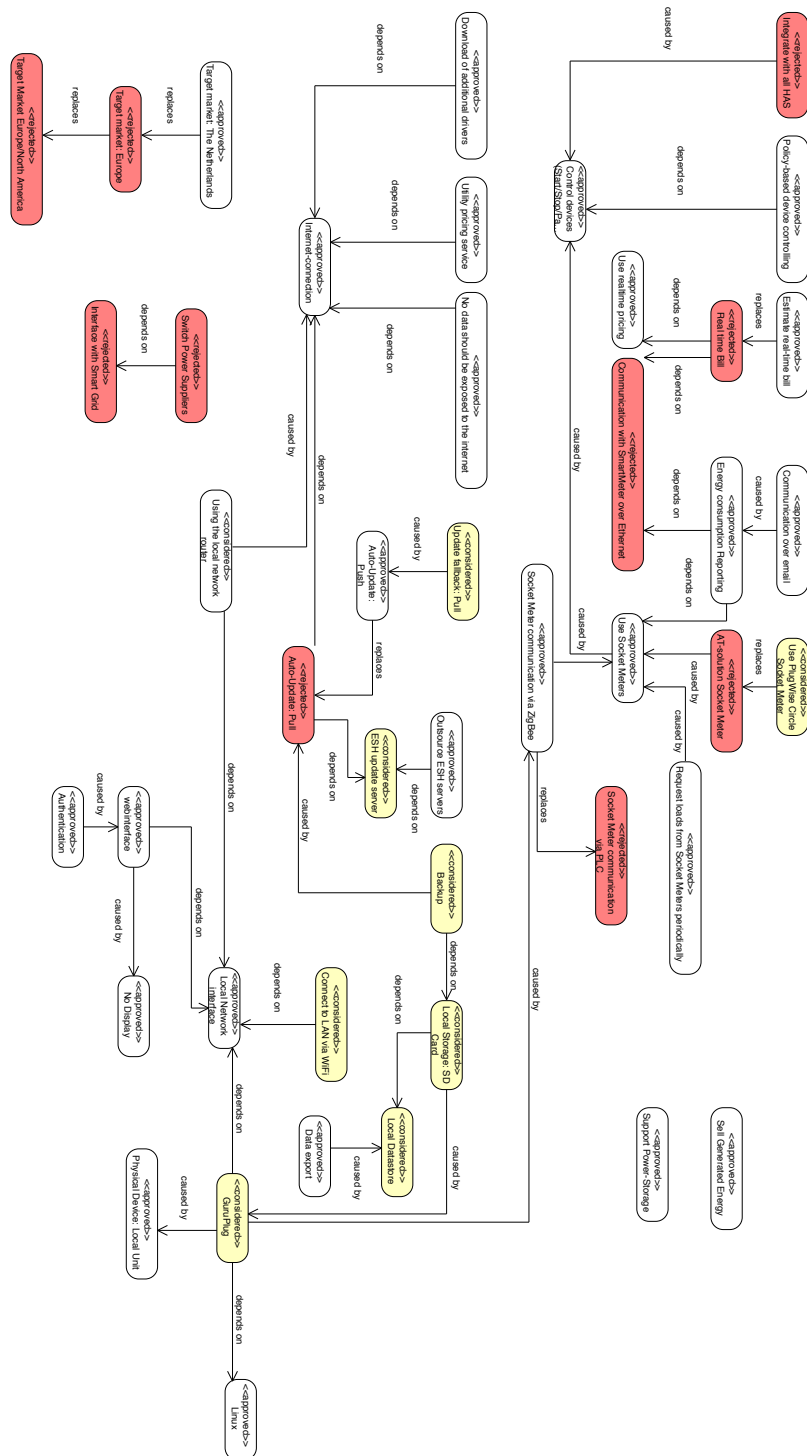


Figure 39: Relationship View

## A3 MessageBus Events

### A3.1 Monitoring

Table 38: Events of the Monitoring component

Name of Event	Action	Description
RequestLoadEvent	Receive	Request of load data. The message field contains a list of DeviceIDs and the timespan of the load data. The monitoring component responds with a DirectedMessage to the sender
LoadDataEvent	Send	Response to RequestLoadEvent, contains the load data of certain devices.
LoadUpdateEvent	Send	This Event will be dispatched when new LoadData is available. The message contains a list of devices.
FaultEvent	Send	This Event will be dispatched when an error within the component occurs.

### A3.2 Pricing

Table 39: Events of the Pricing component

Name of Event	Action	Description
RequestPrice	Receive	Request of current price. The message field contains the current energy price or if the timespan is set will return the energy price of that time. When the timespan is not a single date and time but a from to timespan all the date in between will be returned.
LoadPrice	Send	Response to RequestPrice, contains the price data.
FaultEvent	Send	This Event will be dispatched when an error within the component occurs.

### A3.3 Budgeting

Table 40: Events of the Budgeting component

Name of Event	Action	Description
RequestBudget	Receive	Request of budget. this message gives a certain budget and timespan to the budget component. The budget component then calculates if, for that moment, the budget is met.
LoadBudget	Send	Response to RequestBudget, contains a message to the policyManager with the device, and either a true or false if the budget is met.
FaultEvent	Send	This Event will be dispatched when an error within the component occurs.

## A4 Risk Assessment

<b>Risk</b>	<b>ILS</b>	<b>Responsibility</b>	<b>Threshold</b>	<b>Consequences</b>	<b>Prevention</b>	<b>Reaction</b>
<i>Business</i>						
Wrong budget estimation	High, Low, Very low	Architect	Costs of milestones exceed estimation	Reduced features or dropped product	Keep track of money spent per feature and on total product	Reduce features or drop product
Project scope expansion	Medium, Low, Low	Architect	No milestones met or too late	HPS might ship with uninteresting features for the customer or too little due to massive cutback in features to maintain schedule	Market research about features customer want to buy and stick with it	Start from scratch
Too much competition to cover TCO	High, Medium, High	Architect & Market researcher	Not enough sales to being on track to cover the TCO within the initially specified period	Loss of money or bankruptcy	Make a good analysis about market expectations and competitors, improve unique sellings points to competitors	Increase number of unique features, reduce price of the product
<i>Schedule</i>						
Wrong time estimation	High, Low, Medium	Architect & Project manager	Milestones not achieved or too late	Reduced market share, reduced features	Plan each feature with estimated time, drop features if required to achieve milestones	Reduce features and improve planning for the other milestones
Unexpected project scope expansions	Medium, Medium, Medium	Architect & Requirements engineer	Milestones not achieved or too late	Delayed launch for the HPS, reduced features	Improve market research for the features, drop features if required to achieve milestones	Delay launch, limit other features
Failed to identify required time to develop each functionality	Medium, Low, Medium	Architect	Milestones not achieved or too late	Delayed launch for the HPS, reduced features	Ask multiple developers and make a good estimation	Delay launch, limit other features



<i>Technical</i>						
Smart Grid standardization and usage	Medium, High, High	Designer	Standardization and adoption by the market	Incompatible devices (in stores and homes)	Make the devices updatable to incorporate changes to the Smart Grid	Update the current devices with new software, introduce new hardware, 20 euro cashback for old hardware
Hacker gains access to HPS	High, Medium, High	Designer & Developer	N/A	Unusable devices, wrong configurations set, loss of electricity settings	Code review, security checks, fault tolerance and let it check by hackers before release	Update current devices, allow users to swap their devices at the stores
Continuous changing requirements	High, Medium, High	Market researcher & Requirements engineer	N/A	Require new hardware or software development due to large changes	Finalize features at the start of development	Delay launch for the HPS, reduce features
No technology available or existing technology is in development	High, Low, Medium	Architect & Developer & Project manager	N/A	Development of technology or wait until finalized	Hire technology veterans, use only existing, proven, technology	Delay launch for the HPS, reduce features, hire technology veterans

<i>Implementation</i>						
Best pricing algorithm	Medium, Low, Medium	Developer	N/A	Price algorithm doesn't result in the best price	Implement different algorithms, use beta tests	Update the HPS if a better pricing algorithm is developed
Difficult module or hardware integration	Low, Low, Very low	Software Architect & Developer	Integration takes more time compared to planning	Milestones not achieved or too late	Research into the integration of modules or hardware	Delay launch, limit other features, change modules or hardware, hire more developers
<i>Operational</i>						
No communication in team	Low, Low, Very low	All	Multiple development conflicts caused by lack of communication	Conflicting development efforts	Team meetings	Team meetings
Unable to resolve responsibilities	Low, Low, Very low	Project manager	N/A	Wrong or multiple people taking the lead, leading to conflicts	Assign clear roles where everyone agrees upon before the start of the project	Reassign roles to each member of the project
Product defects	Low, Medium, Low	System Architect, Designer & Developer	+10% return rate within warranty period	Loss of money, bad reputation	Set up a Q&A process, try to improve product quality	Look for product defect cause and improve upon
<i>Other</i>						
Government law changes	Low, Low, Very low	System Architect	N/A	Incompatible product	Make the product updatable and/or upgradable	Update or upgrade the product if possible, new product otherwise
Market development	Medium, Medium, Medium	System Architect	N/A	Outdated product	Keep track of market development	Decide to adapt to the market

## A5 Service Level Agreement

Requirement	Information
Support hours	Incident resolution should happen 95% of the time in fewer than 2 hours for normal incidents, and 30 minutes for complete outages of the services. The support hours are 24 hours a day, seven days a week.
Service Availability	Required availability for these services is 99.5% uptime, not counting planned maintenance times. The 99.5% availability metric will be measured by a rolling 6-month period.
Reliability	The service is guaranteed not to break more than 3 times per year. A break is defined as the loss of access to a vital business function.
Performance	Designed for high performance, the HPS must not wait for response by the server for more than 1 minute out of any 10 minute window. Any failures must be reported to ESH for incident resolution.
Continuity	In the case of a major catastrophe with hardware loss, servers and infrastructure have to be leased from various providers.
Security	Strong passwords must be used to access all services. Strong passwords are defined as being having more than 8 characters, not matching standard “dictionary” definitions. Also required for the servers will be (software based) anti-virus, spyware/malware protection, and firewall protection.

Table 41: Service Level Agreement requirements for the ESH server.

## A6 Mock-up Prototype

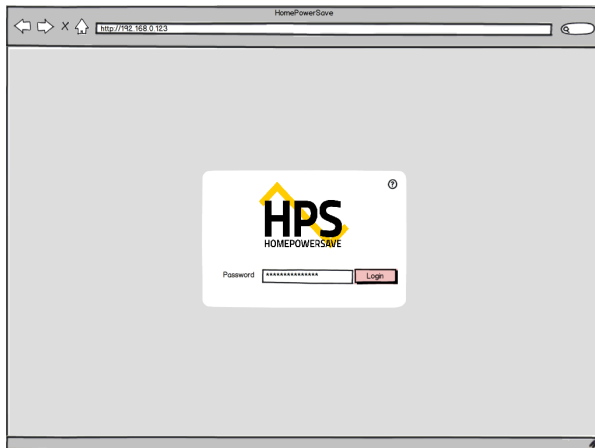


Figure 40: Login

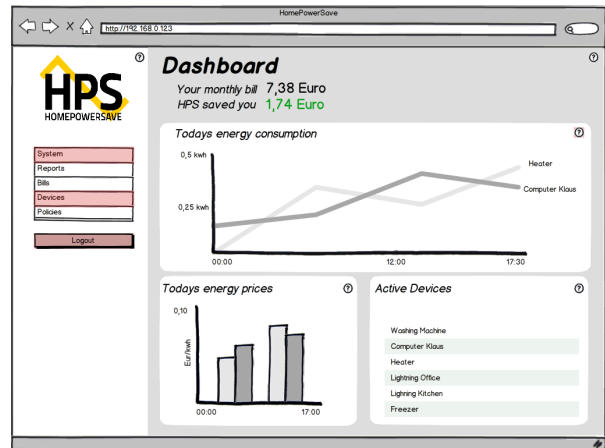


Figure 41: Dashboard / Startpage

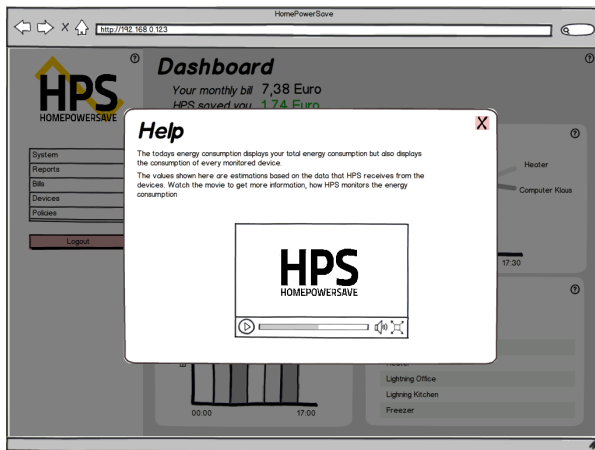


Figure 42: Context-sensitive help

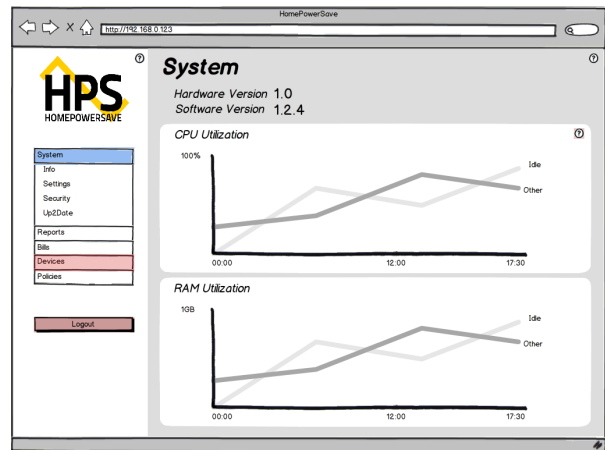


Figure 43: System overview

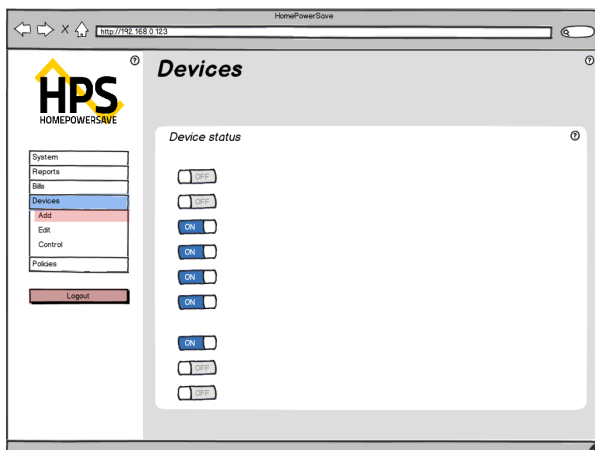


Figure 44: Enable/Disable devices

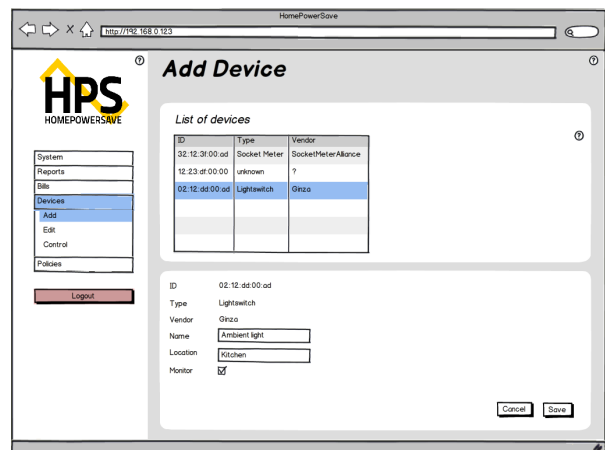


Figure 45: Add device

## A7 Time Tracking

Week 1

Person	Task	Hours
hp	Chapter 1	1
	Meeting	2
		<b>3</b>
es	Chapter 2.3	1
	Meeting	2
		<b>3</b>
mj	Chapter 2.2 + 2.4	1.5
	Organize meeting + group division of chapters 1 & 2	1
		<b>2.5</b>
gk	Meeting	2
	Chapter 2.5 & 2.6	1
		<b>3</b>
cm	Read architecting case and documents	2
		<b>2</b>
eb	Meeting	2
	Chapter 2.7 competitors	1
	Putting all chapters (.docx) together in one document (coach)	1
		<b>4</b>
<b>Group total</b>		<b>17.5</b>

## Week 2

Person	Task	Hours
hp	Chapter 4.3	1.5
	Research SmartGrid and first draft of 4.3	2
	Agenda for meeting + work on a definition about our HPS system	2
	Meetings	4
		<b>9.5</b>
mj	Group division of chapters 3 & 4	0.5
	Research and draft for chapter 3.8 + start for chapter 3.9	2
	Agenda + Research Smart Grid	1.5
	Meetings	4
		<b>8</b>
gk	Business model and Domain model (2.5 and 2.6)	1
	Research for Chapter 3 and 4	2
	Meetings	7
		<b>10</b>
cm	Meetings	3
	Review	1
	Setup L <sup>A</sup> T <sub>E</sub> Xdocument and business vision	3
		<b>7</b>
es	chapter 3.2 and 3.3	1
	Research Smart Grid	1
	Meetings	4
	Review	1
		<b>7</b>
eb	Meetings	7
	Research Smart Grid	1
		<b>8</b>
<b>Group total</b>		<b>49.5</b>

### Week 3

Person	Task	Hours
mj	Chapter 3.9 + 4.1	3.5
	Meetings	8
	Chapter 2.4 + 3.8 + 3.9 + 4.1 + time tracking according to review	2.5
	Group division of chapters 5 & 6 + Research Smart Meters	2
		<b>16</b>
cm	Meetings	7
	Review	1
	L <sup>A</sup> T <sub>E</sub> Xexamples and support	2.25
	Research Smart Grid	1
	Research on Stutensee pilot and real-time energy markets	3
	Use cases/business vision	2.5
	Review of 1,2,3	3.5
		<b>20.25</b>
gk	Meetings	7
	Review	1
	Functional requirements	1
		<b>9</b>
eb	Meetings	3
	Project description to text	3
	Functional requirements	1
		<b>7</b>
hp	Meetings	7
	Further development on 4.3	1
	Commercial requirements	2
		<b>10</b>
es	Meetings	3
	Review	1
	Improvement Stakeholders	1
	Technical non-functional requirements	3
		<b>8</b>
<b>Group total</b>		<b>70.25</b>

Week 4

Person	Task	Hours
mj	Research for and draft of technological roadmap	4
	Updated references and bibliography	0.5
	Meetings	5
	Getting LaTeX glossaries working on Windows + changes to target audience and key drivers	1.5
	Changed target audience according to review + changed order in 'Requirements' + intro for 'Requirements' + small fixes to use-cases	2
	Research into smart meters for Hardware architecture	2
	Presentation	2
	Read/review document of other group	3
		<b>23</b>
cm	Meetings	5
	Technical non-functional requirements	3
	Improvement 1.2 & 1.3	1.5
	Example executive report	1.5
	Error fixing, review of 1 and 2	1.25
	Use-cases, requirements, research ZigBee Smart Energy	2
	Review group 4	2
		<b>16.25</b>
gk	Meetings	5
	Requirements	1
	Improvements chapter 2	2
	Research for hardware	2
	Read/review document of other group	3
		<b>13</b>
hp	Meetings	5
	Presentation	4
	Research	1
		<b>10</b>
eb	Meetings	5
	LaTeX glossaries working on Linux Ubuntu	0.5
	Functional requirements	2
	Read/review document of other group	3
		<b>10.5</b>
es	Meetings	5
	Read/review document of other group	3
		<b>8</b>
<b>Group total</b>		<b>72.75</b>



Week 5

Person	Task	Hours
eb	Meetings	1
	Functional requirements	2
		<b>3</b>
hp	Meetings	1
	Design Alternatives	4
	Design Alternatives Research	1
	Hardware Architecture	2
	Decision Viewpoints	5
		<b>13</b>
es	Meetings	1
	Improvement Stakeholders	1
	Verification	1
		<b>3</b>
gk	Meetings	2
	Functional requirements	2
	Technical requirements	2
	Hardware research	2
		<b>8</b>
mj	Hardware architecture research	2.5
	Hardware architecture socket meters, servers and main device	7
	Functional requirements	0.5
	Assumptions	1.5
	Meetings	2
		<b>13.5</b>
cm	Meetings	2
	use cases, stakeholders, requirements	5
	Decision Viewpoints	5
	Mockup prototype	1,5
		<b>13.5</b>
<b>Group total:</b>		<b>54</b>

Week 6

Person	Task	Hours
eb	Added Architectural Vision	1.5
	Added financial costs	2
	Meetings	3
	initial models	2
		<b>8.5</b>
hp	Meetings	3
	Changing consistency	1
	Hardware	3
		<b>7</b>
es	Meetings	2
	Improvement Stakeholders	2
	Verification	1
		<b>5</b>
gk	Find good tooling and research for Initial models	2
	Initial models	3
	Meetings	2.5
	Converting images to pdf and crop it	2.5
		<b>10</b>
mj	Reworked hardware overview diagram	2
	Changes according to review on hardware overview - socket meters	2
	Hardware overview - server	4
	Competitors	1
	Lots of text and layout fixes	1
	Meetings	2
		<b>12</b>
cm	Meetings	4
	Review	1
	Requirements	2
	Use-cases	1
		<b>8</b>
<b>Group total:</b>		<b>50.5</b>

Week 7

Person	Task	Hours
eb	Meeting	2
	Decomposition of software architecture	6
	CRC-Cards	2
	ESH Update server component	3
		<b>13</b>
hp	Meeting	2
	Clean - Chapter 2	1
	Desisions	4
		<b>7</b>
es	Meeting	2
	Process View	3
		<b>5</b>
gk	Meeting	2
	Rewriting chapter 5	2
	Research for system architecture	3
	Layout fixes and todo's in the document	3
		<b>10</b>
mj	Research for hardware architecture	3
	Hardware architecture chapters 2, 3, 4 and 5	5
	Hardware architecture decisions + concerns	5.5
	Hardware architecture changes according to review	3
	Layout fixes, time tracking calculations and todo's in the document	2
	Meetings	2
		<b>20.5</b>
cm	Meeting	3
	Review	1
	Research ZigBee Discovery/Application protocol	2.5
	Use-cases	2.5
	Requirements	5.5
	Decisions	6
	General layout improvements	1
		<b>21.5</b>
<b>Group total:</b>		<b>77</b>

## Week 8

Person	Task	Hours
eb	Meetings	2
	ESH Update Server, Shared repository, Notification, System Facade, Update (chapter 7)	6
	Research how to do the firmware update	3
		<b>11</b>
hp	Webgrid 5 - High Level Requirements	3
	Initial decomposition of software arch.	6
	Webgrid 5 - Hardware decisions	3
	Added reporting and setup of chapter 8	1
	Meeting	2
	Software Architecture	6
		<b>21</b>
es	Meetings	2
	Improvement on system architecture verification	4
		<b>6</b>
gk	Meeting	2
	Chapter 5 research	4
	Chapter 5 writing	4
	Layout fixes and todo's in the document	3
	Updated assumptions	1
	Start with process view	2
		<b>16</b>
mj	Meetings	5
	Research for + start for draft of requirements verification	4
	Changes to the requirements + changes to decision layout	2
	Research for + draft for the deployment view	5
	Improving deployment view	3
	Several introductions to chapters	1.5
	Review other group	1.5
		<b>22</b>
cm	Meetings	5
	Review	1
	Initial decomposition of software arch.	15
	Usecases and requirements, consistency checks	4.5
	Overview, Message, System Decomp	5
	monitoring, processes for message bus and device management, variability	4
	guide, updating several diagrams	
		<b>32.5</b>
<b>Group total:</b>		<b>108.5</b>

Week 9

Person	Task	Hours
eb	Created database schema for HPS and ESH	3
	Meeting	1
	Presentation	4
		<b>8</b>
hp	Meeting	1
	Review	1
	Chapter 7 Logic + Process	4
	ATAM Research	1
		<b>7</b>
es	Presentation	4
	Review other group	1
	Meeting	1
	Improvement system verification	1
		<b>7</b>
gk	Presentation	4
	Review other group	1
	Meeting	1
	Modified Chapter 5	2
	Process View	4
	Layout updates, improvements, timetracking improved	1
		<b>13</b>
mj	Added more chapter introductions	1
	Moved all current references to bibliography	1
	Resolved a lot of layout problems + rewritten some texts	2
	Moved several tables to Appendix	0.5
	System evolution	2
	Requirements validation + improvements to system evolution	6
	Software decisions	1
		<b>13.5</b>
cm	Meeting	2
	Review	1
	Improvements Logical View	5
	Layout	1
	Document review	4
		<b>12</b>
<b>Group total:</b>		<b>60.5</b>

Week 10

Person	Task	Hours
eb	Update, ESH Update Server, Backup, Facade	4
	Description database / edit database schema / events	4
		<b>8</b>
hp	ATAM Meeting	3
	Update financial model, webgrid and ATAM	2
	Update chapter 7 add text and MVC in web-interface	2
	Utility tree and table	1
	Software Evaluation Research and Writing	4
	System Evolution, review architecture verification, updated policies sequence diagram, updated text in web-interface.	2
	Fixing some jpg to pdf, fixed tables, fixed some spelling added conclusion to ATAM	1
		<b>15</b>
es	ATAM Meeting	3
	Improvements to system verification	3
		<b>6</b>
gk	ATAM Meeting	3
	Update Chapter 5	1
	Update Process View	2
		<b>6</b>
mj	Corrections for spelling and layout issues	2
	Improvements to architecture verification and software architecture	5
	Improvements to deployment view	1
		<b>8</b>
cm	Typo correction	4
	Consistency review	2
	ATAM Meeting	3
		<b>9</b>
<b>Group total:</b>		<b>30</b>

Total

Person	Hours
eb	<b>81</b>
hp	<b>102,5</b>
es	<b>52</b>
gk	<b>98</b>
mj	<b>139</b>
cm	<b>142</b>
<b>Group total:</b>	<b>614,5</b>