University of Groningen

Software Patterns
Team 2

Home Energy Monitoring System

Authors
Putra, Guntur
Fakambi, Aurélie
Schaefers, Joris
Menninga, Wouter

Monday 23rd November, 2015

Version 0.1

Authors

Name	E-Mail
Putra, Guntur	G.D.Putra@student.rug.nl
Fakambi, Aurélie	A.Fakambi@student.rug.nl
Schaefers, Joris	J.Schaefers@student.rug.nl
Menninga, Wouter	W.G.Menninga@student.rug.nl

Revision History

Version	Author	Date	Description
0.1	Schaefers	15-11-15	Added a stakeholders section
	Schaefers	15-11-15	Added a key drivers section
	Putra	15-11-15	Added Introduction and System context
	Fakambi	15-11-15	Added non-functional req. and risks
	Menninga	15-11-15	Added high-level and functional req.
	Menninga	15-11-15	Improvements to risks
0.2	Schaefers		
	Schaefers		
	Menninga	21-11-15	Improvements to High-level requirements
	Menninga	21-11-15	Improvements to Functional requirements
	Menninga	21-11-15	Added system context / elaborated model
	Putra	22-11-15	Added hardware architecture
	Putra	22-11-15	Edited introduction
	Fakambi		
	Menninga	22-11-15	Expanded system context / elaborated model
	Menninga	22-11-15	Improvements to use-cases

Contents

List of Figures

List of Tables

1 Introduction

This document depicts the software architecture of Home Energy Monitoring System as the first assignment of Software Patterns course. We proclaim ourself a software architect team working on the Home Energy Monitoring System (HEMS).

Energy is one of the main concern in household. The problem is how to monitor energy usage of a particular house in order to prevent useless consumption of energy and even how to save energy, resulting in lower household expenses for energy. Some examples of energy consumption for a house are shown in Figure 1.1. This system focuses only on the electricity usage, as one of the main energy consumption of a house.

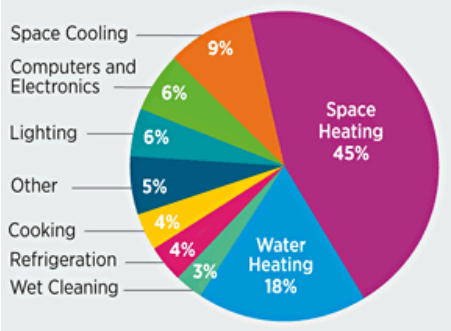

Figure 1.1: An example of home energy consumption [3].

This system allows the user to collect and store their electricity consumption. This system will also have a web dashboard that shows the graph of electricity consumption, which the user can adjust the time range of the graph to see different data, and other useful data such as which device uses the electricity the most, etc. The data, which will be stored in our database, will be used to do analysis that will help user to predict the upcoming electricity bill in the next month and other required analysis. This system also provides a fault-tolerant computing platform to compute these analyses.

The data collection is not our part, we leave this portion of the system to the third party developers. Basically, we provide a service of cloud computing environment, which can be categorized as SaaS[1], to do energy management system.

This system provides monitoring for more than one house. Many houses can connect to the system through Internet. Sensors will be located at every device that users are willing to monitor. Real time measurement of electricity consumption will be sent to the HEMS server via Internet protocol by the sensor.

As the sensors send the electricity measurement, no servers or any computing devices are needed to be installed in each house. Users may see the energy consumption through the web dashboard via computers, tablets, smartphones, or any devices that are able to connect to the Internet. ~~They may also place a TV or a screen in their house so that everyone in the house can see their energy consumption~~.

The rest of the document is structured as follows. The requirements, use cases, stake holders, and key drivers are explained in chapter 2. Chapter 3 describes the analysis and lists the possible pattens to be implemented in this system. Chapter 4 outlines the general system architecture of the system. Hardware selection and architecture are described in chapter 5, while chapter 6 elaborates on software architecture. The evaluation and evolution of this architecture is drawn in chapter 7 and 8.

[1] Software as a service

2 Requirements

This chapter describes our vision and uses it to derive stakeholders. This information helps us to be able to properly write use cases and stories. These will be used to extract functional, commercial, technical and evolution requirements. Afterwards, a risk assessment will take place, to ensure that the project is not at great risk.

2.1 Vision

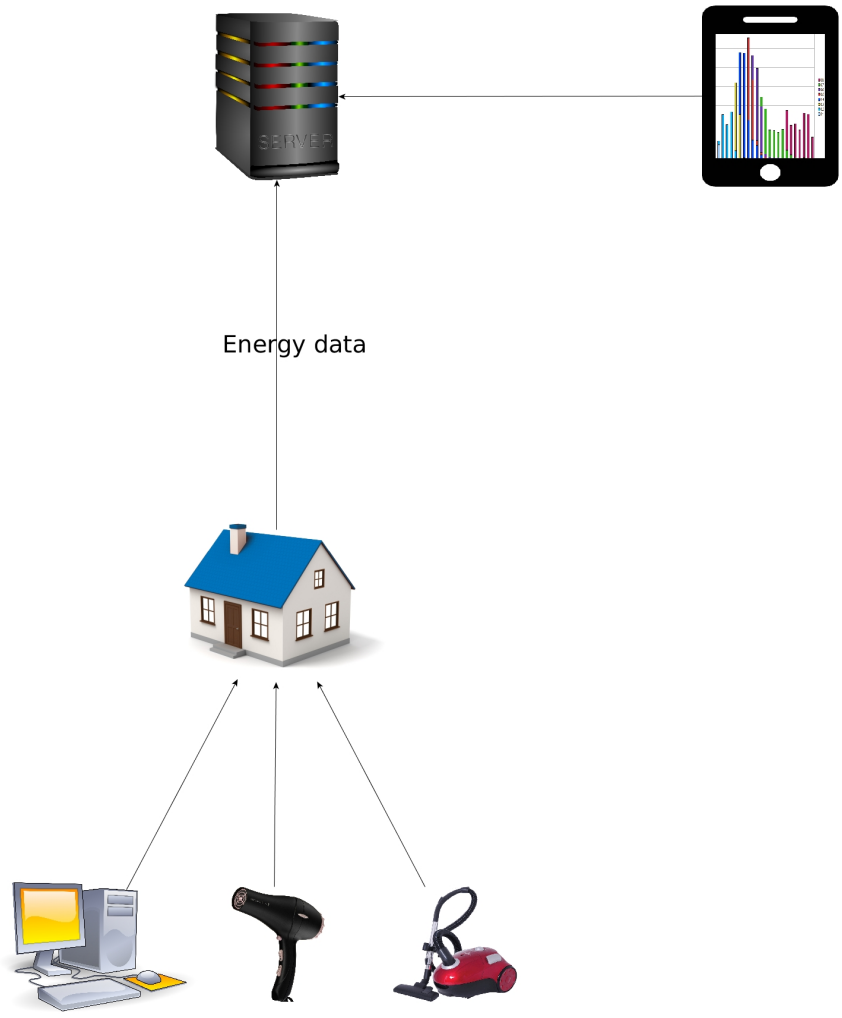

Figure 2.1: Architectural vision

2.2 Stakeholders and their concerns

This section defines all stakeholders of our system and describes the concerns of the stakeholders. A stakeholder might be a person, group of persons, or organization that are involved in our system. There are ten stakeholders, ranged from first parties to third parties stakeholders. Several quality standards from the "Software Requirements" book by Microsoft [4] are used. Those quality standards are described in Table 2.1.

Quality Attributes	Brief description
Availability	The extent to which the system's services are available when and where they are needed
Interoperability	How easily the system can interconnect and exchange data with other systems or components
Performance	How quickly and predictable the system responds to user inputs or other events
Reliability	How reliable the results of the system are (accuracy).
Security	How well the system protects against unauthorized access to the application and its data
Usability	How easy it is for people to learn, remember, and use the system

Table 2.1: Quality attributes of Software Architecture from "Software Requirements" Book [4].

There are six quality attributes, as can be seen in Table 2.1, for measuring stakeholders' concern regarding our system. Furthermore, profitability has been added as another quality standard to improve measuring stakeholders' concern. The stakeholders are listed below and are then explained in more detail below.

- Product owner
- Developers
- Maintainers
- Government
- Home owners

Product owner is the owner of the website. This stakeholder funds the website and its main concern is creating a profit. This affects the quality attributes usability and availability.

Developers have to make sure the system provides the functionality that the users expect the system to have. This means that their main concern is usability. But also interoperability is their main concern, because users will want to connect any energy consuming device to this site in order to monitor the consumption.

Maintainers are mainly concerned that the website is up and running at all times. Meaning their concern is the availability and security of the system.

The government wants to lower the energy consumption of the citizens. It wants to comply with the aims of the EU to reduce the energy consumption by 20% by 2020. Their main concern is the usability and reliability of the system, because the system is only useful to the government if it is used by allot of people and the statistics of the system are reliable.

Home owners want to get an insight in their energy usage. They want to know how to effectively reduce their energy consumption. In order to do this, they might want to receive alerts about sudden increases of energy consumption in certain devices. These alerts have to be accurate and reliable. The main concern of the home owners, thus, is the usability and reliability of the system.

2.3 Key-drivers

The key drivers of this system are:

1. Usability
2. Reliability
3. Availability

Usability has to be the main focus of the system. Users will want to stick to using our system if the usability is better then similar systems of the competitors. Even if competitors have a better availability and or

even reliability, there is a good chance that customers will stick to this system if the usability is better.

Reliability is very important, because for this kind of system the trust everyone has is crucial. Making the stakeholders trust the system is the most important aspect of this system. If the system at some points does not provide reliable data, without specifically informing about it, the entire system is useless.
This not necessarily mean that the system has to be very precise in calculating the statistics. Even if system might be off by 30%, as long as the user is aware of this and it never exceeds this marge of 30%, it is fine. The increases and decreases of the energy consumption will still be clear.

Availability is crucial because if the system isn't up at the times when it should, users will lose trust in the system. If a user set an alarm on a device, alarming them if the device exceeds a certain energy consumption, then ideally that alarm has to go any time the device indeed exceeds the given limit of energy consumption. Because of maintenance to the system or dependent systems, the up time of the system might not be 24/7 at all times. But if the system is not up at a certain point, the users have to know about this scheduled downtime.

2.4 High-level requirements

The high-level requirements describe the high-level functionality of the system. The high-level requirements are used to derive the functional requirements in Section 2.6. The high-level requirements are also used to classify the severity of the risks in Section 2.10.

This table uses different priorities according to RFC-2119[1]. First there is the 'must' ('required') priority, this requirement is an absolute must. The system could not function without the 'must' requirement. Besides 'must' there is the 'should' ('recommended') priority. This priority is highly desirable for the system, but the system could function without this requirement. The last priority is 'may' ('optional'), these requirements are nice to have but not really necessary for the functionality of the system. These different priorities are used through the rest of the document.

Nr.	Prio	Description
HL- 1	**Must**	**Collecting electricity usage data** The system must collect electricity usage data, which has to be stored so it can be used to compute statistics and detect changes in energy usage.
HL- 2	**Must**	**Computing usage statistics** The system must be capable of computing statistics about the energy usage, like total usage per month, but also periods of peak energy usage. It has to be able to compute such statistics not only per house, but also for individual devices.
HL- 3	**Must**	**Displaying statistics** Data which has been gathered and statistics that have been computed must be displayed to the user in an understandable way. One of the main goals of the system is to make users of the system aware about their energy usage. The data and statistics should be displayed in a way that is intuitive and easy to understand for average consumers.
HL- 4	**Must**	**Configuring the system** The users of the system must be able to configure the system, i.e. register their house/add new devices/subscribe to alerts etc., using the web interface.
HL- 5	**Must**	**User-friendly interface** The web interface of the system should be user-friendly and intuitive, such that an average consumer is able to use it.

Table 2.2: High Level Requirements

2.5 Stories and use-cases

In this section the architectural significant use-cases will be presented.

UC- 1	**~~The~~ End user registration**
Goal	The user has to register on the website to create his account.
Stakeholders and interests	**End user** – wants to be able to use the system to gain insight into his/her electricity usage
Primary actor	End-User
Level	User goal
Precondition	The website has the internal process to add an user to the database
Main success scenario	1. The user access the HEMS URL 2. The user click on the 'Sign up' Button 3. The website display a registration form requests the full name, username, email address and address of the user. 4. The system checks if the user isn't already in the database. 5. If not the user is added in the database. 6. The user receives a confirmation link on his email address and clicks on it. 7. The user is redirected to the HEMS website.
Postcondition	The account is created ~~: the user get access to his own Home Energy Monitor interface.~~
Extensions	4a. The user is already in the database. 1. The user gets to see a message telling him the user is already registered.
Related requirements	

Table 2.4: UC-1: The End user registration

UC- 2	**Receiving external energy usage data**
Goal	The sytem needs to receive energy data to make the analysis
Stakeholders & interests	
Primary actor	The energy usage sensors
Level	User goal
Precondition	The energy data is available
Main success scenario	
Postcondition	
Extensions	
Related requirements	

Table 2.6: UC-2: Receiving external energy usage data

UC- 3	**Store energy consumption**
Goal	The system needs to store the energy data it just received.
Stakeholders & interests	
Primary actor	The system
Scope	Monitoring part of the system
Level	Sub process
Precondition	The system has received the data. The database is available.
Main success scenario	1. The data received is directly stored in the database.
Postcondition	The data is stored in the database.
Extensions	The data isn't stored.
Related requirements	

Table 2.8: UC-3: Store energy consumption

UC- 4	**Display the estimated bill**
Goal	The End-User wants to see the estimation of his next bill
Stakeholders & interests	**End user** – wants to know about estimated bill for that month
Primary actor	End-User
Level	User goal
Precondition	The user is registered in the system.
Main success scenario	1. The user gets access to his interface by clicking on " Sign in " Button 2. In the menu he clicks on "Bill".
Postcondition	The estimated bill is displayed
Extensions	
Related requirements	

Table 2.10: UC-4: Display the estimated bill

UC- 5	**Display analysis, daily/weekly/monthly report**
Goal	The End-User wants to display several kind of analysis about hi energy consumption
Stakeholders & interests	**End user** – wants to know about the electricity usage
Primary actor	End-User
Level	User goal
Precondition	The user is registered in the system.
Main success scenario	1. The user gets access to his interface by clicking on " Sign in " Button 2. In the menu he clicks on " Analysis- Chart " 3. The system computes/make the analysis.
Postcondition	Charts with different item are displayed.
Extensions	The statistics aren't made.
Related requirements	

Table 2.12: UC-5: Display analysis, daily/weekly/monthly report

UC- 6	**Configuration**
Goal	The user must be able to configure the system with his own characteristics.
Stakeholders & interests	
Primary actor	The user
Level	User goal
Precondition	The user is registered in the database of the system.
Main success scenario	1. On the dashboard/website the user clicks on the " Settings " button. 2. The user gets access to the Settings section with all the possibilities/internal sections for example :frequency of alert, adding new devices etc.
Postcondition	
Extensions	2 The system didn't take into account the information the user just added. 2a. An error message is displayed " Configuration failed, would you like to try again Y/N ? ". 2b If yes Go to step1.
Related requirements	

Table 2.14: UC-6: Configuration

2.6 Functional requirements

This section lists the functional requirements of the system.

Nr.	Prio	Description
FR-1	**Must**	The system must be able to receive electricity usage data from devices.
FR-2	**Must**	The system must be able to store electricity usage data.
FR-3	**Must**	The system must be able to retrieve previously stored electricity usage data.
FR-4	**Must**	The system must be able to compute the total electricity consumption for a given time period for a particular house.
FR-5	**Must**	The system must be able to compute the electricity consumption per device for a given time period.
FR-6	**Must**	The system must be able to compute an estimated electricity bill for the current month based on the electricity consumption to that point.
FR-7	**Must**	A user of the system must be able to select which statistics to compute in a web interface.
FR-8	**Must**	The system must be able to show the computed statistics in a web interface.
FR-9	**Must**	The system must allow users to register an account on the web interface.
FR-10	**Must**	The system must require users to be logged in, before the user can view electricity usage information about his/her house.
FR-11	**Must**	A user of the system must be able to register a new house using the web interface.
FR-12	**Must**	A user of the system must be able to register a new device for his house using the web interface.
FR-13	**Must**	A user of the system must be able to configure the price of a kWH in the web interface.
FR-14	**Must**	The system must be able to send feedback to registered devices about the current electricity usage.
FR-15	**Must**	The system must be able to receive a statement about the accuracy form the sensors, such that when data might be inaccurate, the user can be informed about this.
FR-16	**Must**	The system must be able to display the history of electricity usage.
FR-17	Should	Users of the system should be able to subscribe to alerts in the web interface, alerting them about sudden energy increases or when they are using more energy than in previous months/weeks.
FR-18	Should	The system should send alerts to users by mail when the user is subscribed for this alert and the condition for the alert is met.
FR-19	Should	The system should be able to show the computed statistics in a graph.
FR-20	Should	The system should cache received usage data when the data storage becomes unavailable, so it can be stored later.

Table 2.15: Functional Requirements

2.7 Commercial non functional requirements

2.8 Technical non-functional requirements

2.8.1 Usability

The sytem will be hopefully used by a lot of people who don't necessarily have knowledge in the technology field so the Usability is an important requirement so the end-user can acces all the information he needs.

US- 1	An application is available for tablets and phone with those OS Windows Phone,Android, and Iphone.
US- 2	The end-user needs maximum ten minutes to get a basic understanding of system features through the UI.
US- 3	Every feature/major option of the system can be accessed through the home page(Single Page Application).

2.8.2 Reliability

RE- 1	A margin of error of 5% in the energy measurements is tolerated.

AVA- 1	The sytem should be available 99.9%. of the time which means down for 44 minutes.
AVA- 2	The system should be down for maximum ten minutes when the user installs a new release(version).

Availability

2.8.3 Security

Ensuring the security for the system is a major issue so all the data needed for its good functioning remain protected and consistent.

SEC- 1	Each user is identified and has to log in in order to view his "Home Energy Monitor" account.
SEC- 2	The data is encrypted.
SEC- 3	~~Passwords of the users are stored and transmitted as a hash~~

2.8.4 Scalability

The system should make the increase of workload, resources and users easy that's why the scalability is an important Key driver.

SCA- 1	The system should function as efficiently even if the number of users increases.

2.9 ~~Evolution requirements~~

2.10 Risk assessment

The system is confronted by several risks which are determined and mitigated in this section. Taking those risks into account allows to avoid them or at least reduce their impact. The risk management involves the identification of the risks, their probability and potential impact or consequences.

The tables below explain the meaning of the definition for probability and consequence.

Probability	Likelihood of occurrence
High	0.65 - 1.00
Medium	0.35 - 0.65
Low	0.00 - 0.35

Figure 2.2: The different probabilities used to classify risks

Severity	Explanation
Severe	A risk that can lead to loss of live or casualties.
Significant	A risk that can lead to damages, can delay the project more than 3 months or causes one of the high-level requirements not to be fulfilled.
Moderate	A risk that can lead to one of the high-level requirements not to be fulfilled to an acceptable level.
Minor	A risk that can lead to one of the high-level requirements not being fully fulfilled, but still fulfilled in an acceptable level.

Figure 2.3: The different severities used to classify risks

2.10.1 Technical

T-RISK1	**The statistics provided by the data processing framework are wrong**
Probability of occurrence	Low
Consequences	Moderate. If the statistics computed by the system are not accurate, this will lead to loss of thrust of the end user in the system.
Prevention	Make sure the algorithms used to compute the statistics are correct.
Reaction	Correct the algorithm, if the change is significant also inform end users about the error.

Table 2.21: T-RISK1 – The statistics provided by the data processing framework are wrong

T-RISK2	**The data center storing the energy measurements becomes unavailable**
Probability of occurrence	Low
Consequences	Significant
Prevention	Store the data in a redundant way, preferably in multiple data centers, so that one data center going offline does not lead to downtime of the system.
Reaction	If the data storage does become unavailable, new incoming data should be cached so it is not lost and users should be informed in the web interface that viewing the statistics is temporarily unavailable.

Table 2.22: T-RISK2 – The data center storing the energy measurements becomes unavailable

T-RISK3	**Somebody gains unauthorized access to someone else's data**
Probability of occurrence	Low
Consequences	Significant. Data about electricity usage can be used, e.g. to derive when people are home. Unauthorized data access will lead to a loss of thrust in the system by consumers.
Prevention	Make sure access to the data requires authentication using a password at all time. Enforce users to use a strong password.
Reaction	Make sure the unauthorized access is removed. Inform end users about which data was accessed by the unauthorized party.

Table 2.23: T-RISK3 – Somebody gains unauthorized access to someone else's data

2.10.2 Business

B-RISK1	**Wrong estimation of the budget**
Probability of occurrence	Medium
Consequences	Significant. The final product does not have the features expected.
Prevention	The team needs an accountant or at least someone taking care of the follow-up of the money. Make sure there are regular evaluations to keep track of the money flow.
Reaction	Remove some requirements or features of the product, or change the hardware components used.

Table 2.24: B-RISK1 – Wrong estimation of the budget

B-RISK2	**Wrong estimation of the budget: the money invested in the fabrication and achievement of the product/system is not covered by the sales (shortfall/deficit)**
Probability of occurrence	Medium
Consequences	Moderate. Stopping the sale
Prevention	The team needs an accountant or at least someone taking care of the follow-up of the money.
Reaction	Adding more features to the product in order to make it more competitive in the market.

Table 2.25: B-RISK2 – Wrong estimation of the budget: the money invested in the fabrication and achievement of the product/system is not covered by the sales (shortfall/deficit)

2.10.3 Schedule

S-RISK1	**The project is not finished at the deadline**
Probability of occurrence	Low
Consequences	Significant. Pressure for all the team members, loss of credibility regarding the customers, selling a product with less features than expected.
Prevention	SRA , Schedule Risk analysis : Estimation of the duration of the project by its manager (with the use of probability and statistics) . Meeting for the team members every week to keep track of the timing and take decisions according to the deadline.
Reaction	Postpone the deadline or remove some features when the deadline can't be postponed.

Table 2.26: S-RISK1 – The project is not finished at the deadline

3 Analysis

3.1 Layers

Context The system has to handle different kinds of incoming request. The web interface of the client wants to receive a web page from the system. The energy data collectors will send the data to the system so it can be stored and analyzed.

Problem The system needs a structure to coordinate the external calls and requests.

Solution Our application exposes a web service to multiple clients and so a the Service Layer pattern will be used. Using this pattern, the system will be divided into the following layers.

- Service layer
- Domain layer
- Data source layer

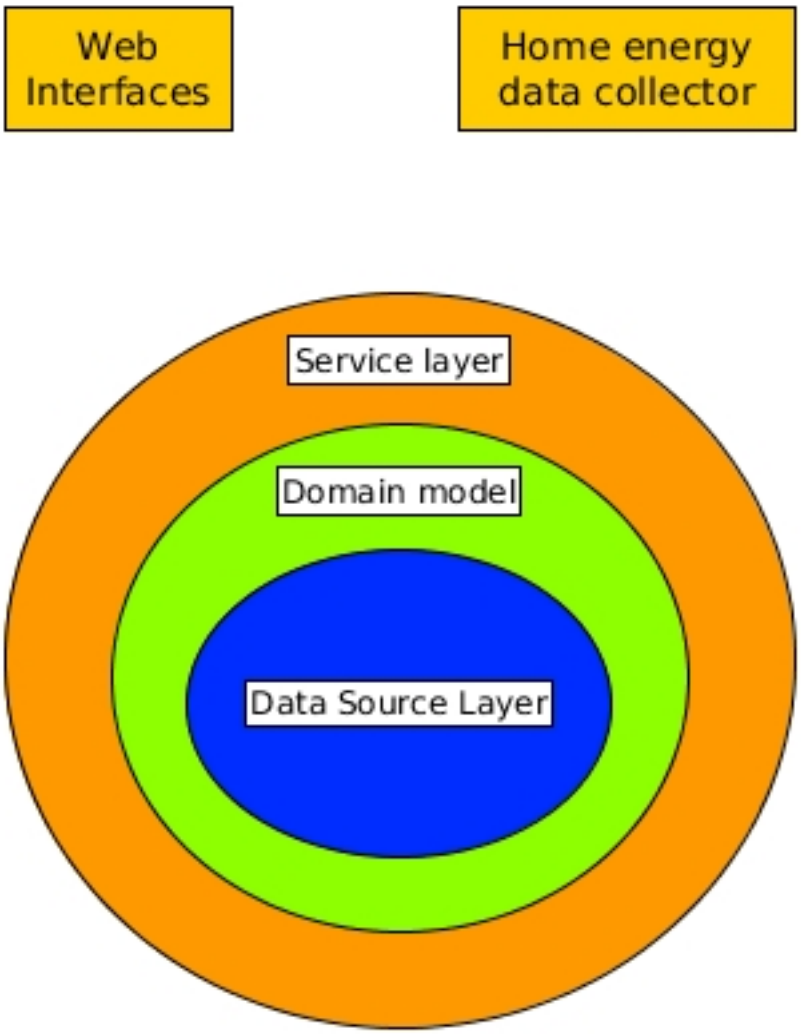

Figure 3.1: Service Layer

The service layer exposes a set of services to be used by clients and for each service, there is a certain

script that will be executed when the service is called. The service layer will be used with the "operation script" variation. This means that the Service Layer consists of thick classes that contain logic.

Source Patterns of Enterprise Application Architecture[2]

3.2 Layer Supertype

Context There is duplicate code within each layer of the system.

Problem Duplicate code makes the system hard to modify and maintain.

Solution With the Layer Supertype pattern, all the classes of a certain layer have the same super class. This super class then contains the features that are very common for the layer.

This pattern will be used in every layer.

Service layer All the services need to take care of security. The client needs to be authenticated and the data needs to be decryption by the service layer. All this security logic will be placed in a super type using the "Layer Supertype" pattern In the service layer it contains the security logic.

Domain layer In this layer, the sypertype will contain common features and functions for handling storage.

Data source layer The data mapper in the data source pattern can use a layer super type that handles all the common behavior, which can greatly reduce the extra work of coding.

3.3 Domain layer

Context The domain logic consists of complex functions for serving web request and analyzing data.

Problem The functionality of the system must be modifiable and must not contain duplicate code to prevent inconsistency.

Solution The domain logic is complex and so it requires the use of the domain model pattern. This means that the domain is Object Oriented, with every class representing one specific, individual, meaningful part. This is the most advanced pattern, reducing code duplication and increasing flexibility of the system.

Source

3.3.1 Unit of work

Context The system does statistical analysis on the sensor data

Problem Concurrent analysis functions might use the same object and data.

Solution Analyzing the data can be computationally intensive, which is why multiple threads will be used do distribute the work. This requires coordination of the data that is being handled. This pattern provides concurrency control. The Unit of work pattern: “Maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems.”

Source

3.4 Data source layer

The main database will be quite simple and the database actions won’t be very complicated. However, the tables needed for the statistics can become very complex.

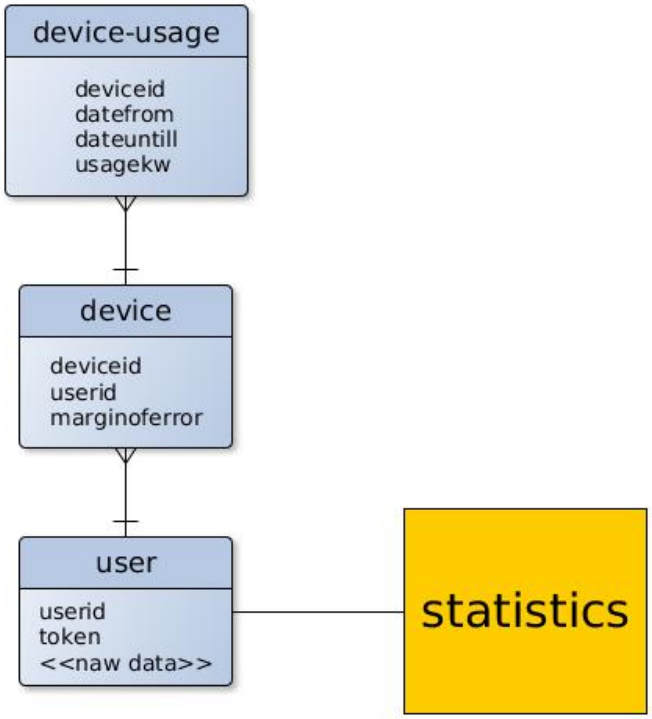

Figure 3.2: Database structure draft

Considered patterns to connect to data sources:

- Table data gateway
- Row data gateway
- Active record
- Data mapper

Because the domain model is used in the domain layer, the data mapper pattern is best to use . The logic that will operate on the data will generate statistics of the data, which can become quite complex. This is why the data mapper pattern is chosen as a pattern for connecting to the data sources. This pattern is the most advanced pattern, but provides the best functionality and abstraction.

With the active record pattern, the database access/communication, data and logic of that data is all stored in the same class. Since the logic that will be executed on the sensor data is very complex, this patterns will not be useful.

A gateway of any kind leads to performance overhead, because for each call coming from the domain model pattern, a database call is made. It lacks the necessary coordination.

3.4.1 Repository

The repository pattern mediates between the domain and data layer. The repository clients create a criteria object, specifying what kind of data is wanted. For example $criteria.equals(Device.NAME, Computer)$. Then the clients use this criteria by invoking repository.matching(criteria) to receive the data from the repository. The client just asks the data, it has no further knowledge about any interaction with any data source/data base.

The repository gives a lot more control over how the data is handled. The benefits are:

- Reduces code (and code complexity)
- Increases performance
- Separated domain and data layers, increasing flexibility and changeability

Performing analysis on the data also consists of executing complex queries on the data source. The database that executes these queries, however, might change. Or the system might decide to use multiple databases and data sources. Using the Repository pattern, these changes can be made fast. The repository also allows for multiple configurations to exist. So an extra repository could be created for testing purposes, only using an in-memory database to increase the test execution speed.

3.5 Client user interface

3.5.1 Controller

Page controller: controller for each page. So upon receiving a request do:

- decode url, extract data
- invoke model to process data
- determine view and use the model data to create the HTML to return

Front page controller:
One controller for all requests/views. This allows building a filter chain, handling authentication, logging etc. Front page is better/helps with concurrency, because a new command object is created on each request. Reducing thread-safety concerns. The model, however, can have shared objects that do require thread safety management.

The front page will be used, because it provides more functionality to the system. The only advantage of a page controller compared to the front controller is that it has a more natural structure.

3.5.2 View

Template view Write HTML code including markers. Replace the markers with the data when the page is requested. (Play framework)

Transform view Convert the domain data to HTML, "transform" the domain data. Upon a request, it get the domain data, for each item in the data it looks for a appropriate "transform" to transform the data to HTML.

The template view will be used, because it is used a lot more then the transform view. Major web frameworks are based on this pattern (the play framework, laravel...). The view patterns don't have an important difference in how beneficial they are to the project.

3.5.3 Model

This section will describe how the Model updates the view and how the view updates the model.

To consider:

Observer Synchronization Synchronize multiple screens by having them all be observers to a shared area of domain data.

Separated Presentation Ensure that any code that manipulates presentation only manipulates presentation, pushing all domain and data source logic into clearly separated areas of the program.

Presentation model Represent the state and behavior of the presentation independently of the GUI controls used in the interface

Supervising controller Factor the UI into a view and controller where the view handles simple mapping to the underlying model and the the controller handles input response and complex view logic.

Model view controller/presenter

4 System architecture

This chapter outlines the general system architecture of the system. The first section shows the system with its inputs and outputs to and from external factors.

4.1 System context

The system context diagram in Figure 4.1 gives an overview of the entities that interact with the system.

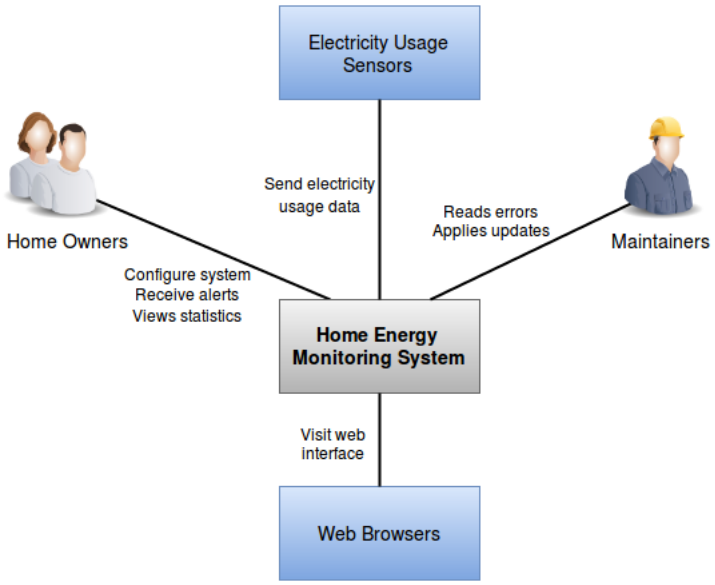

Figure 4.1: System context diagram

4.1.1 Users and roles

Home Owners

The home owners are the main users of the system. They want to know about the electricity usage in their house.

They interact with the system by viewing statistics, configuring the system and they receive alerts if they configured the web interface to send them these.

Maintainers

The maintainers of the system will apply updates to the system and read errors that might have occurred in order to solve these.

4.1.2 External systems

Electricity Usage Sensors

The electricity usage sensors are the sensors that measure the electricity usage of the devices. These sensors work by measuring the electricity passing through a power outlet. This means that they in fact measure the electricity usage of a power outlet and not that of a device. This is relevant if multiple devices are connected to the same power outlet .

Web Browsers

The web interface of the system, where the system can be configured and statistics can be viewed, is presented as a web page. Users will use a range of different web browsers (on a range of different devices)

to visit this web interface. It is important that the web interface works equally well in all these different web browser.

4.2 Elaborated Model

In Figure 4.2 the elaborated system model is shown. This figure gives a high-level overview of the software and hardware of the system. In this figure, the arrows represent the data flow.

Each house sends collected electricity usage data to the systems 'Data receival API', which stores the collected data in a storage cluster. An end-user of the system can visit the web interface using a web browser on any device. Using this web interface, the user can generate statistics data, which is done by a separate software component of the system.

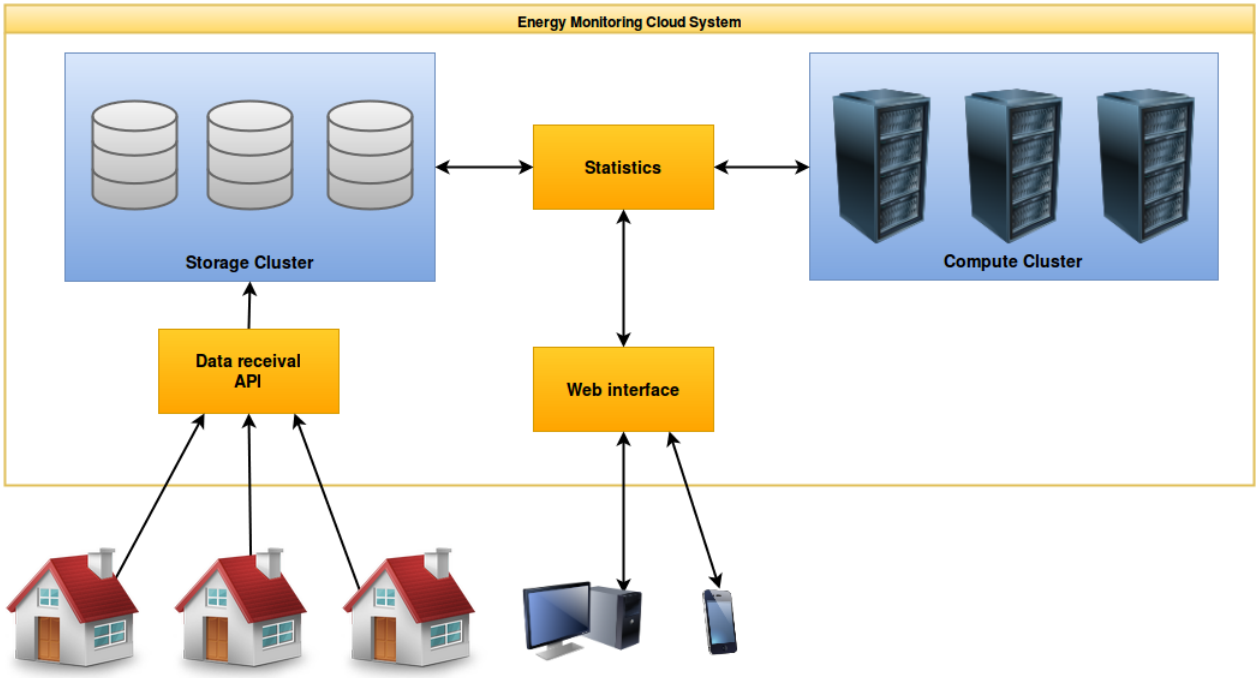

Figure 4.2: Elaborated model

Data receival API
The 'Data receival API' component exposes a REST interface, which is used by the homes to send electricity usage data to the system. The API is responsible for storing the data in the storage cluster or caching the data if the storage cluster is temporarily not available.

Storage cluster
The storage cluster is a set of servers which will store all the data in a replicated way.

Web interface
Users of the system (home owners) visit the web interface (presented as a website) through all kinds of devices. The web interface allows users to ~~compute and~~ view statistics from the collected usage data. The web interface also allows users to register new devices with the system.

Statistics
The statistics part of the system is responsible for computing statistics from the collected usage data. It will receive commands to do so through the web interface.

Compute cluster
The compute cluster is a cluster of servers, which are used to compute statistics from the raw collected electricity usage data.

5 Hardware Architecture

This section describes the hardware architecture of Home Energy Management System (HEMS). The description will be more high-level along with explanations about the hardware platform and the application interfaces between each components of the system. The rest of this chapter is organized as follows; First section, section 5.1, presents an overview of the hardware implemented in this system depicted in big schema. Decisions made in this system are detailed in section 5.2 with tables. Lastly, the hardware is described in section 5.3.

5.1 Hardware Overview

As mentioned in previous chapters, HEMS focuses on providing services to monitor electricity usages based on installed sensors on each customer's house. Thus, this system works on the cloud part, which is providing data storage, monitoring, and analysis, both in terms of application (software or service) and hardware. This system deals with no electricity collecting devices. Therefore, the electricity collection part is delegated to third party developers.

According to chapter system architecture, the main part of the HEMS hardware consists of storage cluster and compute cluster. The storage cluster is responsible to handle incoming data from sensors through the exposed data acquisition API. This cluster is capable to store real-time data. The compute cluster mainly deals the data presentation of the stored usage data. Furthermore, compute cluster is also needed to perform analysis based on stored data.

The detailed hardware selection to perform and build this system is explained in the following section.

5.2 Hardware Design Decisions

This section defines decisions made regarding the hardware selection. Tables will be used to make our justification in regard to hardware selection more clear.

Name	**Compute cluster selection**
Decision	**HW- 1**
Status	**Approved**
Problem/Issue	HEMS needs a reliable computers to do the analytical processing.
Decision	HEMS will use clustered Dell PowerEdge R530 to act as the main analytic cluster and to provide API to the actors.
Alternatives	*HP ProLiant DL360 Gen9 Base* This server rack has 16GB of memory and 2.4GHz of processor speed. As other server computer, this machine utilizes Intel Xeon E5 2600v3. This server is suitable for high dense computing, however the price is not so suitable for this kind of specification. It does not have LCD screen that will help technician to look the current status of the server. *Lenovo System x3550 M4 7914* This server rack has only 8GB of memory. However, the processor is a bit faster, it runs on 2.6GHz. As other server computer, this machine also utilizes Intel XEON E5-2600. The price is a little bit lower than the others but the memory limitation makes it not so valuable. It has LCD screen that will help technician to look the current status of the server. *Dell PowerEdge R530* This 2U server rack has 16GB of memory and 2.4GHz of processor speed. This machine utilizes Intel Xeon E5-2620V3 with 15MB of cache. This server is suitable for high dense computing. It has LCD screen that will help technician to look the current status of the server.
Arguments	

	Reliability	Performance	Interoperability	Security	Scalability	Cost	**Score**
Weights 3	2	3	2	2	2	2	
Dell PowerEdge R530 5	3	5	4	4	4	5	70
Lenovo System x3550 M4 7914 4	3	4	4	3	4	4	60
HP ProLiant DL360 Gen9 Base 5	3	5	4	3	4	3	64

Table 5.1: Decision – Analytic cluster selection

Name	**Storage cluster selection**
Decision	**HW- 2**
Status	**Approved**
Problem/Issue	The system needs reliable computers to store the data.
Decision	HEMS will utilize Synology RackStation RS814RP to store the data.
Alternatives	*Synology RackStation RS814RP* This storage machine has the fastest connection among the others. This machine will run at SATA with 6 Gbps connection. *70BJ NAS-server* This machine form factor is 1U which is suitable for saving space. However, the connection speed is limited to 3 Gbps. *Thecus N8810U-G NAS-server* This machine also runs in 3Gbps connection. However, the form factor is 2U which makes this machine takes more space in the rack.

Arguments

	Reliability	Performance	Interoperability	Security	Scalability	Cost	**Score**
Weights 3	2	3	2	2	2	2	
Synology RackStation RS814RP 5	2	4	4	4	4	4	63
70BJ NAS-server 4	2	3	4	4	4	3	55
Thecus N8810U-G NAS-server 4	2	3	4	4	4	3	55

Table 5.2: Decision – Choice of storage machine

5.3 Hardware Description

This section gives an outline of the hardware implemented in this system. This section also elaborates on hardware decisions.

5.3.1 Storage cluster

HEMS will use clusters of computer to manage the database system and to store our data. The cluster enables the database to replicate data. This means no backup is needed, the data is always available in multiple servers. Furthermore, user account information is hashed using bcrypt after being salted with 128 randomly generated characters to increase security. The cluster will also be accessible by the main analytics part as the data will come and go through the main analytics part. The logical schematic of the storage cluster is depicted in Figure 5.1

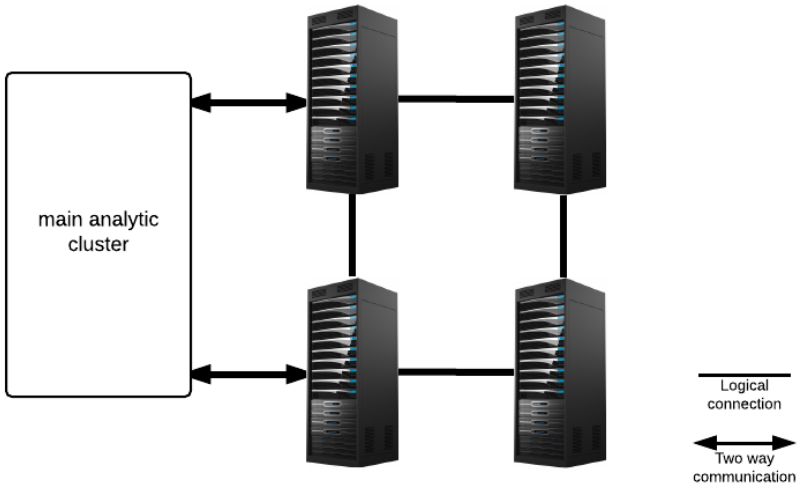

Figure 5.1: Logical schematic of storage cluster of HEMS

As can be seen in Figure 5.1, HEMS will use four database racks to have redundancy in the system. The server is connected as a ring, which is the common way to setup database server. By using this form of architecture, HEMS will be more reliable and fault tolerant. There will also be two physical connection to the main analytic cluster to make this system more fault tolerant in terms of connection. HEMS database cluster will use the same server, Dell PowerEdge R530, for controlling the SATA storage machine.

5.3.2 Compute cluster

Compute cluster will be the main brain of HEMS. The data presentation will be handled by this system. The analysis process also runs on top of this hardware. To increase availability and reliability, HEMS will have six server racks to do the processing as depicted in Figure 5.2.

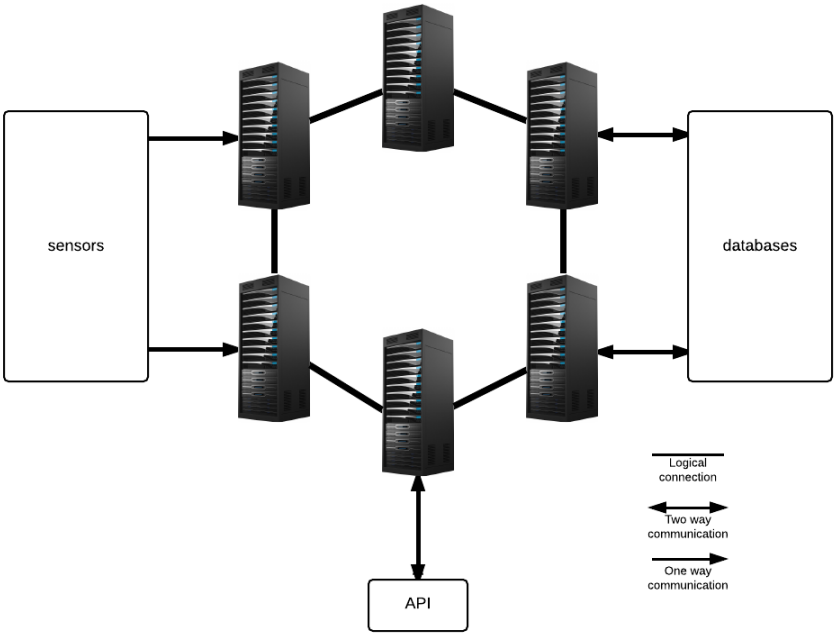

Figure 5.2: Logical schematic of analytic cluster of HEMS

6 Software Architecture

7 Architecture evaluation

8 System evolution

A Time Tracking

Week 1

Person	Task	Hours
Putra	Meetings, writing introduction and system context.	4
Fakambi	Meeting, First approach on non functional requirements and risks	4
Schaefers	Meetings, Added a stake holders and key drivers description.	5
Menninga	Meeting, high-level and functional requirements and improvements to risks	5

Week 2

Person	Task	Hours
Putra	Meetings, editing introduction and adding hardware architecture.	5
Fakambi	Meetings , work on the use-cases	6
Schaefers	Meetings, researched patterns to use and created initial analysis draft	9
Menninga	Meetings, system architecture, use case improvements	8

Bibliography

[1] S. Bradner. Key words for use in RFCs to indicate requirement levels, 3 1997. RFC 2119.

[2] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[3] Rik Langendoen. How to reduce home energy-related costs. `http://greenifynow.com/wp/?p=795`, 2013. [Online; accessed 22-November-2015].

[4] K. Wiegers and J. Beatty. *Software Requirements*. Developer Best Practices. Pearson Education, 2013.