



university of
 groningen

faculty of mathematics
and natural sciences

Mobile visualisation of energy consumption data for improving awareness amongst building occupants

Master's Thesis

Student: M. Meiboom

First supervisor: Prof. A. Lazovik

Second supervisor: Prof. A. Ampatzoglou

Date: August 2013

Contents

List of figures	iv
List of code listings	vii
1 Introduction	1
1.1 Scope of this thesis	1
1.2 Problem statement	2
1.3 Outline	3
2 Related work	5
I Foundations	7
3 Eco-visualisation	9
3.1 Providing effective feedback	9
3.2 Using social incentives	11
4 Data modelling	13
4.1 Modelling	13
4.1.1 Missing data	15
4.2 Data acquisition	15
4.2.1 Plugwise data	16
4.2.2 Blackbox data	16
4.3 Storage of data	18
4.3.1 Modelling the environment	18
4.3.2 Modelling measurements	22
5 Querying data	25
5.1 Inserting data into the graph	25
5.2 Extracting data from the graph	27

II	Application Design	29
6	General design and goals	31
6.1	Goals and functionality	31
6.2	Design goals	32
7	Mobile Client	35
7.1	The mobile context	35
7.2	User experience	36
7.2.1	Heuristics	36
7.3	Resources and data transfer	37
7.3.1	Reducing radio load by reducing the amount of requests . .	40
7.3.2	Reducing the number of individual transfers	40
7.3.3	Utilizing content providers	41
7.4	Visualising charts	42
7.5	Effective navigation	43
8	Expanding visualisations	49
8.1	Hallway display	49
8.2	Advanced visualisation	51
III	Empirical evaluation	53
9	Methodology	55
9.1	Case study	55
9.2	Observations	57
9.3	Analysis	57
10	Results	59
10.1	Exploratory tasks	59
10.2	Questionnaire	59
11	Discussion	61
11.1	Exploratory tasks	61
11.2	Questionnaire and other observations	62
12	Conclusion	63
13	Further research	65
	Bibliography	66
	Appendices	71

List of Figures

3.1	Heuristic model for behavioral change	10
3.2	Visualisation of a scoreboard	12
4.1	Visualisation pipeline	13
4.2	Chart showing a series of time/value measurements.	15
4.3	Data flow for importing raw data from Collector	17
4.4	Data flow for importing raw data from SMOG	17
4.5	Map of the 5th floor in our test environment	19
4.6	Graph representation of the Bernoulliborg living lab	20
4.7	Example measurements and their relation to our topology	22
4.8	Example measurements and their relation to time	23
4.9	Complete database ontology for the GreenMind application	24
7.1	OSI Layered network architecture	38
7.2	State machine for a 3G wireless radio	39
7.3	3G radio state for different request strategies	39
7.4	Using ContentProvider for abstracting efficient data retrieval	43
7.5	Chart rendered using webview	44
7.6	Chart rendered using native library	44
7.7	Relation between visualisation pipeline and mobile app	45
7.8	Partial overview of GreenMind navigation and flow	47
8.1	Relation between visualisation pipeline and hallway display	50
9.1	GUI presented in the user study	57

List of code listings

4.1	A sample record of raw Plugwise data	16
4.2	A sample record of pulse data collected by the Blackbox	17
5.1	Insert measurement into graph structure	26
5.2	Find/create year, month, day and hour vertices for a timestamp t .	26
5.3	Finding occupants for a location	27
5.4	Finding devices for a person	27
5.5	Finding measurements for a device since the start of this month . .	27
5.6	Getting logs for a device for a specific interval	28
8.1	Mapping function for the hallway display	50

In 2014, the University of Groningen will be celebrating its 400 year anniversary. For one month, the University will present students, alumni, residents of Groningen, guests from other universities and many more with a program embracing knowledge, culture and sports. It is during this anniversary that the University wishes to highlight its focus on research areas such as Energy, Healthy Ageing and Sustainability. Hence the theme for these celebrations: For Infinity (4∞).

Anticipating this anniversary, the Sustainability Steering Group of the University of Groningen has asked University staff and students to submit ideas for improving sustainability in both business operations and buildings. A jury awarded the submission from the Distributed Systems group [1], written by Faris Nizamic and Tuan Anh Nguyen, with the Green Mind Award.¹ Part of this award included funding to realize the plans, aiming at water savings, waste reduction and energy savings. The ultimate goal of the plans is to make the Bernoulliborg building and the experience it provides to its occupants, respect sustainability principles. This thesis is the result of implementing parts of the proposed solutions related to monitoring energy consumption. A full description of these ideas can be found in the original proposal [1].

1.1 Scope of this thesis

One of the solutions proposed for the Bernoulliborg building entails measuring power consumption and displaying it in real-time on hallway displays. Occupants are expected to become more aware of their energy consumption by reminding them of the resources they use. As a result, energy consumption is expected to go down. As an addition to hallway displays, a mobile application is desired that would allow individual users to review their energy consumption on a personal level. Such an application may further remind users of their energy consumption and motivate them to help obtaining the goals set for reducing it. The data supporting these applications is obtained from two distinct sources. Global energy consumption is collected by a measuring device linked to the building's energy meter. Consumption for individual devices is measured

¹<http://www.rug.nl/about-us/who-are-we/sustainability/green-mind-award/>

by placing hardware sensors between devices and the power outlets they use.

In this thesis, we will discuss the theoretical foundations supporting the design of the aforementioned mobile application and related components. This includes the integration with systems responsible for measuring, storing and processing visualization data. Throughout the discussion we will regularly use the GreenMind project as a practical example, though the aspects we discuss can be applied in a wider sense. Thus, we will be using the development of the pilot application as a case to illustrate theoretical concepts. This involves a number of aspects:

1. Dealing with potentially large quantities of data from which we want to be able to extract information.
2. Providing effective visualisations of this data, focusing primarily on a mobile context.
3. Designing an application that will keep users involved over a longer period of time.

We will revisit these aspects in later chapters to elaborate further on them. You may have noticed that we did not mention the aspect of performing actual data measurements. This is out of scope for this thesis, though we will briefly discuss the different sources of our data in chapter 4.2.

1.2 Problem statement

Effective data visualisation in a mobile context poses a number of serious complications. This is a direct result of the characteristics of mobile devices, such as limited screen size, limited connectivity and the way in which people use their device. Existing research into mobile visualisation is typically outdated as a result of the rapid evolution of mobile device capabilities. Besides these complications, there is also a lack of high-quality native libraries that support mobile data visualisation. As a result, development tends to require a large amount of effort for basic implementation tasks, distracting from higher level goals such as user experience. Also, development of a mobile application parallel to other tools visualising the same dataset tends to lead to repetition of code and inefficient use of resources. Finally, previous work often does not take into account efficient retrieval of data in a context where we may not be able to reliably connect to a remote source of data.

We will present a solution to these problems by first analysing aspects of the mobile context, the visualisation pipeline, our data model and effective visualisation. We then continue by building upon this foundation, linking individual

aspects together and using the results to discuss the design of the GreenMind mobile application. Finally, we will present a small-scale empirical evaluation of the application. The outcome of this evaluation may help us determine the degree in which our solution supports the goals of the application. It may also provide insights into potential further research.

1.3 Outline

The rest of this thesis is divided into three major parts. Part I consists of the theoretical foundations on which the application was built. Chapter 3 introduces the subject of visualisation of temporal data. Chapter 4 discusses the characteristics of the data used for our visualisation such as data acquisition and storage. Chapter 5 presents operations specifically designed to deal with the size of our dataset.

Part II focuses on the design of the GreenMind (mobile) application, using the foundations discussed in the preceding chapters. Chapter 6 discusses the goals of the application in further detail, which we need to understand before discussing its structure in chapter 7 and a number of visualisation techniques in chapter 8.

Finally, part III presents the setup and results of the empirical evaluation, linking the application design to the experience of the end user. We discuss the setup of this evaluation in Chapter 9. The results are presented and discussed in chapters 10 and 11 respectively.

We conclude with chapters 12 and 13 by summarising our findings and making a number of suggestions for further research.

This thesis largely relies on the GreenMind dataset, which consists of electricity consumption measurements over time. Research into working with such a dataset is abundant and discusses issues such as dataset characteristics [2] and the estimation of missing data [3]. More recent is the work regarding the use of graph databases for storing and working with time series data. In their book *Graph Databases* [4], Robinson et al. provide a comprehensive overview of the characteristics of this relatively new type of Database Management System (DBMS). Vicknair et al. [5] and Holzschuher et al. [6] show that such a DBMS is particularly effective when used with highly interconnected data (such as social structures or building topologies) as well as data structures that can be modelled as a tree, such as time-series [7].

While we will mostly limit ourselves to basic visualisations, there are a number of papers discussing interesting advanced techniques. Lin et al. [8] introduce a tool that uses symbolic representation of time series called Symbolic Aggregate approximation (SAX). This tool allows for the analysis of large datasets and is able to discover both recurring patterns as well as anomalies (irregular patterns) in time series data. Lin et al. also discuss a number of techniques that stimulate visual discovery of patterns in a time series such as cluster and calendar-based visualization, which are discussed in more detail by Aigner et al. [9] amongst others [10]. We will revisit these techniques in chapter 8.

Research by T. Holmes [11] poses the question how visualisation of a building's carbon footprint might increase awareness of this footprint and stimulate residents to be more conservative in their energy consumption. It also focuses on the question of which strategies are most suited for providing such a visualisation. This interdisciplinary approach to the problem from the fields of HCI, art and visualisation was aptly dubbed *eco-visualisation*.

On the 2013 conference on Human-Computer Interaction (CHI2013), Valkanova et al. presented their research on eco-visualisation in a public context [12]. In their research, they discuss how publicly accessible displays influenced the perception and discussion of both personal as well as communal data. They conclude their research by discussing the implications of a public visualization as a tool to increase awareness, a subject that we will revisit in section 3.2.

The other areas of focus of this thesis, mobile visualisation and mobile usa-

bility, have been subject of research over a long period of time. Research into mobile (data) visualisation such as that of L. Chittaro provides a basic introduction to the limitation of the mobile context [13], [14]. However, rapid changes in the capabilities of mobile devices warrant additional research. Nielsen et al. attempt to provide a handbook for creating a satisfactory user experience in a mobile context [15], which is based on earlier research into the area of usability. On the other side of the development cycle, Zhang et al. provide a toolkit for verifying the user experience of mobile applications [16]. This toolkit is still in development, but proves an interesting alternative for performing small-scale user studies.

We end this chapter by noting that on the HCI2013 conference (also on Human-Computer Interaction), several of the sessions were focused specifically at creating a user experience to facilitate a positive change in user behavior. While the proceedings of this conference were not available at the time of writing, this shows that there is a large amount of ongoing research on the subject of persuasive interaction and eco-visualisation.

Part I

Foundations

In chapter 1 we have introduced both the GreenMind project as well as the scope for this thesis. Our final objective is to provide an application allowing users to obtain insights from visualised data. We begin, however, by laying a foundation to build upon. We will do this in the following chapters, beginning with a basic discussion on data visualisation and behavioral change.

3.1 Providing effective feedback

To provide an effective visualisation, we have to ask ourselves what kind of feedback model is most effective for the intended goal: promoting sustainable consumption. Previous studies in this area show that this is a particularly difficult subject. Electricity (as a commodity) differs in a number of ways from other consumer goods: it is abstract, invisible and untouchable [17]. Electricity is consumed as a result of other activities such as making phone calls, working on a computer or heating a meal in the microwave. This makes it hard to relate activities to opportunities for conserving energy.

Another issue resulting from the abstract commodity ‘electricity’ is that consumers do not directly experience the effects of using it. Therefore, consumers lack an emotional involvement and are likely to view energy as a necessary but ordinary product. This makes it even more difficult to try and make sustainable behavior part of a lifestyle [18]. An effective visualisation will have to deal with these issues.

Research in the area of behavioral change by Matthies (2005) resulted in a heuristic model of the process of changing environmentally related behavior. This model is depicted in figure 3.1. The detrimental habits mentioned in this model consist of behavior that has a negative effect on the environment. Matthies argues that such behavior is typically habitual behavior: it is part of a routine and does not require us to think. A lot of our daily activities that consume electricity (such as turning on a light, connecting a mobile charger) are examples of habitual behavior. While such behavior saves us the effort of making conscious decisions, it may also be detrimental to our efforts to improve sustainability. Changing such habits requires us to make a conscious decision to break a habit and evaluate possible alternatives. This process of changing habits is exactly what we aim to do

through visualisation. Therefore, the lessons learned on how this process works are relevant to providing effective feedback. We will briefly discuss some of the steps involved, though for a more elaborate discussion consider reading [17]. For results on an experiment focussing specifically on building occupants, we suggest looking at a discussion on the results of an experiment conducted by the Ruhr-University Bochum.¹

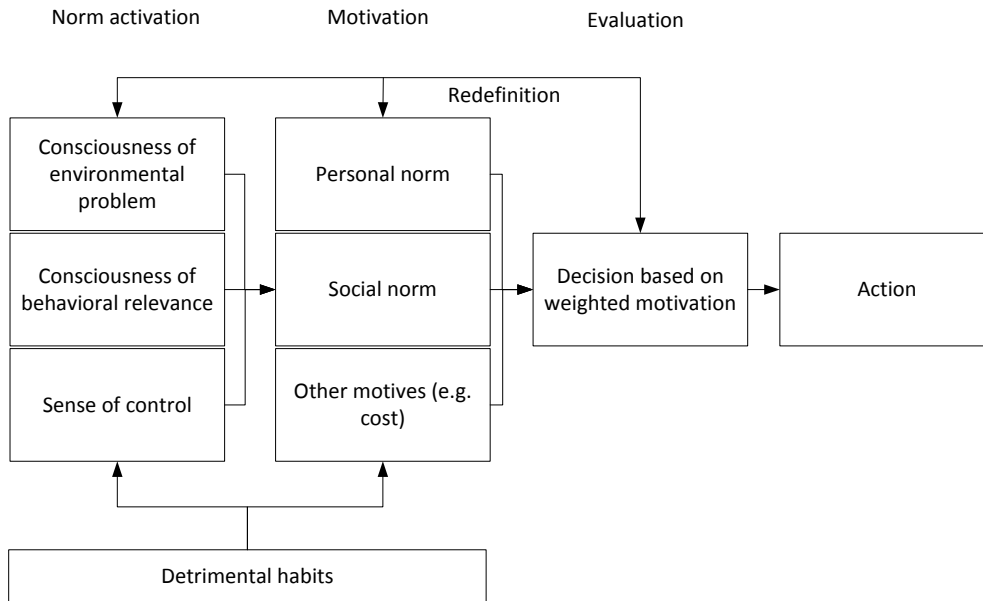


Figure 3.1: Heuristic model for behavioral change (source: Matthies 2005)

Norm activation

Because habitual behavior implies that we are not consciously exhibiting that behavior, changing such behavior requires a trigger. This is called *norm activation* and consists of three components. First, we need to realize that there is an (environmental) problem. Secondly, we need to be able to relate this problem to our own behavior. Finally we need to be able to find alternative behavior that helps solve the problem. This final step is typically called *sense of control*. If a user does not have this sense of control, it is impossible to make a decision to change behavior. For example, a user may recognize a problem ("I am using above average amounts of energy on heating") but feel that the solution to this problem is out of his/her control ("The housing office refuses to put in double glazed windows").

Motivation

Being able to identify a problem and alternative behavior is typically not en-

¹<http://www.his.de/change-energie/projekt/download/Energy%20Consumption%20in%20Organizations.pdf>

ough to result in actual change. Users will first enter a decision-making process in which motives are used as a basis to evaluate the alternative. Motives result from norms we deem important, such as social norms (conforming to what is socially acceptable) or personal norms (e.g. a desire for comfort). These norms may be conflicting and a weighted decision is made using the information available as well as the value we assign to our norms. Thus, *information is critical* in order to make a decision.

By considering the aspects of this behavioral model, we may try to determine how feedback will affect it. The goal of our visualisation then becomes to *trigger norm activation* as well as *influence the decision made* towards a positive change in behavior. Fischer et al. [17] have compared a large number of projects involving visualisation to infer when feedback is most effective. The selection of visualisations for the GreenMind application was partially based on their conclusions. For a comparison between several types of feedback please refer to the work of Fischer et al. For the GreenMind application the following strategy was selected:

- Provide normative comparisons to stimulate social incentives
- Provide historical comparisons to illustrate effects of behavioral change
- Provide breakdowns of the data for specific rooms, appliances and time intervals to stimulate sense of control
- Provide frequent feedback to increase consciousness

3.2 Using social incentives

Recent research in the use of social incentives as a technique to influence the decision process shows that users are very susceptible to changing behavior as a result of social motives ([19], [20]). An experiment at the Indiana University at Bloomington campus confirms these findings. The university placed public displays showing individual energy consumption in student dormitories (see figure 3.2) conducted a series of interviews with students. Nearly half of the students (44.2%) indicated that they participated as a result of seeing others participate as well. The researchers concluded that without a strong social incentive, it would have been unlikely for students to participate over a longer period of time. The general respons also largely confirmed the model we presented in figure 3.1 and the need for a sense of control.

There is a major drawback to extensive use of social incentives, however. While a student campus may be a location where the introduction of an element of competition motivates participation, other demographic groups may be apprehensive with sharing personal data publicly. For example, employees may be

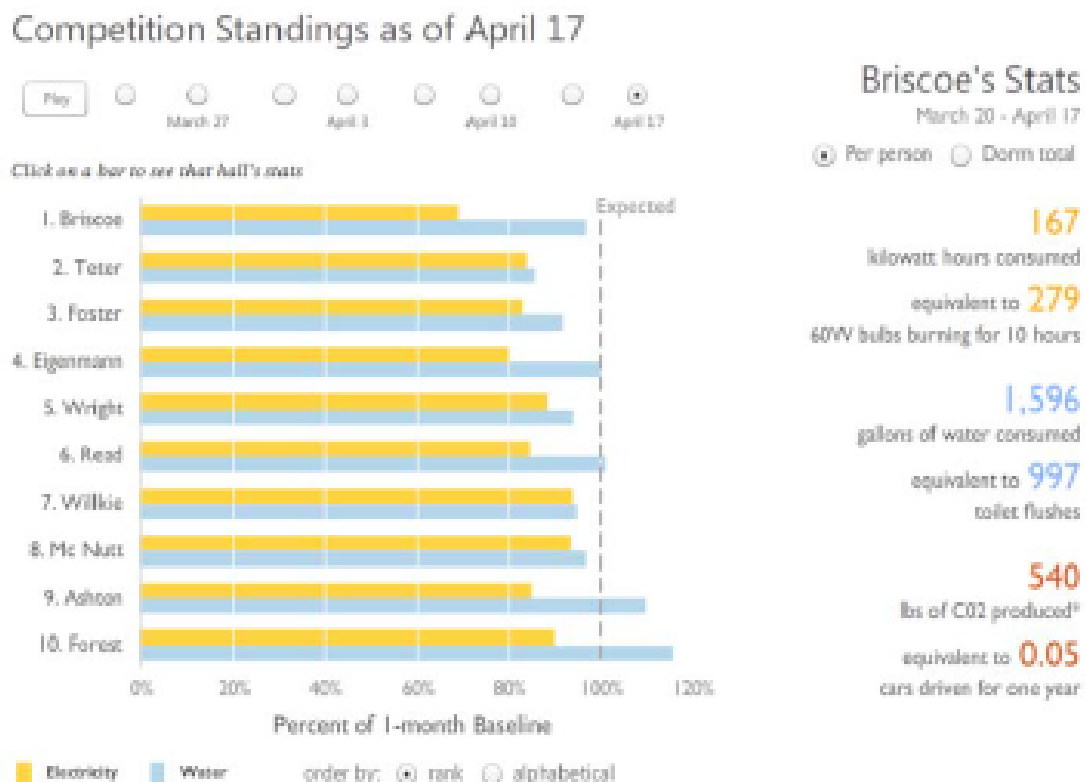


Figure 3.2: Visualisation of a scoreboard

The scoreboard attempts to trigger social motives by comparing individuals with their peers

averse against sharing knowledge about their personal consumption within their place of work. A balance between privacy and the use social incentives should therefor be made based on the target demographic.

In the case of the GreenMind project, the choice was made to introduce the concept of an *average employee* and let people compare themselves (and only themselves!) to this average user through the mobile client. As a result, the publicly accessible hallway display was decided to only display aggregated data.

Chapter 4

Data modelling

In the previous chapter we have discussed visualisation from an end user perspective. To get to the point where we can provide that user with an appropriate visualisation, however, we need to start from raw data. In this chapter we will continue with an analysis of the characteristics of the GreenMind dataset. These characteristics will influence the decisions we make later on. Failing to properly analyse them may lead to ineffective visual representation of that data. This may in turn lead to false interpretations [9]. To add some structure to the discussion, we will use the visualisation pipeline to relate between the different concepts we discuss.

The visualisation pipeline (see figure 4.1) describes the (step-wise) process of creating a visual representation out of raw data. By dividing this process into steps, we are able to separate different concerns and create a modular system. Separate steps (components) each have a well-defined interface and are loosely tied to other steps. As a result, we can modify one step without extensive knowledge of other steps, simplifying development and improving maintainability of our system.

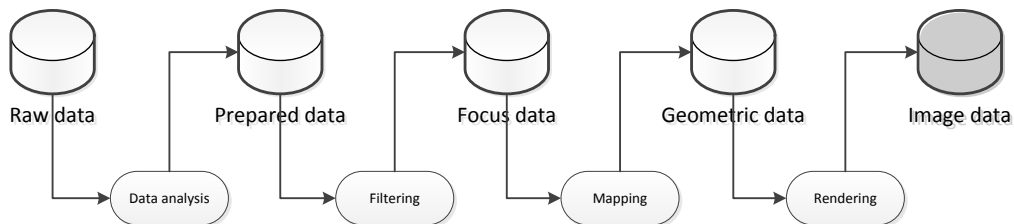


Figure 4.1: The visualisation pipeline, illustrating the steps involved in creating visualisations from raw data

4.1 Modelling

We start off the analysis of our data by making a fundamental observation regarding the base type of our data. In the paper *Task by Data Type Taxonomy* [2], Shneiderman et al. propose a taxonomy in which seven basic data types are

distinguished. One of these types is temporal data. This may not come as a surprise, as we encounter time-based data series in many application domains. Some examples of temporal data include weekly NASDAQ stock prices, the daily maximum temperature in The Netherlands since 1930 or the change over time of the penguin population on Antarctica. In general, time series are observations on a variable ordered in time. These observations may come from a number of sources, such as a simulation or from sensor-measurements. The latter is the source for the GreenMind dataset, which we will discuss further in section 4.2.

Let us first look at a basic definition for a time series (we will briefly discuss other representations in chapter 8.2). Consider that the data consists of a sequence of time-value pairs. A single observation pair $\langle t_i, v_i \rangle$ then represents the observed value v_i at a point in time t_i . We obtain an ordering on these pairs by ensuring that $t_i \leq t_{i+1}$ for all i . Note that this allows multiple observations for the same point in time. Using this simple model, we now have a definition for our time series T as well as an ordering on it:

$$T = \{\langle t_i, v_i \rangle : i \in \mathbb{N}\} \quad (4.1)$$

$$\forall i \in \mathbb{N}, t_i \leq t_{i+1} \quad (4.2)$$

Given an observation $\langle t_i, v_i \rangle$, what does v_i represent? Observations of temporal data can be based on one of two temporal primitives: *time points* or *time intervals* [9]. A time point represents a discrete point in time at which its value was observed. A time interval, on the other hand, represents a duration on an interval domain such as the preceding hour. The value associated with a time interval then represents that entire duration and could be, for example, the average speed of a moving object or the total number of requests made against a server.

In the case of the GreenMind project, data consists of the total consumption measured during a certain interval. Therefore it makes sense to let v_i represent the total amount of consumption during the interval ending at t_i . To be able to compute aggregated values such as average consumption, we also need to know when our interval started. The pair $\langle t_i, v_i \rangle$ does not store this value, so instead can we use the preceding interval t_{i-1} ? It turns out we can not for a number of reasons. We may not have any data available for the preceding interval, for example due to failing measurement hardware. Also, we may not even know which pair represents the preceding interval.

Instead of trying to derive our interval from the time series afterwards, we deal with this problem by choosing a fixed interval of 1 hour for our data. This also implies that the transformation function mapping measurements to raw data is responsible for protecting the integrity and consistency of our data. We will briefly discuss the component implementing this function in section 4.2.

4.1.1 Missing data

As a result of data records measured over time, missing observations in time series data are very common [3]. Possible causes include faulty equipment, lost records, or human error. Even though measurement equipment may provide options to (partially) restore missing records, we will have to consider how to deal with gaps in our data. Keep in mind, however, that our goal is to present a visualisation of our data and not an exact representation. Therefore, we may be able to get away with estimating missing values without affecting the interpretation of the visualisation. Several techniques for estimation are discussed by Fung et al. [3].

To separate concerns about data integrity from visualisation, we introduce a transformation function $f(t_{start}, t_{end})$ that, given two points in time, will return a subset of our time series data for the interval $I = (t_{start}, t_{end}]$ without gaps and with data pairs defined on an hourly interval. Note that this set is *excluding* t_{start} , which represents the measured consumption in the *preceding* hour (which is outside of the interval I). This function should also estimate a value for any missing points t_i within the interval.

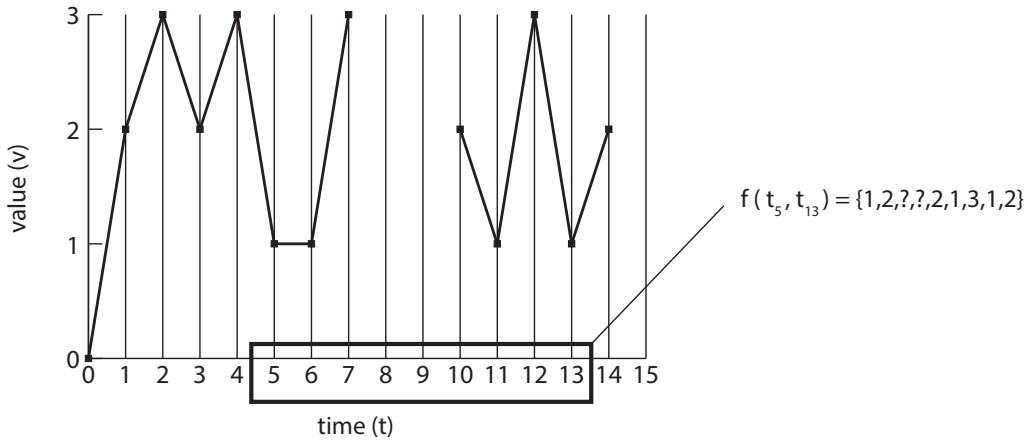


Figure 4.2: Chart showing a series of time/value measurements.

Note the missing values for t_8 and t_9 . The function f will estimate the missing values and return the interval $I = (t_5, t_{13}]$.

4.2 Data acquisition

The visualisation pipeline shown in figure 4.1 starts out from raw data. This data has been collected from hardware measurements, the details of which we will

discuss only briefly in this section. What is more interesting is how we can prepare the raw data so that it is suited for use in our application. This is especially relevant because we are dealing with *two distinct sources of raw data*, as mentioned in section 1.1. By transforming the raw data into a single prepared data model, we can reuse the components in the rest of the pipeline without having to account for differences between individual data sources.

4.2.1 Plugwise data

Our first set of raw data is obtained using Plugwise¹ hardware. Measurements are taken by placing sensor plugs in between power outlets and the device we want to measure. These sensors then form a wireless mesh network, allowing us to collect data from any plug within that network. Within the GreenMind project, a component named *Collector* is responsible for collecting and storing the raw data in a database. This component also ensures that measurements are grouped in intervals of one hour. For details on how the Collector component handles data acquisition, please refer to [\[paper Marcel/Sijmon\]](#). A sample of the obtained data is shown in listing 4.1.

Listing 4.1: A sample record of raw Plugwise data

```

1  {
2    "@type": "d", "@rid": "#10:0", "@version": 0, "@class": "HistoryLog",
3    "DeviceAddress": "000D6F000098C09E",
4    "LogAddress": "0D059948",
5    "Time": "2013-05-28T06:00:00.000+02:00",
6    "Pulses": 3531,
7    "Watt": 2.0916033,
8    "Kwh": 0.0020916034,
9    "CorrectedPulses": 3506.9475,
10   "CorrectedWatt": 2.0773556,
11   "CorrectedKwh": 0.0020773557,
12   "@fieldTypes": "Pulses=l,Watt=f,Kwh=f,CorrectedPulses=f,CorrectedWatt=f,CorrectedKwh=f"
13 }
```

4.2.2 Blackbox data

The second set of raw data is obtained from a dedicated hardware device called Blackbox. This device has been developed by an external party and is used to measure pulses emitted by traditional metering equipment such as electricity meters, gas meters, water flow meters etc. By placing a Blackbox at a building's utility room, the total consumption for the entire building can be measured. To translate pulses to actual energy consumption, a conversion is done based on the

¹<http://www.plugwise.com/idplugtype-u/about-plugwise>

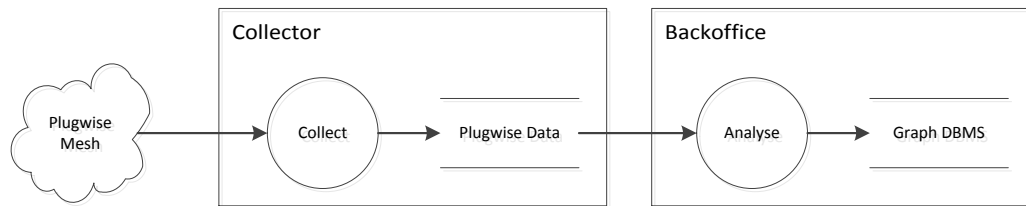


Figure 4.3: Data flow for importing raw data from Collector

pulse frequency of the measured device. For example, a typical energy meter emits a pulse every time 10Wh of energy passes through. Counting the number of pulses emitted during an interval of one hour and multiplying by 10 then gives us the total consumption in Wh. The pulse counts collected by the Blackbox are sent to a webservice (called SMOG) that stores them and provides a webinterface for configuring conversion rates on a per-device basis. A sample of the obtained data is listed in listing 4.2. Note that these measurements are not yet mapped to an interval of 1 hour.

Listing 4.2: A sample record of pulse data collected by the Blackbox

```

1 {
2   "id": "21588057",
3   "meter_identifier": "0004251c50ba",
4   "port_identifier": "1",
5   "read_at": "2010-07-10 03:04:04",
6   "pulse_no": 244837,
7   "created_at": "2010-07-10 03:04:05",
8   "updated_at": "2010-07-10 03:04:05",
9   "timestamp_string": "1278731044249",
10  "timestamp": 1278731044249,
11  "version": 213
12 }
```

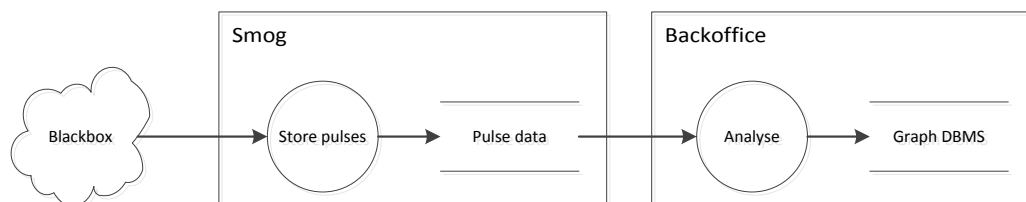


Figure 4.4: Data flow for importing raw data from SMOG

An inspection of listings 4.1 and 4.2 already shows some similarities between

the two datasources. As stated, our objective is to store these into our unified data model. The details of this model will be discussed in the next section.

4.3 Storage of data

The raw data we have obtained from our data sources could be stored in a single relational database table. We would then be able to simply query this table for specific time intervals (assuming relevant indices have been created). However, in the case of the GreenMind project, we also want to connect our data to related entities such as the device measured, its owner and its location within the building. In a traditional relational DBMS, performing queries that include a related model typically involves join operations. Performance of such queries may deteriorate as the size of our dataset grows. Instead we have chosen to use a graph database, which is typically optimized for connectivity. This provides us with a number of benefits [5]:

- Queries in a graph database are localized to a portion of the graph only. As a result, the execution time of a query is proportional to the size of the subgraph traversed to satisfy that query (and not to the size of the entire dataset). As our dataset grows over time, the performance of querying the data remains constant.
- Graph databases provide a large amount of flexibility regarding their schema. This allows us to expand upon the initial structure of the data model without disturbing existing data. As they depend less on a fixed schema, they are better suited to deal with changing data and schema evolution.
- Relationships in a graph naturally form paths. Querying and traversing the graph involves following paths. Because of the path-oriented nature of our data model, the operations provided by the DBMS are well-aligned with the way in which the data is laid out, making them extremely efficient.

For a more elaborate discussion of graph databases, consider reading O'Reilly's introductory book *Graph Databases* [4]. We will continue by presenting the database ontology for our dataset.

4.3.1 Modelling the environment

The ontology for storing our dataset was designed with two goals in mind: allowing the use of graph traversal for effective querying, and modelling of the building environment in which measurements are taken. The latter will allow us to deal with (regular) changes in this environment. For example, devices may be moved within a building and as a result belong to a new owner. By indicating ownership using an edge, we can simply update the device vertex to reflect the

new situation.

In the test environment in which the GreenMind project is deployed, the physical layout of the building (its topology) is reasonably easy to understand. The building itself consists of a number of floors. Each of these floors contains a number of rooms which in turn have one or more working areas. Each of these working areas can be occupied by a person and may, for example, contain a desk with a PC on it. Figure 4.5 shows a map of the 5th floor of our test environment, while figure 4.6 shows a graph that represents a part of this environment. Note that this graph can be represented as a connected acyclic (simple) graph. Therefore, we can also look at this graph as a tree - we will sometimes use this analogy and use the terms *textitparent* and *textitchildren*.

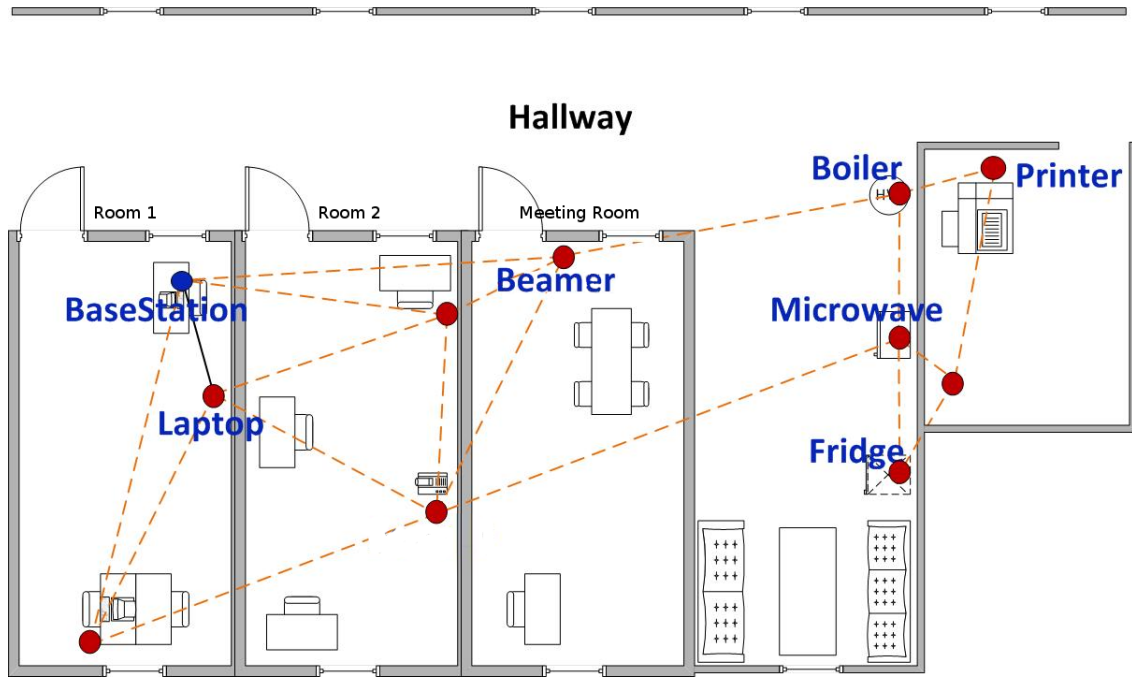


Figure 4.5: Map of the 5th floor in our test environment

At this point, we have not yet discussed the relation between devices and occupants. How do we deal with areas that are shared by multiple people? Who is the owner of the microwave in the kitchen or the beamer in the meeting room? In our model, we solve this issue by allowing locations to be *shared*. For such locations, we determine the occupants by looking at its parents. A device is then related to any person who has access to the area it is installed in. We have listed some sample data for this relation in table 4.1, based on the graph structure in figure 4.6. We define occupancy and ownership as follows:

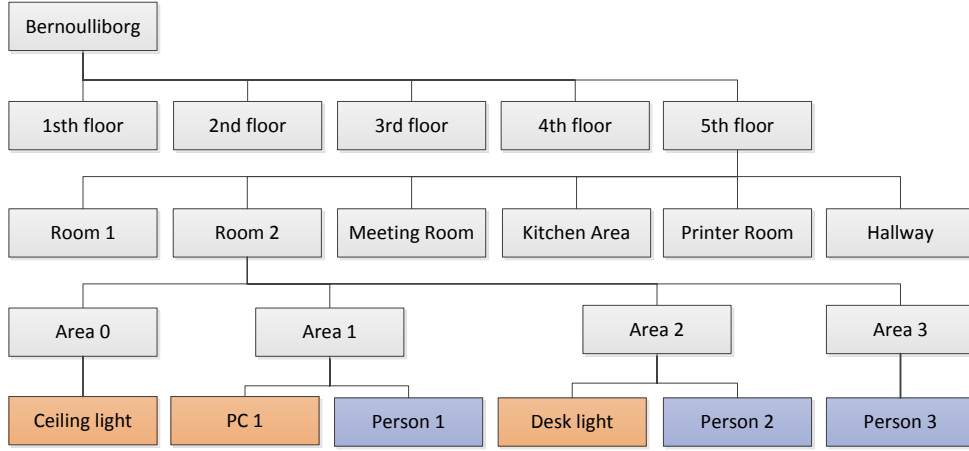


Figure 4.6: Partial graph containing locations, devices and occupants in our test environment

$$Occupants(loc) = \begin{cases} Occupants(loc.parent) & loc.shared \\ \{c.occupiedBy : c \in loc.children\} & otherwise \end{cases}$$

$$Owners(dev) = Occupants(dev.installedIn)$$

Using these definitions, we can infer ownership for the ceiling light in Area 0. The area itself has no occupants, but it is a shared area and thus we look at its parent: Room 2. Room 2 is a regular area, so its occupants are the collection of its children's occupants: Area 0, 1, 2 and 3. Hence:

$$\begin{aligned} Owners(Beamer) &= Occupants(Beamer.installedIn) \\ &= Occupants(Area\ 0) \\ &= Occupants(Room\ 2) \\ &= \{c.occupiedBy : c \in Room\ 2.contains\} \\ &= \{P1, P2, P3\} \end{aligned}$$

Thus, we can use the graph representing our environment to reason about the ownership relation. As a result, the amount of manual modifications to the graph required when the physical environment changes is reduced (which is an important factor in keeping the system maintainable and actual).

Location	Shared?	Occupants	Devices
Room 2		P1, P2, P3*	
Area 0	Yes	P1, P2, P3**	Beamer
Area 1		P1	PC1
Area 2		P2	PC2, Desk light
Area 3		P3	PC3

Table 4.1: Using the graph to relate devices and occupants

The occupants for a location can be inferred from edges within the graph. For example, *Room 2* has three subareas with occupants: P1, P2 and P3 (*). We determine occupancy for the shared *Area 0* by looking at its parent, *Room 2* (**).

4.3.2 Modelling measurements

The topology we discussed in the previous section could have easily been modelled using a relational DBMS, especially when the number of locations and devices is small. Our choice for a graph DBMS does provide some flexibility with respect to the data schema, but it really starts to pay off when we look at modelling the storage of measurement data. The continuous growth in temporal data makes it crucial that we can efficiently process queries.

Recall that each measurement represents an interval of one hour. Each such measurement was taken for a specific device and, at that time, the device may have belonged to one or more persons. While device ownership might change over time, the owner at the moment that a measurement was taken is definitive for that measurement (it's now a part of history)! Therefore, in addition to storing measurements, we store the relation between measurement, device and person as an additional edge in our graph. A partial graph is shown in figure 4.7. Using this graph, we can extract all measurements for a user or a device by traversing the edges between the user or device vertex and connected measurement vertices.

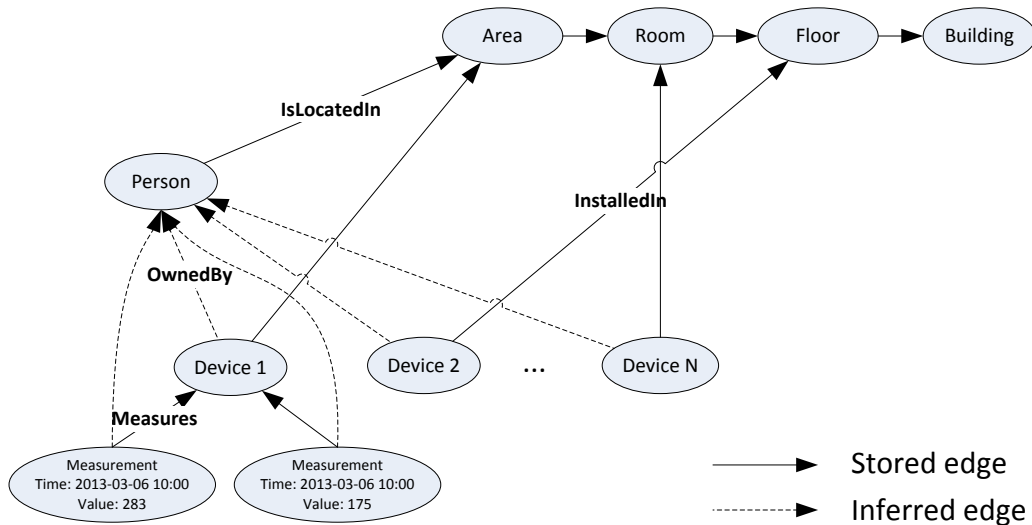


Figure 4.7: Example measurements and their relation to our topology

The solid LocatedIn and InstalledIn edges are managed using a back office, while the dashed relations are inferred from them during runtime

What to do, however, if we want to extract measurements for a specific time interval? For example, how can we obtain the total consumption this month? We do not wish to query all measurements and filter them by time, since this would still require us to process all measurement records. Instead, we introduce one

more concept to our ontology: time. By perceiving a point in time as consisting of separate components (such as year, month, hour), we can build a tree out of time components. Each node in this tree then represents a specific interval. For example, a vertex `Month` with value 7 and an edge to the vertex `Year` with value 2013 would represent the interval Juli 2013.

You might have remembered that our measurements represent time intervals of one hour. Thus, by building a tree including vertices of the type `Hour` we can connect our measurements to this tree. Figure 4.8 contains a sample of such a tree.

So what have we gained by adding time vertices? Consider our query for the total consumption this month. We could now simply find the vertex representing this month and then traverse down the tree to collect all measurement logs that belong to this month. Note that we are no longer dependent on the total number of measurements in the database, but only on the effort to find the month vertex (which is negligible) and the number of measurements for the month (which is exactly our result set)!

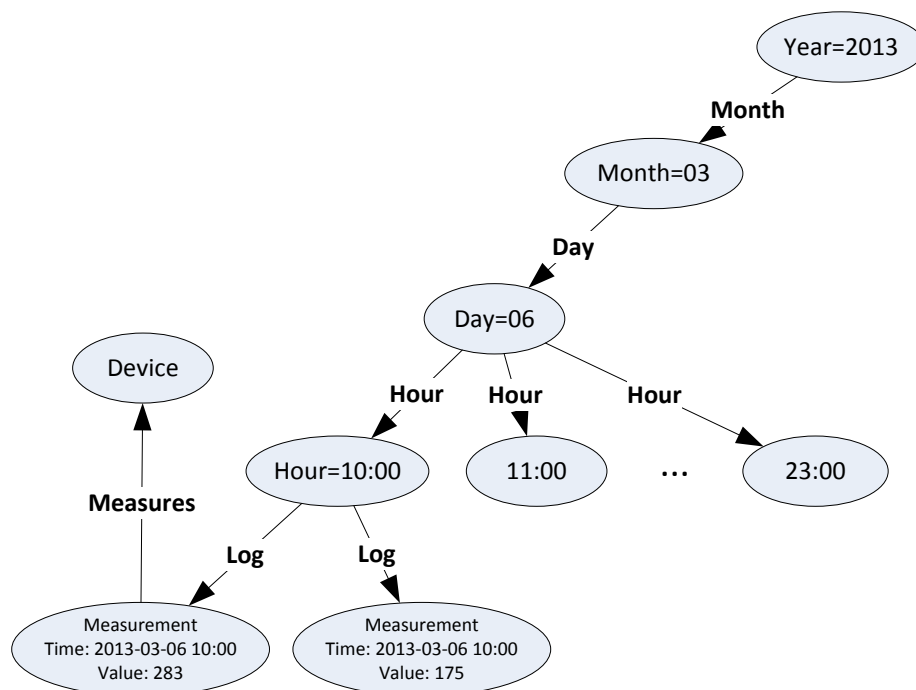


Figure 4.8: Example measurements and their relation to time

Note that, as for the building topology, we can also view our time tree as a simple undirected graph. Thus, we can easily store it in our graph DBMS using vertices and edges. The final ontology that contains both the building topology as well as the time tree is shown in figure 4.9. We will discuss querying and processing data in this ontology in the chapter 5.

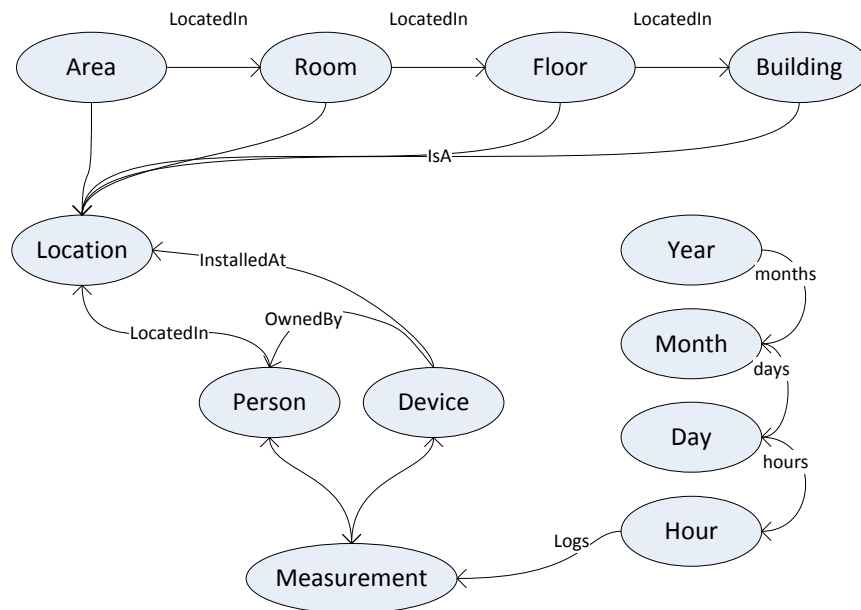


Figure 4.9: Complete database ontology for the GreenMind application

Note that the actual graph DBMS allows traversing both incoming and outgoing edges, effectively making relations bidirectional

The ontology presented at the end of section 4.3 forms the basis for the storage of prepared data (see figure 4.1). In this chapter we will discuss the transformation of raw data to fit this ontology, as well as querying the transformed data. This discussion relies for a large part on operations for graph traversal.

5.1 Inserting data into the graph

The main task to be done when inserting raw data into the graph database is the creation of edges between measurement vertices and vertices representing time, persons and devices. Storing the raw data into the measurement vertex is then straightforward. We will shortly discuss an algorithm that performs the creation of these edges, but first we will discuss two operations provided by the graph DBMS: `traverse` and `flatten`. Note that different graph databases may provide different means of querying data. Many support the Resource Description Framework (RDF) query language SPARQL Protocol and RDF Query Language (SPARQL) and the imperative, path-based query language Gremlin. Despite these different query languages, the basic concepts are similar.

Traverse

The `traverse` operation allows us to cross the relations between vertices (represented by edges). It also allows us to specify the conditions on which these edges are traversed, such as the edge class (i.e. relationship type) and direction (incoming vs outgoing). Traversal returns a collection of vertices that can be reached from the starting vertex while obeying the specified conditions.

Flatten

The `traverse` operation returns a collection of vertices, which may consist of different classes. In case where we are only interested in a specific class we can use `flatten` to project the collection onto this class. For example, traversal allows us to reach logs for a user's devices through the path `Person ← [OwnedBy] ← Device ← [For] ← Measurement`, but will also return the `Person` and `Device` vertices. Using `flatten` we can select only the measurement vertices.

Using the traverse operation in conjunction with flatten allows us to find the `Person` vertex to which a measurement should be linked. Each measurement has a MAC address associated with it which uniquely identifies the measured device. By then traversing the edge for the `OwnedBy` relation we can find the `Person` vertex and link it to the vertex for the measurement.

Linking an `Hour`-vertex requires some extra effort. The time dimension is infinite, and the vertex representing an hour in the future may not have been created yet. Therefore, when a measurement vertex is created, we break its timestamp into the appropriate year, month, day and hour pieces and check for each piece if a vertex is present. If not, we create these vertices. After processing all the pieces, the time-tree is again up-to-date and we link the measurement to it.

The complete process of transforming raw data to a measurement vertex is shown in listing 5.1. The function `findHour` it uses is shown in listing 5.2.

Algorithm 5.1 Insert measurement into graph structure

Require: Raw data log `l`

Ensure: Updated graph containing representation of `l`

```

 $h \leftarrow \text{findHour}(l.\text{timestamp});$ 
 $d \leftarrow \text{findDevice}(l.\text{mac\_address});$ 
 $p \leftarrow d.\text{OwnedBy};$ 
 $\text{measurement} \leftarrow \text{createNode}(l);$ 
 $\text{createEdge}(l, h);$ 
 $\text{createEdge}(l, p);$ 

```

Algorithm 5.2 Find/create year, month, day and hour vertices for a timestamp `t`

Require: Datetime `t`

Ensure: Updated graph contains node for `t`

```

 $l \leftarrow [t.\text{year}, t.\text{month}, t.\text{day}, t.\text{hour}];$ 
parent;
for all segment  $\in l$  do
    node  $\leftarrow \text{findNode}(\text{segment});$ 
    if node is empty then
        node  $\leftarrow \text{createNode}(\text{segment});$ 
        createEdge(node, parent);
        parent  $\leftarrow \text{node};$ 
    end if
end for
return node

```

5.2 Extracting data from the graph

Assuming that all the raw measurements have successfully been inserted into the graph, we can now use this graph as a basis for the next step: filtering the data. We argued in section 4.3 that we can query our data efficiently using graph traversal, regardless of the amount of data stored. Let us consider a number of relevant queries and how to execute them using `traverse`.

Listing 5.3: Finding occupants for a location

```

1  /*
2   * This will traverse any incoming edges which, for locations, are either
3   * a child room, a person or a device. Hence, this will find all persons
4   * occupying a location.
5   *
6   * ? = id of the location vertex
7   */
8  select from (traverse V.in, E.out from ?) where @class = 'Person'
```

Listing 5.4: Finding devices for a person

```

1  /*
2   * Finds devices for a person. The person vertex is identified using a subquery
3   * which uses the account for a person.
4   * To prevent traversing measurement vertices, traversal is constraint by the OwnedBy
5   * class and the targeted PlugwiseDevice class.
6   *
7   * ? = Account name for the user
8   */
9  select from (
10     traverse V.in, E.out from (
11         select from Person where account = '?'
12     ) while (@class = 'Person' or @class = 'OwnedBy' or @class = 'PlugwiseDevice')
13 ) where @class = 'PlugwiseDevice'
```

The examples in listings 5.3 and 5.4 could also be implemented efficiently using join-operations, especially when the number of locations and devices is small. However, querying for the actual measurement data is where our choice for a graph DBMS starts to make a difference in performance. The queries shown in listings 5.5 and 5.6 both use the time-index to retrieve the desired result set using graph traversal.

Listing 5.5: Finding measurements for a device since the start of this month

```

1  /*
2   * Finds all measurements for a device in the last month.
3   * Logs are found by traversing hour vertices found using a subquery.
4   * This uses flatten to project the found logs into a single listing.
```

```

5  *
6  * ? = Parts of the timestamp
7  * ?? = Device MAC address
8  */
9  select from (
10     select flatten( logs ) from (
11         select union( logs ) as logs from (
12             traverse hour from (
13                 select from (select flatten(day) from (select from Month where year = ? and
14                     month = ?)) where day >= 1
15             )
16         )
17 ) where device_address = ? order by timestamp asc

```

Listing 5.6: Getting logs for a device for a specific interval

```

1  /*
2  * Finds all measurements for a device in a specific interval.
3  *
4  * We may have to traverse multiple time vertices to obtain the resultset. These
5  * may in turn span multiple years, or months. This makes it important to carefully
6  * build the condition for selecting the vertices.
7  *
8  * ? = Parts of the timestamp, MAC Address for the device
9  * ?? = Device mac address
10 */
11 select from (
12     select flatten( logs ) from (
13         select union( logs ) as logs from (
14             traverse hour from (
15                 select from (select flatten(day) from (select from Month where ?)) where ?
16             )
17         )
18     )
19 ) where device_address = ? order by timestamp asc

```

In the previous chapters, we have shown how we can store and process the raw data for our visualisation application using a graph DBMS. We have shown the steps involved and related them to the visualisation pipeline. In the following chapters, we will use this knowledge to provide a design for the GreenMind application. We will also relate the visualisations it provides to the criteria for providing effective feedback which we discussed in chapter 3.

Part II

Application Design

In the following chapters we will relate the foundations from part I to the design of the GreenMind application. It is not our intent to present a full architectural design. Instead, we will elaborate on the goals of the application and link them to the choices we make for visualising our data. By discussing the functionality supporting these goals, we also provide a basis for the user study in part III.

6.1 Goals and functionality

The hallway display and mobile client we discuss in the following chapters are two components of a larger system. The overall system not only provides us with consumption data, but also provides context to the environment (is a PC being used?) and control over devices (such as putting a PC to sleep when it is not being used). The components in this system work together towards the overall goal of the GreenMind project: making the Bernoulliborg building a (more) sustainable environment.

What distinguishes the hallway display and mobile client from the other components is that they are the only two components that explicitly interface with building occupants. Therefore, they perform an essential role in involving these occupants with the project. As briefly discussed in chapter 1, one of the major goals of the GreenMind project revolves around educating occupants of the Bernoulliborg. The idea behind this goal is that, by increasing awareness, building occupants will become more conscious about their energy consumption. Our application supports this goal by:

1. Informing users of energy consumption, to increase awareness
2. Informing users of the overall status of reduction efforts
3. Stimulating users to reduce their personal energy consumption

The hallway display supports these goals by providing generic visualisations of consumption data. The mobile client allows us to go a bit further, by also showing personal data to the device user. The user might use this extra information to discover new ways to reduce his or her consumption, thus increasing the user's sense of control. The basic functionality provided by the client includes:

- Viewing personal consumption data

- Viewing personal devices and their consumption
- Viewing trends: change in consumption over time
- Comparing personal consumption against average consumption

Note that the client provides some other functionality as well, and we will look at possible other features that may further contribute to our goal in chapter 13.

6.2 Design goals

The major goal of the GreenMind project is to promote a sustainable environment. Our application contributes to this goal by stimulating users to reduce their consumption. Note, however, that the application by itself will only provide information to users. From this information, they may or may not extract knowledge on how to save energy. Thus, keeping the user interested is paramount to achieving our goal. Therefore, the major design goals for our application are aimed at keeping the user interested and allowing for easy extension of the application with new functionality.

A 2011 research on mobile app analytics¹ shows that over 50% of mobile users abandon an application after using it less than five times. While scientific research into this area is sparse, information from commercial parties suggests a number of major reasons for abandoning an application:²

- The application crashes more than once
- The application is not intuitive to use
- The application requires permission to collect personal data
- The application is not optimized for the device
- The application uses a large amount of memory and/or battery

This list further reinforces our observation that focusing on the user is crucial while designing the mobile application. Since data visualisation makes up a large part of the application, we will revisit this subject when discussing the actual implementation of the rendering step of the visualisation pipeline. In summary, the following design goals were most important:

Usability & Performance

Focuses on building an app that is intuitive to use and provides a pleasant

¹<http://www.localytics.com/blog/2011/26percent-of-mobile-app-users-are-either-fickle-or-loyal/>

²<http://venturebeat.com/2013/06/03/five-reasons-users-uninstall-mobile-apps/>

user experience (UX). This includes a focus on making efficient use of resources: an app reporting on energy consumption can not afford to drain the device it runs on from resources, such as energy and networking bandwidth.

Modularity

One way to involve users is by providing new functionality to an app regularly. By designing for modularity, we also allow for easier extension of the app. Modularity may also allow us to share code between the hallway display and mobile client.

Integrity & Security

In order for the user to be able to make assumptions based on the visualised data, that visualisation may not be misleading or incorrect. Incorrect data may result in wrong conclusions and prevent users from changing their behavior. Besides that, users may feel uncomfortable if others gain access to their personal data. Thus, both security (data tampering) and integrity (internal processing) are points of focus.

We will continue by discussing the translation of these (and other) design goals to a design for both the mobile app and hallway display.

The mobile application (app) is one of two interfaces available that interfaces with building occupants. The other is the consumption display. While the consumption display primarily focuses on communicating general (aggregated) consumption information, the mobile client focuses on providing insight from a more personal perspective. This makes it crucial that the client is designed for an optimal user experience. If users are unable to use the app for making assumptions on how they might reduce their energy consumption, they may lack a sense of control. Consequently, it is highly unlikely that the user will alter his behavior

In this chapter, we will discuss the implementation of mobile data visualisation while keeping in mind the design goals for the application. Looking back at the visualisation pipeline introduced in chapter 4, our discussion will mostly focus on mapping and rendering the data. First, however, we will consider the specific challenges that result from the mobile context.

7.1 The mobile context

The smartphone has become an integral part in many people's lives since the introduction of Apple's iPhone in 2007 and Google's Android platform in 2008. Together with the increased availability of internet through wifi and mobile networks such as 3G, this resulted in a device that allows us to perform many tasks which previously would have required a desktop computer. There are a number of characteristic differences between mobile devices and desktop computers [13], some of which affect the use of a mobile device for visualisation:

- Display size is limited, mostly in terms of physical dimensions.
- Connectivity is slower, affecting the responsiveness of applications while performing networking.
- There is a lack of native libraries suited for visualisation.
- Hardware and software developments move very fast, resulting in a wide ecosystem of devices and (versions of) operating systems to consider.

Besides these differences, there is another major difference: the way in which we use mobile devices. When working with a desktop PC, we typically focus a

large portion of our attention on the task at hand, and for a longer period of time. For mobile devices, attention is typically shared with concurrent activities (Sohn et al. [21]). In their research, Sohn et al. acknowledge that mobile tasks are typically driven by information needs arising from the user's current context. They also researched whether users were able to address their needs and, if not, the reason why they were not able to do so. Of the 30% of needs that were unaddressed, *not important* (35%), *no internet access* (23%) and *did not know how to address* (23%) were most commonly given as a reason. Other research shows that slow response of an application also plays a major role in failing to complete tasks [22].

As a result of the concurrent nature in which we use our mobile device, Chittaro et al. distinguish some further complications [13]. A user's cognitive resources are limited, potentially making the use of an app a secondary task. Distractions caused by the device can limit the user in its capabilities of performing their primary task, and vice versa. This provides us with a good reason to present the visualised data at-a-glance, so that it may be easily understood and that the amount of cognitive resources required is limited.

The mentioned characteristics of a mobile device, both in terms of the device itself as well as the way in which it is typically used, prompts one to carefully consider the design of a mobile app. We have already mentioned a number of general design goals for the GreenMind app in section 6.2. We will expand on these goals in the following sections.

7.2 User experience

We have mentioned the importance of UX several times. It is now time to relate this design goal to an actual solution that provides a good user experience. We do this by using a number of heuristics, prioritised by the considerations we made in sections 6.2 and 7.1. These heuristics should serve as guidelines when making implementation choices, and we will later illustrate them using the GreenMind app's design.

7.2.1 Heuristics

One of the most commonly known collection of heuristics for the design of user interfaces is that of Nielsen et al. [23], which dates back to 1994 but is still widely used. Marta Rauch used Nielsen's research to present similar heuristics for mobile devices. Developer guidelines for the Android platform provide us with even more heuristics. What follows is a selection of these heuristics based on our goal: providing a good user experience for mobile visualisation.

Consistency and standards

Users should not have to wonder whether different words, situations, or

actions mean the same thing. Follow platform conventions. If it looks the same, it should act the same. By following standards, we also ensure that our application is prepared for future updates to the platform.

Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time. Especially in the case where we are loading data for a visualisation, the user should be aware that there is a background process active. This also implies that we should design an application that responds quickly, instead of having the user guess whether a delay in response is intended or if something went wrong.

Carefully design navigation

Clear and consistent navigation is an important part of user experience. Mobile users have a mental model of how navigation works on their device, and navigation that behaves in unexpected ways is a source of frustration. By considering the relations between content in an application, one can choose an appropriate form of navigation that allows users to effectively and intuitively discover this content.

Recognition rather than recall

Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. With respect to visualisation, charts should be easily understood without requiring a large learning curve. This includes the considerations regarding visualisation made in chapter 3.

Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information competes with the relevant units of information and diminishes their relative visibility.

Error Prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

7.3 Resources and data transfer

When talking about data visualisation, most research typically focuses on the visualisation itself and not on obtaining the data. As a result of heuristics, the mobile context forces us to think about data acquisition as well. If we don't, the process of retrieving data might cause the app to become irresponsive or drain networking bandwidth and the device's battery. Especially the latter would not send a very positive message towards the user, considering that we are building

an app that aims to stimulate the *reduction* of resource usage. In chapter 5 we briefly mentioned a back office component responsible for importing processed data. We will use this same component to provide the mobile client with the filtered data it requires. For this, the back office provides an API that handles authentication and data requests. A full description of its architecture is out of scope of this thesis, but can be found on the GreenMind project wiki¹ or from the author by request.

If you have some experience with implementing applications that use networking, it is likely that you are familiar with the layered networking architecture or Open Systems Interconnection (OSI) model [24]. This model was designed to hide implementation details in several separate layers and allows us to perform networking at the desired level of abstraction. This makes it easier to develop complex applications, but also hides potential sources of problems.

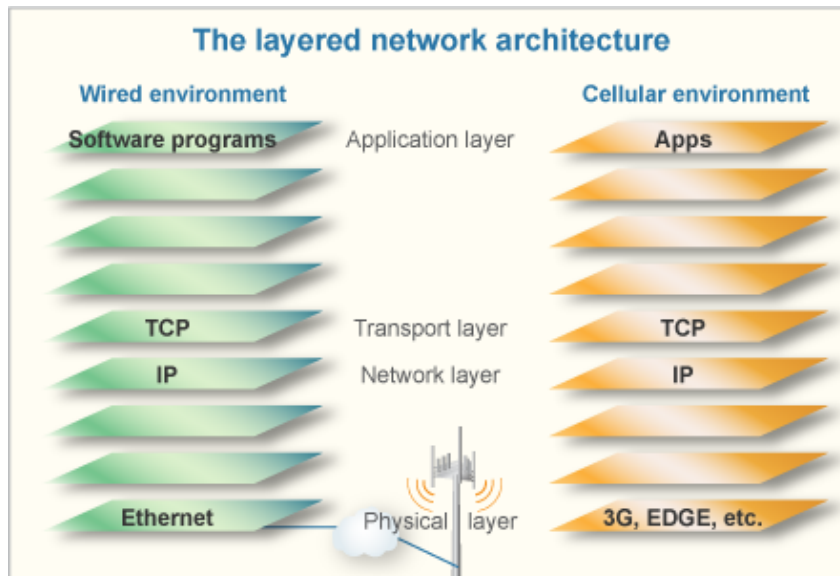


Figure 7.1: OSI Layered network architecture

Working with the OSI model causes a problem on Mobile devices, though. This is the result of a fundamental difference between the traditional desktop environment and the mobile platform. While we can typically assume the availability of a fast internet connection and near-unlimited resources in a wired environment, we have to account for a limited amount of resources in a mobile context. Networking on mobile devices requires a significant amount of battery power while connected. Opening a connection over 3G - a standard for mobile telecommunication - introduces latency of approximately two seconds. To deal with the trade-off between maintaining an energy-consuming connection and introducing high latency, a typical device's radio is designed as a state machine

¹<http://www.sm4all-project.eu/greenmind.trac/wiki/BackOfficeArchitecture>

(see figure 7.2). The presence of this state machine is the fundamental difference we have to be aware of even when developing on a much higher level of abstraction. The radio reduces its energy consumption by transitioning to a state that requires less energy after a period of inactivity, effectively shutting itself down. This tail time is set large enough so that multiple data transfers can be executed in the same time window, without the radio changing state. This also reduces the latency for subsequent requests. Once the radio moves to its idle state, the next request will require the radio to switch back to full power and will cause additional latency of up to two seconds before forwarding the request [25].

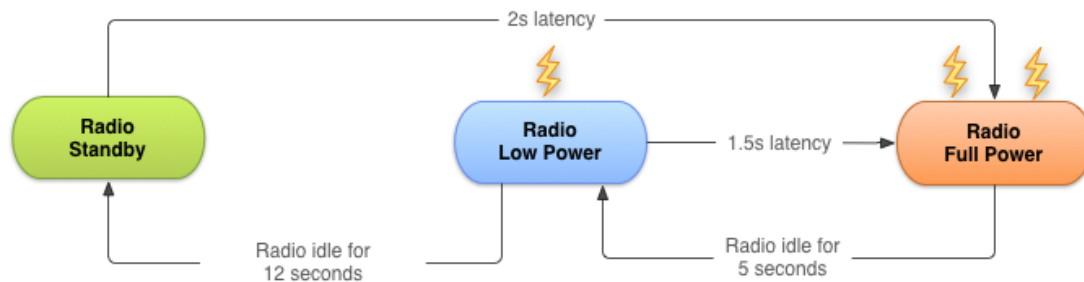


Figure 7.2: State machine for a 3G wireless radio, courtesy of the Android Open Source project.

So how does this state machine affect the design of mobile applications and why should we be aware of it? Consider a news application that checks a remote service regularly for updates - say, every minute. While the request for updates itself may be very small, the overhead of transitioning from the idle state to fully connected every minute adds latency. If not handled appropriately, this latency may severely reduce the responsiveness of the application. This in turn may negatively impact the user's experience. Due to the frequent scheduling requests, we also spend a significant amount of time outside the radio's idle state (see figure 7.3a). Depending on the amount of updates, a better option would be to let the server push those updates to the mobile client when they are available (figure 7.3b).

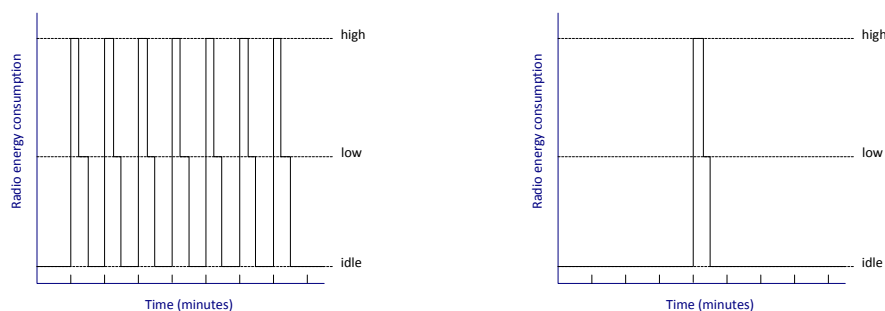


Figure 7.3: Example of subsequent radio states for pull (left) and push requests

In the context of our application, there are a number of strategies we can use to mitigate latency and battery drain. We will discuss two of them in depth: reducing transfer of redundant data and reducing the number of individual transfers made.

7.3.1 Reducing radio load by reducing the amount of requests

Left: radio behavior for frequent scheduled requests. Right: using push instead of pull-requests, moving the load to the server.

It is not a far stretch to state that we can reduce the amount of data transferred by preventing data being transferred multiple times. However, since we are visualising our data on a mobile device, we can not simply download the data once and from that point on use local storage. We may have to account for limited storage, and some visualisations are built on aggregate data (such as displaying an average) that needs to be recalculated regularly. So, we need a solution that allows us to determine where to obtain our data based on the characteristics of that data. This solution would then be responsible for:

- Providing an interface to retrieve data, effectively providing data transparency
- Determining whether we can use local data, remote data or a combination of both to fulfil a request
- Synchronising local data as we receive new remote data

Let us look at the situation from the perspective of the visualisation pipeline again. The (remote) graph database contains vertices representating our (decorated) time series. The local rendering component expects us to provide a geometric representation of this data. This representation is provided by a mapping operation. This operation would typically load from the remote database and process the data. To conserve the modular structure of the pipeline, it would be highly preferable if the mapping operation needs not concern itself with getting data from a number of possible sources. Luckily, our intended solution abstracts away from the data source, thus providing data transparency. The mapping operation can then access the data through this abstraction. We will discuss the full solution for this strategy in section 7.3.3.

7.3.2 Reducing the number of individual transfers

We have shown in figure 7.3 that it is beneficiary to reduce the number of transfers we make, especially if those transfers cause the radio to move out of the idle state. However, some applications will be very dependent on remote data and

we may not be able to reduce the amount of data we have to retrieve. In such cases, we can instead consider reducing the *amount* of transfers.

One strategy to do so is the use of batch transfers. In applications where we may not need the result of a data request immediately, we could simply save up those requests. We could then use a schedule in which we allow request to be executed as a group after a period of time, or by piggybacking them with another request that has to be dealt with immediately. One might compare this strategy with a traffic crossing on which cars will cross in batches to increase throughput, while giving precedence to high-priority traffic (such as an ambulance).

In our application, data requests are typically triggered by a user loading a specific view. As a result, we can not use the batch strategy without letting the user wait as well. Instead, we apply another strategy that also allows us to reduce the number of individual transfers: *prefetching*. Like with batch-transfers, the effectiveness of this strategy is highly dependent on the use of an application. Therefore, it requires making predictions about future requests the user is likely to trigger. Revisiting the case of a simple news application, the user might ask for a list of titles for recent sports news. Usage data might show that a large percentage of users will then look at several of the articles those titles link to. By prefetching the article contents together with their titles, we omit the need for retrieving individual articles. Do note that this is a trade-off with limiting the transfer of redundant data, as the user might also decide not to view article at all.

In summary, note that the effectiveness of both the strategies discussed before depends on the type and use of our application. In either case, thinking about connectivity is worth the effort when trying to optimise resource use at a low level.

7.3.3 Utilizing content providers

So far we have identified two major issues with the transfer of data. One is the potential negative impact of a nonresponsive app. The other is the inefficient use of resources. The solution we use to counter these issues is the use of an advanced data access layer based on the *ContentProvider* pattern.²

This pattern allows us to separate the retrieval of data and the application using Uniform Resource Identifiers (URIs). The *ContentProvider* provides a service that returns the data resource associated with a URI. Applications can use the Android query mechanism to ask for specific data and, if a service is available for the associated URI, the application receives a special pointer from the *ContentProvider*. This pointer *may* contain the requested data. It is this pointer, called a *Cursor* which is of special interest. By returning it to the calling

²<http://developer.android.com/guide/topics/providers/content-providers.html>

application immediately, the `ContentProvider` prevents blocking the application and its user interface. It can then start requesting remote data and fill the cursor asynchronously. Whenever data is added to this cursor, the application is notified that its content has been changed. The application can decide how to act on receiving new data.

We deal with the issue of inefficient data retrieval by extending the implementation of the `ContentProvider` with a local cache. Whenever a remote request is made, the `ContentProvider` will store the data received in a local database similar in structure to that of the prepared data. The provider may then be able to service future request using local instead of remote data. As a result, the number of redundant requests is reduced.

There is one minor drawback to this setup. The `ContentProvider` has to be able to determine when it can serve requests using local data, and when it needs additional remote data. This means that for each data URI we serve, we also have to think about checking local data for integrity and expiration.

The general structure of this solution is presented in figure 7.4. This includes the flow of requests and responses in the system after making a request for data.

7.4 Visualising charts

The GreenMind app was built as a native Android application. This provides significant benefits with respect to user experience. It allows us to exercise a large amount of control over the responsiveness of the application as well as the use of native design elements, increasing recognition by the user. There is one major drawback, however. Android lacks a native charting library that provides a large amount of control over the aesthetics and behavior of charts. This works against our attempt to provide an aesthetic and minimalist design, and would negatively impact the attempt to provide information at-a-glance. An alternative is to benefit from the flexibility and rendering power of some web-based charting libraries. Using a combination of HTML, JavaScript and CSS styling, we can have a large amount of control over how our charts will look and interact. In case of the GreenMind app, a data-driven approach was chosen that uses the D3.js library³. This approach allows a focus on manipulating data and binding it to a geometric representation of objects (using the Document Object Model (DOM)). The D3.js library provides functionality to render a visualisation from this geometric representation, thus allowing for a high degree of flexibility as well as a separation between mapping and rendering of data.

There are some minor drawbacks, however, to rendering our data using web scripting. As we have separated our rendering logic from the native Android

³<http://d3js.org/>

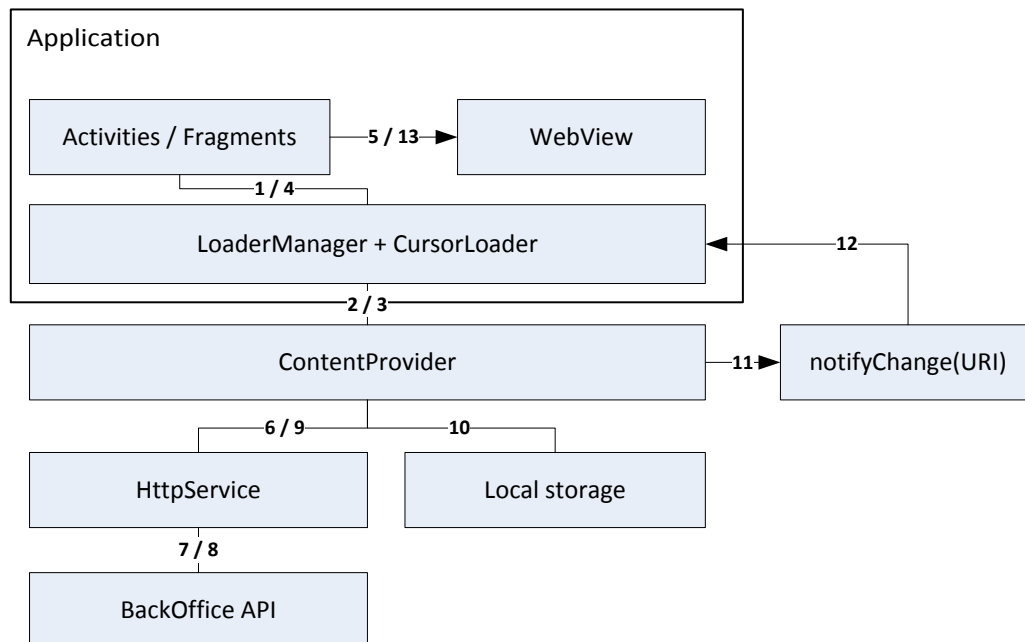


Figure 7.4: An activity makes a request using a URI (1), which is passed to a matching ContentProvider (2). Immediately after that, a cursor is returned which may hold data (3,4). The activity may be able to use this data to take further action (5). In the meanwhile, the ContentProvider retrieves new data from a remote service (6-9) and updates the local cache (10). After new data is received, the ContentProvider updates the cursor with the new data and notifies the activity (11, 12) which may take further action (13).

code, we will have to find a way to bind our data to a webview while still utilising the power of a content provider. For this, a special fragment is used that functions as a bridge between the native app and Javascript. By using a clear model for representing the rendered data we can keep this fragment relatively simple. The complete structure of our solution is shown in figure 7.7. To relate between the basic concepts and the sample case, this figure shows both the components making up the GreenMind system as well as the visualisation pipeline.

7.5 Effective navigation

We have discussed in section 7.2.1 that clear and consistent navigation is an essential component of the overall user experience. Consistent navigation can reduce the learning curve as well as the effort required to use an app. We can achieve this by aligning the navigation within an application with the mental model of its users. Thus, when designing the navigation within an app, it makes



Figure 7.5: Rendering a chart on Android using a Webview and frontend scripting (HTML, JS, CSS) allows us to control the rendering of each component of the visualisation, so that we may modify it as needed.

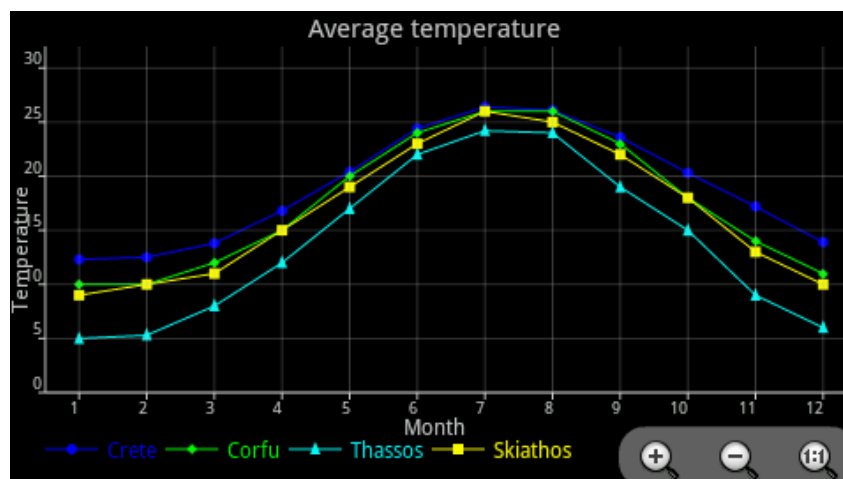


Figure 7.6: Chart rendered using native library

Currently available native libraries embed the rendering within the library, making it very hard to alter the styling and other details of the visualisation's components.

sense to look at navigational standards and other applications.

The Android platform currently provides a number of design patterns for navigation. It is strongly recommended by the Android development team to make use of these patterns to improve an app's recognizability. Like with all software patterns, each of these navigational patterns has its own positive and negative characteristics. A brief comparison of top-level patterns is shown in table 7.1.

The choice for using a navigation drawer as the top-level navigation for the GreenMind app was based on a number of considerations. First of all, the amount of different views provided by the app started out small (personal, group, built-

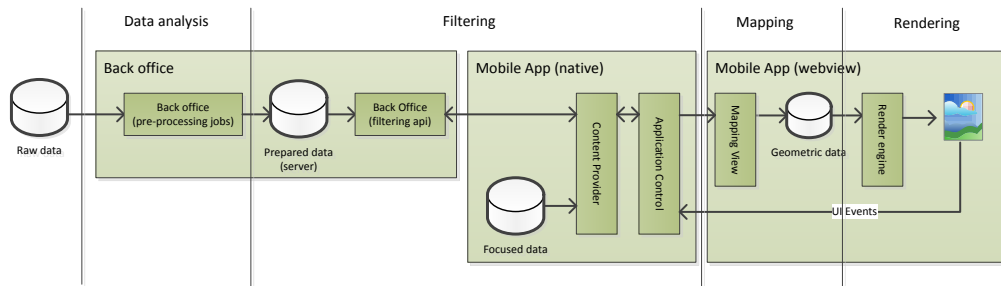


Figure 7.7: Relation between visualisation pipeline and mobile app

ding and device views) but was likely to increase over time. Fixed tabs would not accomodate for this growth, but both scrolling tabs and a navigation drawer might. Secondly, the amount of pixels on a mobile device is limited. To improve the quality of the visualisations we tried to reserve as much screen real estate as possible for drawing charts. Using tabbed navigation would require us to give up on vertical screen space. Finally, we anticipated the use of contextual or temporal navigation at some point (for example: selecting the interval for a chart). Displaying multiple levels of distinct navigation within a single view might be confusing.

An overview of the GreenMind app's navigational flow is shown in figure 7.8. Notice that platform specific navigation such as *up*, *back* and *home* have been tied into the flow to match their intended effect as close as possible.

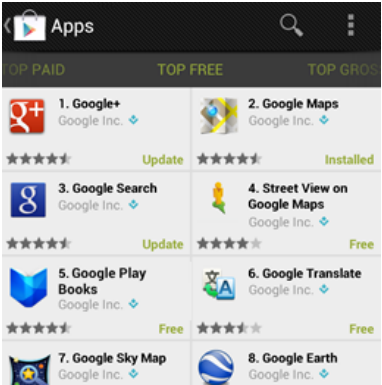
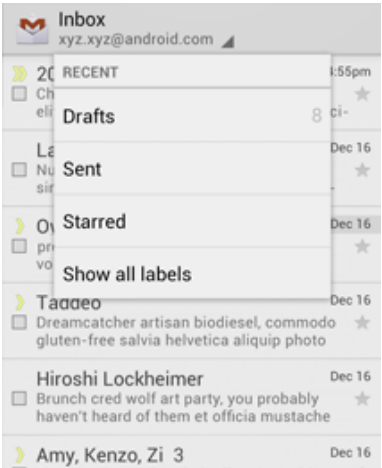
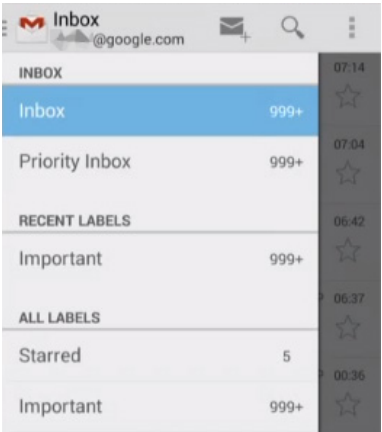
<p>Tabs</p> <p>Fixed tabs allow for easy exploration and switching of content. By being always visible, they increase awareness of their presence. However, they also require vertical screen space and the amount of tabs that can be shown is limited to a maximum of three. Additional tabs are supported by the scrolling tabs pattern, which trades scalability for visibility.</p>	 <p>The screenshot shows the 'Apps' page in the Google Play Store. At the top, there are three tabs: 'TOP PAID', 'TOP FREE', and 'TOP GROSS'. Below the tabs, there is a grid of app cards. Each card displays the app icon, name, developer, and a star rating. Some cards also show an 'Update' or 'Free' button. The apps listed include Google+, Google Maps, Google Search, Street View on Google Maps, Google Play Books, Google Translate, Google Sky Map, and Google Earth.</p>
<p>Spinner</p> <p>Spinners allow us to reduce the required amount of screen space by merging with the action bar. They are typically used for parametrization of similar views, such as the selection of an email account for which to view the inbox. It is also commonly used for switching between views of a dataset.</p>	 <p>The screenshot shows an email inbox interface. At the top, there is a header with the 'Inbox' title and the email address 'xyz.xyz@android.com'. Below the header, there is a list of email items. A spinner menu is open, showing a list of email accounts: 'RECENT', 'Drafts', 'Sent', 'Starred', and 'Show all labels'. The spinner menu is positioned over the email list, allowing the user to select a different account to view the inbox.</p>
<p>Navigation drawer</p> <p>The navigation drawer is a pattern that hides at first, making it less visible compared to tabs and spinners. However, it provides a large amount of space for navigating an app and allows multiple levels of navigation inside it.</p>	 <p>The screenshot shows an email inbox interface with a navigation drawer open on the left side. The drawer contains a list of navigation items: 'INBOX', 'Inbox', 'Priority Inbox', 'RECENT LABELS', 'Important', 'ALL LABELS', 'Starred', and 'Important'. Each item has a star icon and a count (e.g., '999+', '5'). The drawer is positioned on the left side of the screen, allowing the user to navigate between different views of the inbox.</p>

Table 7.1: Common patterns for top-level navigation in Android

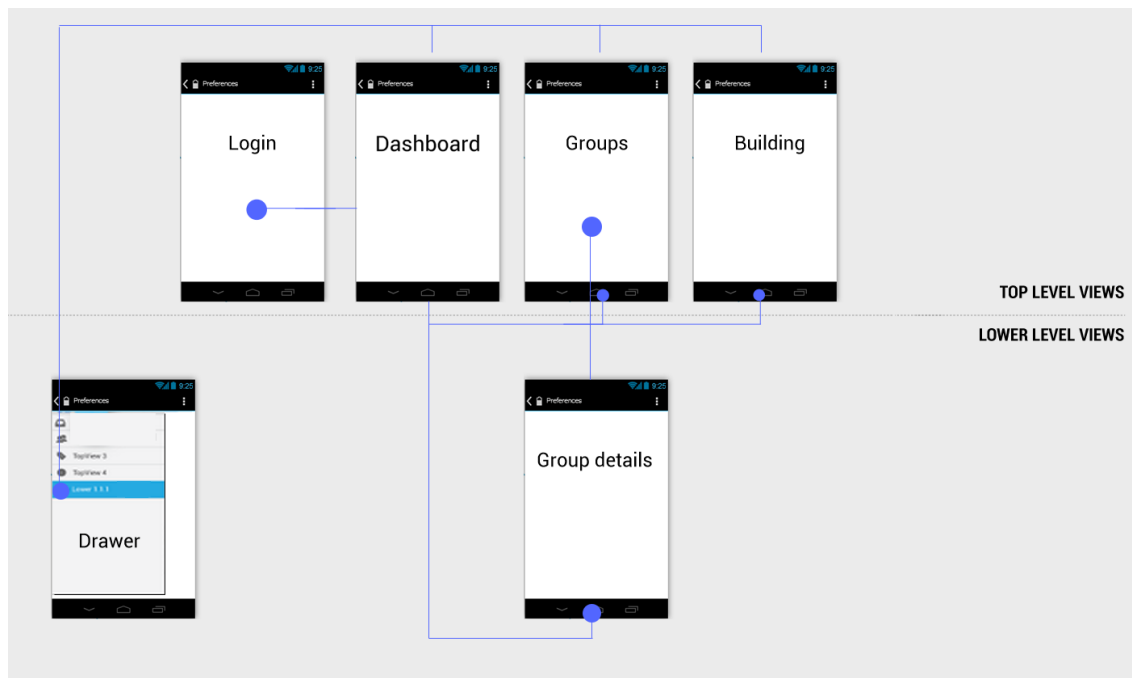


Figure 7.8: Partial overview of GreenMind navigation and flow

Chapter 8

Expanding visualisations

In the previous chapters we have focussed primarily on the design of the mobile application and relating it to the concepts discussed in part I. In this chapter we will shift our focus back towards visualisation. As a result of the modular setup of the GreenMind application and the implementation of steps in the visualisation pipeline, we are able to easily create additional visualisations. We will illustrate this by discussing some extensions in the following sections. Doing so will also show that the presented solution indeed addresses the issues discussed in the problem statement from section 1.2 and thus allows for a shift in focus towards creating visualisations instead of dealing with basic implementation issues. Next, we will discuss the hallway display component of the GreenMind app to illustrating code reuse. After that we will discuss a number of more advanced visualisations.

8.1 Hallway display

We have briefly discussed the hallway display component in chapter 6. This component provides a number of visualisations similar to that of the mobile client, with the difference that it aggregates data to remove the privacy concerns of individual users. By showing multiple visualisations, the display allows users to view the dataset from multiple angles. The display is mounted in a public location and has no input peripherals attached. Therefore, it aims to attract attention from passers-by through clear visualisations that are pleasant to look at. We've discussed a number of criteria for such visualisations in chapter 3. The implementation for this display turned out to be straightforward.

As a result of our preprocessing efforts to transform raw data into our prepared data store, we can reuse the prepared data and also use the same filters for accessing it. Thus, the only fundamental difference between the mobile client and the hallway display is the way in which we link the output of our filtering function to the geometric mapping function.

Please recall the complete pipeline for the mobile client we presented earlier in figure 7.7. We create the hallway display interface by replacing part of the mapping step with one suited to use with a regular browser. The implementation for this retrieves its data from the same API as the mobile client and can

reuse part of the mapping and rendering code as a result of our choice to use webscripting for displaying charts. What remains for the mapping function to be done is retrieving focused data and passing it to the mapping code responsible for geometric transformation. An overview of the new interface's structure is shown in figure 8.1. A number of visualisations from the hallway display are shown in section 8.2.

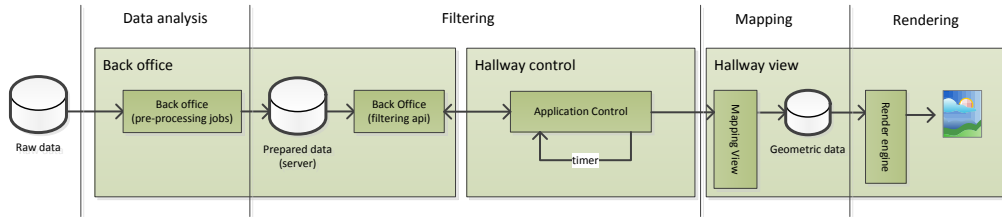


Figure 8.1: Relation between visualisation pipeline and hallway display

So, what does the mapping function for the hallway display look like? We can derive a number of basic tasks for the new function based on the display's functionality:

- Select a visualisation to render
- Collect the focused data needed for the visualisation
- Transform the focused data into geometrical primitives
- Trigger the rendering function with the geometric primitives

A rough implementation for this function has been listed in 8.1. Note that the complexity of this function is very low and that it is able to reuse the code for *transformData()* and *render()*. The only differences with the mobile app are the *loadData()* function, replacing *CursorLoader*, and the way in which a new render is triggered (scheduled instead of event-based).

Algorithm 8.1 Mapping function for the hallway display

Require: Viewspec[] vs

```

i ← 0;
while true do
  view ← vs[i];
  data ← loadData(view.query);
  chart ← transformData(data, view.transformation);
  render(chart);
  i ← (i + 1) mod vs.length;
  sleep(10000);
end while

```

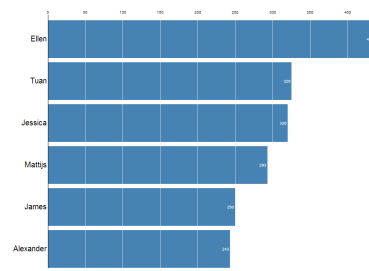
8.2 Advanced visualisation

In chapter 3 we have discussed the foundations for providing (eco-)visualisation. We discussed a number of criteria for providing effective feedback using a model for stimulating change in behavior. In this section we will use the GreenMind dataset to provide a number of potential visualisations that use the lessons learned from chapter 3. A number of these visualisations attempt to use social incentives to influence behavior, a technique discussed in section 3.2. While the GreenMind application does not (fully) use these techniques due to privacy concerns, such visualisations may have a powerful effect in situations where these concerns are less of an issue. Each of the shown visualisations has been constructed by altering the geometric mapping step of the visualisation pipeline. As a result, they can be implemented relatively easy in either the mobile application or the hallway display.

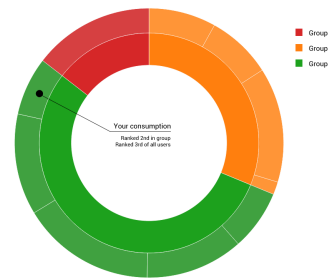
One of the type of motivations for behavioral change we mentioned resulted from social norms. By showing a person his/her energy consumption in relation to others, we hope to trigger a person to *conform to the norm* (given s/he is consuming above average amounts of energy). A simple visualisation using this social incentive is shown in 8.1a, and ranks a user compared to others as well as the *average user*. The effect could be made even stronger by showing the names of *all* users, though that again requires careful thought on what is acceptable in terms of privacy.

a) Horizontal bar ranking

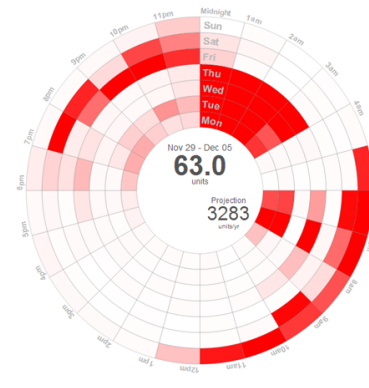
Ranking users by their energy consumption attempts to trigger a social motivation to be at or below average. This assumes that people do not want to be associated with the negative case of consuming above average). By showing an average line users are provided with a quantifiable target for reducing their consumption. Ideally, to provide a sense of control, users should be able to compare themselves with individual entries. This way, they may identify how to reduce their consumption.

**b) Circular ranking**

The circular ranking uses the same triggers as the horizontal ranking, but attempts to increase the social aspect by adding the concept of a neighbourhood. Users belong to the same group (e.g. a floor within a building, or a predefined research group) are shown together. This allows a user to compare himself with others that may be doing work of the same nature, which in turn may prompt the question why a user's consumption is different from that of his direct peers.

**c) Circular (spiral) chart**

One of the issues that often comes up with periodic data in a regular histogram or line chart is that it is hard to discover patterns visually. One attempt at making this easier uses a spiral to lay out the data. In this spiral, each revolution represents a fixed interval of time. By changing the interval (either interactively or using an educated guess), patterns may arise. On the right, the total energy consumption is shown for one week. In this case, the visualisation contains a distinct group of high consumption during weekday mornings.

**d) Calendar chart**

From a manager's point of view, it may be interesting to view general consumption over a longer period of time to start exploration. A calendar chart uses a preselected period to give a color-coded overview of a large number of periods. Doing so allows us to provide a quick glance of our dataset. In an interactive setting, such a chart should allow the user to zooming in on any of the periods.

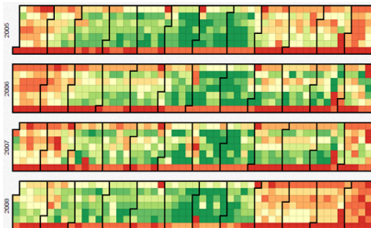


Table 8.1: Alternative visualisations based on the GreenMind dataset

Part III

Empirical evaluation

In the following chapters, we will present a small-scale empirical evaluation of the GreenMind app. Goal of this evaluation is to verify that the application presented in part II contributes to the goal of making the user more aware of his energy consumption. We have argued in chapters 6 and 7 that the application design should primarily focus on providing a pleasant user experience through an easy to use app and easy to understand visualisations. We will verify these design goals by devising a number of tasks that participants should be able to perform, the outcome of which should provide a participant with insight into his consumption and possibly with opportunities to reduce it.

Please be aware that this evaluation was not intended as an exhaustive comparison between our application and other possible approaches, nor an extensive analysis of user behavior. Instead, it should provide us with insight into the degree in which the application meets its goals and whether any (usability) issues can be identified. In this chapter, we will discuss the setup of the case study for this evaluation. After that, we will present the reader with the results as well as a discussion of those results.

9.1 Case study

There are several models available for performing an evaluation of a mobile application, ranging from testing in a controlled laboratory to letting test subjects use an app casually [26]. Highly-controlled testing typically involves the use of (expensive) software that is able to record and analyse a user's session of using the application. The availability of such software in the public domain is limited, so we have instead chosen for a (mostly qualitative) case study. Another important motivation for choosing this approach is to include the context while using the app, as it is likely to influence on the user's focus. The main research questions we attempt to answer with this case study are:

- How do users navigate the application? Are there any unsuccessful attempts at navigating the application and, if so, what types of alternative navigation are users expecting?
- How do users interpret information presented on single screens? Are they able to make assumptions based on the presented visualisations?

- How do users combine information from multiple screens? Are users able to make a connection between different visualisations?

As a result of the nature of these research questions, an exploratory setup has been chosen for the case study [27]. In this approach, the user is asked to answer a set of open questions. The answers to these questions can be obtained using the application. By observing how participants use the application to answer the questions, we attempt to make generalisations regarding our research questions.

For the case study, a total of five participants was recruited. While this may seem a small number, a minimum of five participants typically is enough to discover most usability issues with an application.¹ This does assume that we let those participants perform as many small tasks as feasible. The participants came from different demographic backgrounds. Their age ranged from 23 to 59 years while their experience with smartphones ranged from novice to experienced. Three participants were already familiar with the Android platform while the other two only had previous experience with iOS. Most participants had installed at least 5 to 10 applications on their smart phone manually, while 1 participant installed less than 5.

A special build of the application was used which retrieved its data from a stubbed version of the back office Application Programming Interface (API). This ensured that all users obtained the exact same data and should thus be able to draw the same conclusions from it. It also allowed us to prepare the data to contain interesting information, and observe whether subjects were able to extract this information from the visualisations. Figure 9.1 contains a rendering of the user's energy consumption for a specific day. In this case participants were asked about their working hours for that day, which was used to test the ease with which the chart could be interpreted.

Subjects that were in possession of an Android-based phone running version 3.2 (Honeycomb r2) or higher were allowed to use their own device. Those that were not were provided with an HTC Sensation running Android version 4.0.3 (Ice Cream Sandwich). Each subject was presented with a short introductory text about the application and was then asked to perform the case using the instructions found in the appendix. There was no time limit on the case. Users took an average of 17 minutes to answer the questions related to information extraction and perform the tasks required to do so.

The experiment was performed at a location of the subject's choosing. This was not only out of convenience for the subjects, but also to mimic the environment in which a user would typically use the app. By keeping the setup casual, the expectation is that the amount of distractions will be higher than in a controlled environment and thus give more relevant results. At the chosen location, a webcam was mounted using a rotatable arm so that it was able to record the

¹<http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users>

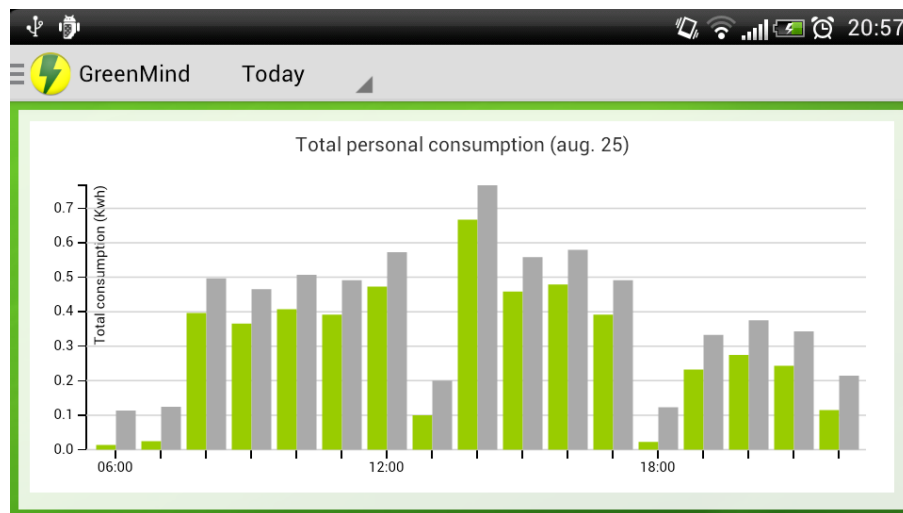


Figure 9.1: A sample screen presented to the participants of the evaluation

device's screen while allowing the user some freedom in using the device.

9.2 Observations

In order to discuss the results of the experiment, a number of observations were made for each subject. This implies that each participant is a single unit of observation, for which the session was recorded using a webcam and microphone. Besides audio and video, a logging tool was used to record the state and flow within the application. The camera recording also allowed us to identify interaction with the device that did not result in a change of state, since the logging tool would not have captured such information.

Based on the answers to the open questions of the questionnaire, a score indicating to what extent subjects were able to recognize and extract the information presented by the application was assigned. In the case that a subject was unable to answer a question, he was asked to indicate at which point he got stuck. This provided us with both a qualitative as well as a quantitative indication of possible defects.

9.3 Analysis

To answer our research questions, generalizations were made from the observations for individual users. This makes the entire set of participants our main unit of analysis. The generalized observations were interpreted with respect to *ease of navigation* and *ease of interpretation*, which are the two main concepts supporting our research questions. Due to the qualitative nature of the case study, the method for data extraction and analysis are to some extent open for discus-

sion. Analysis was done by reviewing the video and audio feeds on a per-task basis and searching for recurrence of behavior that can be used as evidence for answering our research questions.

With respect to uncovering defects in the application, observations for individual users were also analysed. Thus, individual users were also a unit of analysis. For this, data was extracted by downloading the logfiles from the used device after each case. Evidence for defects were extracted by filtering error messages from these logfiles. Note that with respect to our research questions this data is less interesting, but from the point of view of further development may be very valuable.

By performing the evaluation we were able to obtain both qualitative and quantitative measurements. In this section we will merely make note of the raw, aggregated results. However, it is good to be aware that both the audio and video feeds are also part of our resultset, containing indicators of the user experience such as small displays of emotions like surprise, frustration or satisfaction. We will discuss the conclusions we made from those observations in the discussion of our results in chapter 11. For details on the exact tasks and questions presented in the experiment, please refer to the appendix.

10.1 Exploratory tasks

Table 10.1 lists the completion of the exploratory tasks by participants. In case participants did not complete a task, they were asked to indicate where they got stuck. The motivations given by participants will be discussed in the next chapter.

Task/Question	Completion rate
1. Account login	100%
2. Determine device ownership	100%
3. Compare between self and others this week	100%
4. Compare between self, this week and this month	60%
5. Estimate impact of reduction on total building use	20%
6. Estimate impact of reduction on group use	40%
7. Estimate working hours	100%
8. Account logout	100%

Table 10.1: Task completion by experiment participants.

10.2 Questionnaire

After the exploratory tasks participants were asked to express their experience using the app using a short questionnaire. The results of this questionnaire are listed in table 10.2. Questions were answered on a scale of 1 to 5, 1 being the most negative possible answer and 5 being the most positive.

Question	Score
14. It is clear where to click	76%
15. App does not give unexpected response	80%
16. The presented data looks trustworthy	88%
17. The application is attractive	64%
18. The application feels like other applications	76%
19. The application is easy to navigate	80%

Table 10.2: Usability score as indicated by participants.

We will start our discussion of the user study results with a number of general observations made during the experiment. After that, we will proceed by discussing each of the tasks/questions separately. We will also suggest possible improvements to the issues discovered where applicable.

Reviewing the recordings from the experiment, we can state that participants were very expressive during the use of the application. This included talking out loud when stuck or when making a discovery from the presented data. This information is very valuable in the sense that it exposes subtle details which are hard to capture otherwise.

11.1 Exploratory tasks

The initial tasks 1, 2 and 3 presented to the participant asked to log in and make some simple observations. None of the participants failed in completing these tasks, though some had trouble using the Android on-screen keyboard. Most notably, we observed that 2 out of 5 participants had difficulty finding the main navigation drawer. Once found, all participants defaulted to the navigation drawer as a starting point when unsure how to proceed. Some participants tried tapping the screen title, expecting some navigation. This suggests to link the title to opening the navigation drawer as a possible improvement.

Another observation was that 4 out of 5 users attempted some form of touch-interaction with the chart showing user consumption, and that one of those tried swiping to navigate the data with respect to time. This use of touch and gestures may thus be a good starting point for the introduction of further exploration of data.

From table 10.1 we can conclude that participants had the most trouble completing tasks 4, 5 and 6. These tasks required linking data from multiple charts together in order to draw a conclusion. We can confirm the difficulty of these tasks from our qualitative measurements as well: both video and logfiles show that users keep *looking around*, unsure how to complete the task at hand. The audio for 3 out of 5 participants indicates a feeling of being lost or at least unsure how to proceed. For 2 out of 5 subjects, the distinction between groups and users was unclear while only 1 subject correctly linked the proportion of building consump-

tion to user consumption. This indicates that users of the app may have trouble associating the presented data to their context. This might be solved using more visual cues (such as group icons) or a short introduction when first starting the app (similar to startup tips).

The final two tasks could again be completed with information from a single screen and proved no issue for participants. One participant again tried the use of touch gestures to further explore the chart of the users hourly consumption. While we expected that users might simply log out of the application by using Android's *Home* button, all users were able to find the logout button in one or two attempts.

11.2 Questionnaire and other observations

Regarding the questionnaire, we observed that feelings about the *looks* of the app were mixed. However, attractiveness is a vague criterium and nearly all participants indicated that the *feel* of the app was familiar. The only participant having troubles with the feel was actually an experienced iPhone user, which may explain the awkwardness of navigation on an Android device. Finally, none of the participants indicated encountering unexpected results such as crashes or an unresponsive application. This suggests that our efforts for designing a structure for effective and efficient visualisation pays off in terms of user experience.

We also asked participants whether they expected or could think of any other functionality that might be part of a tool such as the GreenMind app. Not entirely unexpected, 3 out of 5 users indicated the desire to also compare consumption between their devices. This functionality was actually discussed for the GreenMind app but left out due to practical reasons (only 1-2 devices per user). However, this does strengthen the conclusion we made earlier that tools for eco-visualisation should allow the user to explore in order to experience a sense of control.

In this thesis we have presented a solution to the complexities of implementing effective (mobile) data visualisation. Using the GreenMind application as a leading example, we have discussed a number of theoretical aspects and tied them together using the visualisation pipeline as a basis. This has resulted in a solution that not only separates different implementation concerns, but also takes into account the specific challenges of the mobile context. Moreover, as a result of separating concerns, the process of rendering visualisations can be easily ported between platforms. This allows us to shift focus from solving implementation problems towards providing effective visualisation.

We started our discussion in part I with the foundations required for building data visualisations and the complexities of using visualisation to influence the behavior of users. This resulted in the conclusion that designing such visualisations is a complex task. This task may require advanced visualisations that provide the user with a sense of control, which in turn may lead to users deciding to change their behavior. Looking at the tools available for mobile data visualisation, we observed that current (native) libraries for the Android platform do not provide the flexibility and control needed to create such visualisations. Furthermore, by analysing the characteristics of the data supporting the visualisation, we discussed the use of a graph DBMS in order to be able to query and explore large amounts of temporal data effectively.

In part II we started analysing the specifics of the mobile context, using our findings to bridge the gap between our data model and the final rendering on a mobile device. This resulted in a design based on the traditional visualisation pipeline but tailored towards the mobile context. The performance and usability of the final application were evaluated during the case study, but a number of other design goals were also mentioned: modularity, integrity and security.

- Modularity was improved by taking the visualisation pipeline as a basis and tailoring it to the mobile context. Using a combination of native Android code and webscripting we were able to implement geometric mapping and rendering using the D3JS library and reuse the mapping and rendering code in the GreenMind hallway display.
- Data integrity and security were improved through the application's back

office. This component provided extensive preprocessing based on the model discussed in part I, while exposing the focused data to other components through an API. This created a single interface for data filtering and allowed us to keep authentication and authorization centralized.

We concluded this thesis with an empirical evaluation of the GreenMind app. This evaluation allowed us to verify the usability of the app and discover defects and other possible improvements. Most notably, we found that the use of touch-interaction and (swiping) gestures could be used to further increase the control a user has over the provided visualisations. Other suggestions for further research are discussed in chapter 13.

Altogether, we have presented a solution that approaches the issues from our problem statement and solves them by combining established techniques (such as the visualisation pipeline) with relatively new ones (such as a graph DBMS and hybrid mobile development). This solution solves many implementation challenges and as a result supports efficient implementation of (eco-)visualisations.

Chapter 13

Further research

In this thesis, we have kept the scope of the discussed aspects narrow. However, there is a wide range of research that could be done that may give insight into possible improvements for the GreenMind application. Also, there are plenty of options for expanding the app itself based on the findings in this thesis.

The most promising area for further research seems to be in the area of behavioral change. While we have provided a solid basis on which advanced visualisations can be build, the efficiency of these visualisations may be improved. New visualisations may also be added that attempt to trigger users in new ways. Research will have to show in what ways the current visualisations can be improved, for example by conducting a more elaborate evaluation of the application or A/B testing.

Not directly related to visualisation, but also very promising, is the use of gamification. Gamification combines both social and personal incentives (e.g. beating your own high score, receiving awards based on reduction in your consumption) to keep users interested over a longer period of time. While this area was initially out of scope, the mobile platform is very suited for such an approach.

From the perspective of the GreenMind application, further research could be done in the area of data exploration. Most of the discussed visualisations are used for presenting information at-a-glance, which is the logical result of using hallway displays and mobile devices. However, a web interface which is accessible via a desktop PC would allow interaction that surpasses the possibilities on a mobile device. As such, thought could be given to the further exploration of personal and general data consumption.

The dataset collected by the GreenMind project may also be interesting as a source for further research. Machine learning and pattern recognition techniques such as SAX may allow for numerical analysis of the data (as opposed to visual). This may in turn lead to new insights into how energy consumption may be reduced.

Bibliography

- [1] T. A. Nguyen and F. Nizamic, “Bernoulliborg - the building of sustainability,” tech. rep., University of Groningen, 2012.
- [2] B. Shneiderman, “The eyes have it: a task by data type taxonomy for information visualizations,” in *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pp. 336–343, 1996.
- [3] D. S. C. Fung, “Methods for the Estimation of Missing Values in Time Series,” Master’s thesis, Edith Cowan University, 2006.
- [4] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*, vol. First edition. O’Reilly, May 2013.
- [5] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins, “A comparison of a graph database and a relational database: a data provenance perspective,” in *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE ’10, (New York, NY, USA), pp. 42:1–42:6, ACM, 2010.
- [6] F. Holzschuher and R. Peinl, “Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j,” in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT ’13, (New York, NY, USA), pp. 195–204, ACM, 2013.
- [7] I. Assent, R. Krieger, F. Afschari, and T. Seidl, “The ts-tree: efficient time series search and retrieval,” in *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, EDBT ’08, (New York, NY, USA), pp. 252–263, ACM, 2008.
- [8] J. Lin, E. Keogh, and S. Lonardi, “Visualizing and discovering non-trivial patterns in large time series databases,” *Information Visualization*, vol. 4, pp. 61–82, July 2005.
- [9] W. Aigner, S. Miksch, W. Muller, H. Schumann, and C. Tominski, “Visual methods for analyzing time-oriented data,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 14, no. 1, pp. 47–60, 2008.

- [10] W. Muller and H. Schumann, "Visualization methods for time-dependent data - an overview," in *Simulation Conference, 2003. Proceedings of the 2003 Winter*, vol. 1, pp. 737–745 Vol.1, 2003.
- [11] T. G. Holmes, "Eco-visualization: combining art and technology to reduce energy consumption," in *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition, C&C '07*, (New York, NY, USA), pp. 153–162, ACM, 2007.
- [12] N. Valkanova, S. Jorda, M. Tomitsch, and A. Vande Moere, "Reveal-it!: the impact of a social visualization projection on public awareness and discourse," in *Proceedings of the 2013 ACM annual conference on Human factors in computing systems, CHI '13*, (New York, NY, USA), pp. 3461–3470, ACM, 2013.
- [13] L. Chittaro, "Visualizing information on mobile devices," *Computer*, vol. 39, no. 3, pp. 40–45, 2006.
- [14] L. Chittaro, "Visualization of patient data at different temporal granularities on mobile devices," in *Proceedings of the working conference on Advanced visual interfaces, AVI '06*, (New York, NY, USA), pp. 484–487, ACM, 2006.
- [15] J. Nielsen and R. Budiu, *Mobile usability*. Pearson Education, 2012.
- [16] X. Ma, B. Yan, G. Chen, C. Zhang, K. Huang, and J. Drury, "A toolkit for usability testing of mobile applications," in *Mobile Computing, Applications, and Services* (J. Zhang, J. Wilkiewicz, and A. Nahapetian, eds.), vol. 95 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 226–245, Springer Berlin Heidelberg, 2012.
- [17] C. Fischer, "Feedback on household electricity consumption: a tool for saving energy?," *Energy Efficiency*, vol. 1, no. 1, pp. 79–104, 2008.
- [18] B. Birzle-Harder and K. Götz, *Grüner Strom: eine sozialwissenschaftliche Marktanalyse*. Studentexte des Instituts für Sozial-Ökologische Forschung, Inst. für Sozial-Ökologische Forschung, 2001.
- [19] J. E. Petersen, V. Shunturov, K. Janda, G. Platt, and K. Weinberger, "Dormitory residents reduce electricity consumption when exposed to real-time visual feedback and incentives," *International Journal of Sustainability in Higher Education*, vol. 8, pp. 16–33, 2007.
- [20] W. Odom, J. Pierce, and D. Roedl, "Social incentive & eco-visualization displays: Toward persuading greater change in dormitory communities," in *Workshop Proc. Of OZCHI*, vol. 8, 2008.
- [21] T. Sohn, K. A. Li, W. G. Griswold, and J. D. Hollan, "A diary study of mobile information needs," in *Proceedings of the SIGCHI Conference on Human Factors*

- in Computing Systems*, CHI '08, (New York, NY, USA), pp. 433–442, ACM, 2008.
- [22] Gomez, “Why the mobile web is disappointing end users,” tech. rep., Gomez, 2009.
- [23] J. Nielsen, “Heuristic evaluation,” in *Usability inspection methods* (J. Nielsen and R. L. Mack, eds.), pp. 25–62, New York, NY, USA: John Wiley & Sons, Inc., 1994.
- [24] H. Zimmermann, “Osi reference model—the iso model of architecture for open systems interconnection,” *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 1980.
- [25] A. Gerber, S. Sen, and O. Spatscheck, “A Call for More Energy-Efficient Apps,” tech. rep., ATT Labs Research, 2011.
- [26] J. Zhang, J. Wilkiewicz, and A. Nahapetian, “A toolkit for usability testing of mobile applications,” in *Mobile Computing, Applications, and Services*, vol. 95 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 226–245, Springer Berlin Heidelberg, 2012.
- [27] P. Baxter and S. Jack, “Qualitative case study methodology: Study design and implementation for novice researchers,” *The Qualitative Report*, vol. 13, no. 4, pp. 544–559, 2008.

Appendices

Instruction

Dear test subject,

Thank you for participating in this user study. This study aims to assess the usability of the GreenMind Android application and find possible improvements. In this study, you will be asked to use this app to answer a number of questions. Afterwards, you will be asked to fill out a questionnaire about how easy or difficult it was for you to answer these questions. Please take your time and read the instructions carefully before starting the experiment.

During the experiment we will be using a webcam to record the smartphone's screen. Recording this session allows us to review how you are navigating through the application. You are free to pick up the device and hold it in any way that you are comfortable using it, as long as it is roughly in view of the camera. The device has also been connected to a laptop to monitor the application on a more technical level. The data that is collected from this experiment will only be used for analysis of the GreenMind app, and will in no way refer to you personally.

If you have any questions concerning the setup, or if you encounter any problems completing the tasks, you can ask the supervisor for support. The entire experiment will take approximately 20 minutes.

Introducing the GreenMind app

The application that is installed on the smartphone before you is called the GreenMind app. For this occasion, you will be using the app as if you were an employee at the University of Groningen - which you may or may not be in real life. The app allows you to look at the electricity consumption of some of the devices you use at the faculty. It also allows you to compare your personal consumption to that of others.

Exploratory tasks

Below you will find a number of tasks and questions that you can complete or answer using the GreenMind app. Some of these are straight-forward while others require you to interpret data presented by the application. Please take your time completing these tasks and inform the supervisor if you run into any problems.

1. An account has been created for you with username p1234567 and password Android. Open the GreenMind app and use this information to log in.
2. We already established that you are an employee at the University of Groningen. What devices do you own that are measured using the app?

3. How does the energy you consumed this week compare to that of other users?

4. Was this week representative for your average energy consumption? Why (not)?

5. Do you think that reducing your consumption by 30% would have a large impact on the consumption of the entire building? Please use data from the app in your answer.

6. Do you think that reducing your consumption by 30% would have a large impact on the consumption of your research group? Please use data from the app in your answer.

7. At what times of day do you estimate that you are typically in your office? Do you typically start early or work late?

8. Finally, end the session by logging out of the application.

Questionnaire

About you

9. How old are you? I am _____ years old.
10. What is your gender?
- ☐ Male
 - ☐ Female
11. We define a smartphone as a mobile phone with internet access and the ability to install custom applications (apps). What kind of operating system does your smartphone run?
- ☐ Android
 - ☐ iOS
 - ☐ Windows Phone
 - ☐ Other: _____
 - ☐ I do not own a smartphone (skip to question ...)
12. For how long have you owned a smartphone?
- ☐ Less than 6 months
 - ☐ 6-12 months
 - ☐ 1-2 years
 - ☐ More than 2 years
13. How many apps have you downloaded onto your smartphone?
- ☐ None
 - ☐ 1 to 5
 - ☐ 6 to 10
 - ☐ More than 10

About the GreenMind app

The following questions are about how easy or difficult you found it to find answers to the open questions. Please indicate for each question how you experienced using the app.

14. Did you feel like you knew what to do and where to click?

never ☐—☐—☐—☐—☐ always

15. Did you encounter any unexpected results?

often ☐—☐—☐—☐—☐ never

16. Did you feel you could trust the data presented by the application?

not at all ☐—☐—☐—☐—☐ very much

17. Did you find the application attractive?

not at all ☐—☐—☐—☐—☐ very much

18. Did this application act and feel like other applications you (may) use?

not at all ☐—☐—☐—☐—☐ very much

19. How easily could you find what you wanted with this application?

very difficult ☐—☐—☐—☐—☐ very easy

20. Would you be concerned with other people being able to see your personal energy consumption?

very ☐—☐—☐—☐—☐ not at all

21. Are there any features you would expect from an application showing energy consumption, that was not provided by the GreenMind app?
