LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
Department "Institut für Informatik"
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Heinrich Hußmann

Diploma Thesis

Developing a Dashboard for Real-Time Data Stream Composition and Visualization

Daniel Filonik
daniel.filonik@stud.ifi.lmu.de

Bearbeitungszeitraum:	01. 02. 2012 bis 14. 08. 2012
Betreuer:	Dipl. Medieninf. Max-Emanuel Maurer, Dr. Dominikus Baur
Verantw. Hochschullehrer:	Prof. Dr. Heinrich Hußmann

Abstract

Technological advances have led to an influx of affordable hardware that supports sensing, computation and communication. This hardware is increasingly deployed in public and private spaces, tracking and aggregating a wealth of real-time environmental data. Although these technologies are the focus of several research areas, there is a lack of research dealing with the problem of making these capabilities accessible to everyday users. This thesis represents a first step towards developing systems that will allow users to leverage the available infrastructure and create custom tailored solutions. It explores how this notion can be utilized in the context of energy monitoring to improve conventional approaches.

The project adopted a user-centered design process to inform the development of a flexible system for real-time data stream composition and visualization. This system features an extensible architecture and defines a unified API for heterogeneous data streams. Rather than displaying the data in a predetermined fashion, it makes this information available as building blocks that can be combined and shared. It is based on the insight that individual users have diverse information needs and presentation preferences. Therefore, it allows users to compose rich information displays, incorporating personally relevant data from an extensive information ecosystem.

The prototype was evaluated in an exploratory study to observe its natural use in a real-world setting, gathering empirical usage statistics and conducting semi-structured interviews. The results show that a high degree of customization does not warrant sustained usage. Other factors were identified, yielding recommendations for increasing the impact on energy consumption.

Zusammenfassung

Technologische Fortschritte haben dazu geführt, dass die Verfügbarkeit von kostengünstiger Hardware zum Aufnehmen, Verarbeiten und Kommunizieren von Daten stark angestiegen ist. Diese Hardware wird zunehmend in öffentlichen und privaten Bereichen eingesetzt, wo sie eine Fülle von Umweltdaten in Echtzeit verfolgt und aggregiert. Obwohl diese Technologien im Fokus von mehreren Forschungsgebieten stehen, herrscht ein Mangel an Forschung, die sich mit dem Problem auseinandersetzt diese technischen Möglichkeiten normalen Nutzern zur Verfügung zu stellen. Diese Arbeit stellt einen ersten Schritt zur Entwicklung von Systemen dar, welche zukünftigen Anwendern ermöglichen sollen die bereitstehende Infrastruktur zu nutzen, um für sie maßgeschneiderte Lösungen zu erstellen. Sie untersucht, wie man diese Idee im Kontext von Energie-Monitoring nutzen kann, um konventionelle Ansätze zu verbessern.

Das Projekt hat einen nutzerorientierten Gestaltungsprozess angewendet, der die Entwicklung eines flexiblen Systems zum Zusammenstellen und Visualisieren von Echtzeit-Datenströmen entscheidend geprägt hat. Dieses System verfügt über eine erweiterbare Architektur und definiert eine einheitliche Programmierschnittstelle für verschiedenartige Echtzeit-Datenströme. Anstatt diese Daten in einer festgelegten Art und Weise anzuzeigen, stellt das System diese Informationen als Bausteine zur Verfügung, welche kombiniert und ausgetauscht werden können. Es basiert auf der Einsicht, dass einzelne Benutzer vielfältige Informationsbedürfnisse haben sowie unterschiedliche Präferenzen bezüglich der Präsentation. Deswegen erlaubt das System dem Anwender eine Anzeige mit umfangreichen Informationen zusammenzustellen, inklusive persönlich relevanter Daten aus einem reichhaltigen Informationsökosystem.

Der Prototyp wurde im Rahmen einer ersten Studie evaluiert, um die natürliche Nutzung in einem realen Umfeld zu beobachten. Dabei wurden empirische Nutzungsstatistiken gesammelt und halb-strukturierte Interviews geführt. Die Ergebnisse zeigen, dass ein hoher Grad von individueller Anpassung nicht zwingend fortdauernde Nutzung herbeiführt. Andere Faktoren wurden identifiziert, auf deren Basis Empfehlungen aufgestellt werden, um den Einfluss auf den Energieverbrauch zu erhöhen.

Aufgabenstellung

People are faced with the increasing availability of urban data and services in their everyday life. At the same time, residents themselves are more and more becoming producers of data that is linked to their environmental performance, such as domestic power consumption, generated PV solar power, water, gas, petrol use, etc. While there are efforts to make public data accessible (e.g. http://data.australia.gov.au) and share privately generated data (e.g. http://pachube.org), many sources of data are still proprietary and only accessible through a limited set of services or websites. In order to solve a particular problem, such as commuting to work, a person has to access a variety of different services, e.g. bus schedules, weather information, road congestion, availability of parking, etc.

The goal of this project is to design and develop a prototype of a dashboard interface that will allow users to compose and visualize relevant data streams side-by-side. The purpose of the dashboard is to give the user a comprehensive and intuitively accessible overview of data that relates to their local household. The project acknowledges that individual users have different preferences with regard to the data they want to see displayed and the style in which they want it to be presented.

In order to account for personal preferences, the project aims to create a repository of widgets that can easily be upgraded and extended. As a result, the dashboard will be able to accommodate new sources of data and expand the range of available metrics and visualizations. Unlike existing applications, the dashboard will cater to personal interests by integrating data from social networks, popular web services and government databases.

As part of the evaluation, the project will focus on data streams related to a user's environmental performance. The study will investigate whether a customized dashboard display can raise awareness of personal energy consumption. The Dashboard prototype will be developed for iPad tablets and evaluated in a user study with Brisbane residents.

- Familiarize self with work in the areas of *Urban Computing* and *Information Visualization*.
- Conduct a focus group discussion to collect requirements for the Dashboard prototype.
- Create framework for working with heterogeneous real-time data streams.
- Develop a Dashboard prototype, including an exemplary set of visualization widgets.
- Conduct a user study to evaluate the Dashboard prototype.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, 13. August 2012

. . . .

Acknowledgments

I would like to express my gratitude for all the help from my supervisor Dipl. Medieninf. Max-Emanuel Maurer, as well as the entire team at the Urban Informatics Research Lab at Queensland University of Technology in Brisbane, Australia. In particular, I would like to highlight the central roles of Assoc. Prof. Marcus Foth and Dr. Markus Rittenbruch in making this project possible. Finally, I would like to thank my mother, Anna Filonik, for her unconditional support and encouragement throughout all of my personal and academic endeavors.

Contents

1 Introduction

This thesis investigates the potential of real-time data stream composition and visualization in the context of energy monitoring. The ultimate goal is to provide end-users with tools that allow them to combine information from various data sources in order to support day-to-day activities and allow them to make informed decisions. As a first step towards exploring this concept, a prototype of a customizable Dashboard system was developed and evaluated in a user study.

The majority of the work was carried out abroad in cooperation with the Urban Informatics Research Lab at Queensland University of Technology in Brisbane, Australia. It is a continuation of previous research conducted at the lab. Furthermore, it ties into an overarching study aimed at connecting people to their resource consumption through real-time data visualization. Throughout the course of this project, there was a lively exchange with local PhD students. In particular, the evaluation of the system was coordinated with ongoing PhD research that explores novel concepts to encourage domestic energy conservation through real-time in-situ electricity monitors in Brisbane homes.

The widespread deployment of hardware for sensing various types of data is becoming technologically and economically feasible. These devices are being installed in public urban areas as well as private spaces such as smart homes. As a result, citizens are increasingly producers as well as consumers of real-time data streams. The purpose of the Dashboard prototype is to give the user a comprehensive and intuitively accessible overview of data that is relevant to their local household. The project acknowledges that individual users have different preferences with regard to the data they want to see displayed and the style in which they want it to be presented.

The development of the Dashboard system was guided by the experience and expertise of Assoc. Prof. Marcus Foth and Dr. Markus Rittenbruch. In line with prior research work at the Urban Informatics Research Lab, the project adopted a user-centered design philosophy. Therefore, a key contribution of this thesis lies in the evaluation of the developed Dashboard prototype with actual users, observing their natural interaction with the system in the environment where they would normally use it. The goal of the study was to determine the viability of the Dashboard concept and to identify potential problems. A central question was how participants would make use of customization aspect of the composable information display and whether it would influence their interaction with the system. The results showed that a high degree of customization did not necessarily warrant sustained usage. Furthermore, the observations revealed several challenges for the adoption of such decision support tools.

1.1 Energy Challenges

One of the greatest challenges that humanity is facing today is the development of a sustainable and secure energy supply. The technological progress since the industrial revolution has facilitated enormous economic growth and prosperity. However, this progress comes at the cost of an ever increasing demand for energy. Most developed countries meet a large portion of this demand by burning fossil fuels (coal, oil and gas), which leads to large quantities of carbon dioxide being released into the atmosphere.

Carbon dioxide is the most significant human contribution to *greenhouse gases*, other noteworthy components being methane and nitrous oxide, which originate mainly from modern farming and agriculture. The IPCC – a panel of international experts representing the most authoritative scientific consensus on the topic climate change – has asserted that the warming of the climate system is unequivocal and it can be attributed to human activities with very high confidence. In fact, most of the temperature increase can be explained by the increase of greenhouse gas concentrations in the atmosphere [68].

There is a wide range of motivations for developed countries to reduce the dependence on fossil fuels – ecological, economical, political and ethical.

From an *ecological* point of view, the change in temperature is expected to have serious repercussions. Depending on the geographical location various severe weather phenomena are predicted to occur with increasing frequencies and intensities. For example, across Europe these events range from storms and floods to droughts and soil erosion. Other countries are even more vulnerable to the changing climate. Australia, for example, will have to cope with extreme heat and rising sea levels. Around the world, rising temperatures threaten various sensitive ecosystems and biodiversity.

Many of the *economical* implications stem directly from the *ecological* consequences, as large sectors of the economy are expected to be affected. Most notably, the changing climate will have a direct effect on agriculture, leading to a reduction in crop yields and water shortages. The total cost of failing to act on climate change is estimated between 5% and 20% of global gross domestic product, whereas a transition to an energy efficient low-carbon economy would approximately amount to 0.5%. However, on the flipside, moving away from fossil fuels holds numerous economic opportunities. Eco-industries are expected to have strong growth as the demand for green and renewable energy technologies increases. Failing to invest in these areas will give other countries a competetive edge and result in a possible brain drain [21].

From a *political* perspective, reliance on fossil fuels is a sensitive issue in the countries that need to import these energy sources. A secure energy supply is indispensable to the industrial sector. Therefore, a dependency on a small number of foreign suppliers is undesirable. The prices can change dramatically due to shortages and regional conflicts, causing volatility in the markets.

Finally, it is our *ethical* obligation towards future generations to manage our finite resources in a responsible manner. The effects of temperature changes are expected to last centuries due to the slow climate processes and feedback mechanisms involved. Another *ethical* issue is that the poorest developing countries are especially vulnerable to climate change [21, 68].

Realizing the importance of acting against climate change, the EU is striving to take a leadership position in this endeavor. It has set the goal to limit global warming to less than 2 °C relative to pre-industrial levels. As a critical step towards achieving this goal, the EU member states have unilaterally committed themselves to reduce greenhouse gas emissions by a minimum of 20% by 2020. If the developed countries fail to act collectively, it is likely that the 2 °C threshold will be crossed by 2050 [21, 42].

Several objectives have been outlined in order to reach the emission targets. For one, the EU is aiming to substantially increase the market share of renewable energy sources such as wind, water and solar. Further, the EU is imposing limits on emissions in energy-intensive industries, power generation and aviation with an emissions trading scheme. However, the strategy can only succeed if it is complemented by continuing efforts to improve energy efficiency [21].

In a nutshell, the plan to decrease dependency on fossil fuels and reduce greenhouse gas emissions rests on two central pillars. First, lowering the demand for energy by achieving higher energy efficiency. Second, increasing the portion of renewables generating the energy supply. Both of these factors play a fundamental role for developing an environmentally sustainable energy policy. The importance of these two areas is amplified in countries that do not want to rely on nuclear power as a bridging technology. This is the case in Germany, for example, which enacted a nuclear phase-out in the wake of the Fukushima disaster.

The key area with regard to this work is the effort to increase energy efficiency. Technological advances are showing considerable promise for reducing energy consumption. The biggest potential savings lie in the areas of mobility, domestic energy consumption and building renovation/modernization. It is estimated that European households could generate energy savings of up to 30% by purchasing more efficient electrical appliances [10, 42].

Lifestyle and behavior patterns play an important role in mitigating climate change. Emphasizing the importance of resource conservation can bring about changes of individual behaviors and cultural patterns resulting in significant reductions of energy use. Further, educating people about the need to conserve energy is an important factor for fostering market acceptance of efficient products. This, in turn, leads to increased consumer choice [47].

However, increased efficiency can also have pitfalls in the form of direct and indirect rebound effects. On the one hand, a *direct rebound* occurs when a consumer immediately reinvests the savings generated from an energy efficient technology. This can happen in the form of increased use of that same technology or the use of an entirely different one. On the other hand, an *indirect rebound* describes a situation when energy usage declines to such an extent that the price drops, which consequently stimulates consumption. Further, the production of energy efficient technologies can be very energy intensive in and of itself. Therefore, it is important to carefully consider and weigh the effects of introducing new interventions, as well as constantly monitor and reevaluate them [10].

1.2 Technological Developments

We live in a time of rapid technological progress. Hardware enabling sensing, computation, communication and actuation is becoming smaller and cheaper to manufacture. The integration of these capabilities can be observed on various scales, from everyday objects to smart homes and urban environments. Through networking technologies, the individual building blocks are connected to form large systems with ever increasing complexity. Vast amounts of information are accumulated and processed in such systems. While this may be a fairly recent phenomenon, Steenson points out that communication technology has been improving since the early 19th century, leading to an increase in the availability and timeliness of information. This development is key for the growth of an intelligent society [70].

One of the key technological achievements of our time are the vast telecommunications networks that span the globe. An ever increasing share of the population in developed countries is accessing the Internet on a daily basis, although there are still some areas where availability of broadband access is limited. Nevertheless, the web has transformed many diverse aspects of our lives, such as education, commerce and entertainment. It has introduced new means for human communication and social interaction. In short, the Internet has become an essential infrastructure for our modern civilization [32].

Although the Internet has already had a profound influence on our daily lives, the ways in which we experience it are still fundamentally changing. Desktop computers are loosing their dominant position as the primary device for browsing the web. The year 2012 it is expected to mark the first time when a majority of users will connect through mobile and wireless devices [32]. As the adoption of smart phones and tablets is rising sharply, they enable completely new kinds of content and services. These connected devices make it easier than ever to share and retrieve information. Users act as consumers – making use of government data or other urban services – and increasingly also as producers – contributing to social networks or collecting data in their domestic environment. Companies are becoming more aware of the value that third party applications can bring to their products and choose to make data accessible through open APIs. Similarly, governments are working to make public data more accessible.

Further, the Internet has given rise to the open source movement. In order for a piece of software to be considered open source, its source code needs to be made available under a permissive license that allows unrestricted redistribution and creation of derivative works. This philosophy has paved the way for an unprecedented amount of knowledge sharing and collaboration across the world. Various individuals and organizations contribute anything ranging from small domain specific libraries to full featured software packages. This creates a rich repertoire for developers to draw from and it allows them to apply the available tools in new areas [55].

A recent development originating from the open source movement are single-board microcontrollers and computers like *Arduino*, *BeagleBoard* or *Raspberry Pi*. These open hardware platforms empower amateurs to experiment and quickly prototype innovative ideas. Often, this work is referred to as physical computing, because the platforms are capable of controlling sensors and actuators in order to interact with the physical environment. Large online communities form around these platforms with members constantly exploring new and creative uses of the hardware.

Commercial products are also starting to integrate sensing and actuation hardware. Such products range from intelligent home control systems to quantified self tools that track aspects of a person's everyday routine. The culmination of this trend is the Internet of Things, a scenario in which all objects of daily life can be identified and exchange information with computers. Online services like *Cosm*[1] – formerly named *Pachube* – aim to become platforms that support this development by aggregating and hosting sensor data.

These technological developments provide a fertile breeding ground for developing solutions to the challenges of our times. They open up technology to a broader audience and encourage collaboration. As a result, they have created a 'hacker culture' that turns passive users into active developers creating solutions for their specific needs. This thesis leverages several components from the open source sphere. However, more importantly, it hopes to capture the 'hacker spirit' by allowing people to customize and extend the software on various levels according to their level of skill.

1.3 Technologies Applied to Energy Challenges

Drawing on the potential of existing and emerging technologies holds great promise for addressing our energy challenges. In many areas, technological advances can significantly reduce inefficiencies and prevent energy wastage. The rising demand for mobile devices has been a driving force behind the development of low power processors. When used in mobile devices, these processors provide longer battery life without sacrificing performance. Other products are showing similar efficiency gains without compromising convenience. However, such improvements can only be part of the solution. Therefore, it is necessary to leverage other technological developments to generate additional savings. For example, communication networks can increase availability and timeliness of information about energy consumption. In turn, this will allow energy providers to become more efficient and enable intelligent systems that optimize energy use.

The energy industry is taking steps to modernize their network infrastructure to fit the demands of modern energy policies. Renewable energy sources introduce new challenges for the utility companies due to distributed generation and supply fluctuations. It is necessary for electric utilities to upgrade the existing power grid in response to these challenges. The resulting *smart grid* is characterized by a new networking layer for bidirectional data transfer. This allows two-way communication between the supply-side and the demand-side. Therefore, it will enable electric utilities to fine-tune energy supply and act in synergy with the consumer [37].

The *smart meter* is a key component of the new energy infrastructure. They are the connection point of households to the *smart grid*. At intervals of less than one hour, these devices measure, store and transmit energy consumption to the electric utilities for monitoring and billing purposes. They facilitate decentralized energy generation by separately tracking the amount of electricity being fed back into the grid. Further, they provide the basis for variable tariffs, which can provide financial incentives to conserve energy [37].

The long term vision for the *smart grid* includes *smart homes* equipped with networked appliances that can be remote controlled to balance energy load and optimize efficiency. In the short term, it is likely that *smart meters* will act as a catalyst for the introduction of efficient technologies. However, there is concern that smart metering by itself is susceptible to rebound effects and

[1]http://cosm.com

not an effective tool for changing energy consumption habits. Thus, a successful energy reduction strategy requires us to look beyond the efficiency gains of *smart homes*. In order to affect lasting behavior changes, other approaches have to be explored [3, 10, 37].

An *energy monitor* offers comparable functionality to a *smart meter* with regard to measuring energy consumption. However, these devices follow a consumer oriented approach with a clear focus on providing energy usage information to end-users. Depending on the model, they can be plugged in to a power point or installed at the switch box, measuring consumption of individual devices or the entire household. Being limited in scope and independent from electric utilities, these devices have the potential to reach the market quicker. In order to be commercially successful, companies are putting greater emphasis on providing accessible user interfaces and delivering an enjoyable user experience.

However, the data generated through energy monitoring has to be viewed within the context of a busy landscape of information. Considering the distractions of modern life, it is easy to see energy data being drowned out by other attention-grabbing media. Rather than competing with other information sources, the Dashboard design proposed in this thesis aims to incorporate them, resulting in a display that is highly relevant to the user. Simplification is a crucial requirement for the effective communication of such diverse information. Therefore, the Dashboard application uses techniques from information visualization in order to provide the user with information relevant to their objectives at a glance. Additionally, the use of appropriate visualizations helps to make abstract and complex information more accessible.

1.4 Structure of Thesis

This thesis is divided into nine separate chapters, which are briefly outlined in the following paragraphs. The text loosely mirrors the chronological progress of the project, deviating whenever it makes sense thematically and enhances the reading flow.

Section 1 The *Introduction* provides real-world background on the energy challenges of the 21st century. These challenges were a key driver behind the development of the Dashboard system, which was applied in the context of domestic energy conservation.

Section 2 The *Related Work* section presents a comprehensive literature review, covering the key research areas of *Ubiquitous Computing*, *Information Visualization* and *Human-Computer Interaction*. This foray ends with a brief look at the use case of energy monitoring, examining some of the existing commercial solutions.

Section 3 The *Methodology* section summarizes the main research methods that were deployed during this project. In particular, it outlines the iterative development process, which was informed by user-centered design practices.

Section 4 The *Design Process* details the main influences that shaped the design of the Dashboard system. In particular, the *Focus Group* and the *Expert Review* are highlighted for their impact on the functional requirements of the final prototype.

Section 5 The *System Functionality* section takes the reader on a tour of the Dashboard system, introducing the web service and the native tablet application. It covers the specific responsibilities of each component and traces the task flows through the system.

Section 6 The *Implementation* of the Dashboard system is subsequently described in detail, including the rationale behind the major technology choices. The section starts with a high-level overview of the system architecture and branches off in two directions, diving into the server-side and client-side implementation. These parts are broken down further, illustrating individual components of the modular software design.

Section 7 The *Evaluation* section describes the extensive field trial of the Dashboard system, spanning two separate study phases. Each phase yielded a wealth of quantitative and qualitative data, which is documented in the respective subsections.

Section 8 The *Discussion* presents the research findings gathered during the exploratory user study. It highlights opportunities and challenges for the adoption of flexible decision support tools in the context of domestic energy conservation. Furthermore, it acknowledges the limitations of the present research and provides an outlook for future work.

Section 9 The *Conclusion* summarizes the outcomes of the project and discusses its contribution to future research.

2 Related Work

This thesis finds itself at the cross section of several major research areas. The current chapter introduces recent work within these areas and identifies the key challenges with regard to this project. First, the literature review explores *Ubiquitous Computing*, as well as the diverse streams of research branching from it. Next, the discussion focuses on *Information Visualization* and highlights the different ways it has been used in the context of energy consumption. Finally, this overview looks into *Human-Computer Interaction* and the insights it can provide for improving tools for energy monitoring.

Parts of this literature review have been published in an article for a special issue of the "Journal of Urban Technology" on "Street Computing".

2.1 Ubiquitous Computing

The diffusion of computational capacities into our surroundings is the subject of a number of research areas. Weiser laid the groundwork for *Ubiquitous Computing* in his article "The Computer for the 21st Century" [74]. His vision called for seamless interaction between users and numerous small computers embedded in everyday objects. Weiser argues that such an approach would give users better access to existing and previously unavailable information. For Weiser, the integration of computers into the human environment is essential in order to allow users to interact with this information in a natural way. Consequently, he sees *Ubiquitous Computing* as a promising solution for the problem of information overload [74].

More recently, researchers have started to point out the disconnect between this envisioned future and the way technology has evolved. While *Ubiquitous Computing* has already become a reality "in the form of densely available computational and communication resources" [4], the devices through which information is accessed are "highly present, visible, and branded" [4]. Nevertheless, a lot of research has been invested in developing technologies and applications that aim at making Weiser's vision a reality. The range of research topics is very broad and covers a large scope of application areas. However, many of the projects limit themselves to "small-scale well-defined patches of the built environment such as smart houses or rooms" [35].

In contrast to that, *Urban Computing* is a relatively recent area of research that looks at the impact of ubiquitous information processing at the scale of a city. The focus shifts from integrating computing into everyday objects towards everyday urban settings and lifestyles [35]. Rather than formulating a particular vision for the future, *Urban Computing* aims to explore and understand the implications of emerging technologies on urban landscapes today [58]. Most importantly, it takes into account the complex and dynamic social interactions that take place in cities.

Another research field revolving around the role of users is *Participatory Sensing*, which looks at the potential of computational capabilities in the hands of the general public. Using everyday mobile devices, the goal is to "enable public and professional users to gather, analyze and share local knowledge" [9]. Unlike traditional distributed sensing, *Participatory Sensing* has to consider issues like ensuring data credibility, protecting privacy, and encouraging participation [9]. If these challenges are properly addressed, applications in this space hold the potential to engage and empower citizens in new ways [58].

The previously mentioned research areas investigate the integration of information processing technologies into the environment from several different perspectives. However, a literature review shows that there is a lack of research dealing with the problem of making these computational capabilities accessible to everyday users. In order to address these challenges the team at the Urban Informatics Research Lab introduces the notion of *Street Computing*, which draws heavily from the existing pool of research. *Street Computing* tries to develop tools and techniques that will help users help themselves. Metaphorically speaking, it is taking computing to the street by giving the general public – rather than researchers and professionals – the power to leverage the available infrastructure and create solutions tailored to their individual needs. It is inspired by Colburn's "Street Science" in that it recognizes the importance of local urban insights to improve scientific inquiry as well as policy and decision-making. Local involvement delivers crucial information for solving problems in urban communities and increases public trust, leading to a healthier political system [14]. Furthermore, by providing users with the right tools, *Street Computing* enables citizens to take a more active role in the evolution of their cities and thereby reduces the burden placed on government services.

The following sections present the major challenges and open research questions in the area of *Street Computing*. These issues where identified by reviewing literature from related research areas and analyzing projects that fit into this particular space. This article aims to point out reoccurring problems and group them so they can be addressed in future research.

Integrating Heterogeneous Data Sources

Already today, our urban environments are monitored and analyzed through distributed computing and communication technologies. Complex and mostly independent systems support our everyday activities in the city, such as public transportation, mobile communication and financial services. Further, various corporate and government entities collect data about traffic, pollution or weather. It is foreseeable that this computing environment will incorporate an even wider variety of devices and services from different manufacturers and developers. In order to accomplish the goal of making this wealth information from urban systems accessible to users, *Street Computing* needs to address the issue of working with heterogeneous data sources. Therefore, to provide a solid foundation for future projects it is crucial to achieve platform and vendor independence, as well as architecture openness [41].

Zambonelli envisions those separate urban systems as individual organs that can "collaboratively contribute capabilities and services aimed to support increasing quality of life levels" [78]. He advocates the development of proper collaboration and middleware tools that allow "all the components of the infrastructure [...] to dynamically and adaptively connect and collaborate with each other" [78]. Such tools would greatly simplify the development of applications for *Street Computing*. Currently, most projects resort to creating this functionality from scratch, investing considerable time and effort. Providing adequate solutions to this challenge would benefit scientific progress by allowing researches to focus on other questions that lie beyond.

For instance, the *Weave* architecture is one specific effort to develop standard APIs for services offered by sensing fabrics. Its goal is to simplify the design of applications across heterogeneous sensor networks that are independently managed. It recognizes that it is often necessary for applications in urban sensing to "fuse information from sensing fabrics extant in the environment, to meet requirements for which the fabrics were not designed a priori" [39]. The work distinguishes between a generic API for urban sensing and a vertical APIs specific to application domains. The architecture is put to the test by focusing on the domain of search applications and applying it to concrete scenarios [39].

While it may not be feasible or even desirable to try and formulate an all-encompassing generic standard for *Urban Computing* – given the wide variety of application domains and problems – it would be helpful to have guidelines and best practices that manufacturers and developers can follow. Certain aspects can certainly be abstracted, such as the protocols for publishing and subscribing to data streams. However, even with a successful attempt at standardization, it is still necessary to have ways of integrating legacy and non-conformant systems.

Extracting Information and Knowledge

As the technologies for gathering data are widely deployed within urban areas, including smart homes, the leading question becomes how to process the collected data. The relationship between data, information and knowledge is explored in information systems and knowledge management literature. It is often represented as a hierarchy in which the higher levels are increasingly rich with meaning. While this concept is generally agreed upon, there is less consensus about "the processes that transform elements lower in the hierarchy into those above them" [61]. According to the classical definition by Ackoff, knowledge allows us to apply data and information to give answers to questions of 'how' [1]. Such questions lie at the core of the problems that urban communities face, for example "How to reduce traffic congestion?", "How to support healthy and sustainable lifestyles?" or "How to plan and develop neighborhoods with a high quality of life?".

Comprehensive ontologies suitable for *Urban Computing* are a critical prerequisite for applying semantic technologies. In turn, these technologies hold enormous potential for *Street Computing* by semantically interpreting the sensor output and thereby allowing non-technical users to benefit from data which would otherwise be meaningless to them. In the design of the search API for the *Weave* architecture proposed by Kulathumani et al. sensor fabrics are viewed as providers of an extensible database of queryable objects. As part of their future work, the authors aim to extend the search functionality to enable queries with richer semantics. In order to accomplish this goal, they are planning to develop "a richer ontology for urban sensing that builds up from physical phenomena, sensing signatures, detectors, and richer classes of objects" [39].

Effective decision-support systems require actionable information and knowledge about the situation in question. In order to provide this information, algorithms need to be developed that can combine static knowledge about the urban environment with the dynamic data streams originating from sensors. Della Valle et al. propose a system that adapts reasoning techniques to work with the resulting time-varying knowledge. Their solution combines semantic reasoners with throughput-efficient data stream management systems [17].

The *Semantic Streams* framework by Whitehouse et al. presents another approach that supports the quick extraction of semantic information from sensor output. The framework is based on a declarative language for specifying and composing inference units, which are "minimal units of sensor data interpretation" [75]. Existing inference units can be dynamically composed to generate new interpretations of sensor data. This allows non-technical users to pose queries over semantic interpretations of sensor data without having to write code performing the inference themselves [75].

Supporting Mashups and End-User Programming

An area that is still largely unexplored in the context of *Urban Computing* is the potential for *End-User Programming*. This aspect is a fundamental part of *Street Computing* and its vision of empowering users. Making it possible for the general public to leverage the existing technical infrastructure and create solutions tailored to their individual needs requires the development of new tools and techniques. A user-centric approach is necessary to make the available functionality easy to understand and use.

The issue of *End-User Programming* can be addressed from several different directions. There is substantial room for improvement when it comes to querying the sensing environment. Users need better ways of retrieving the information relevant to their individual problems. Query languages based on semantic interpretations of sensor data permit a more natural way of inquiry by allowing the use of high-level concepts. This relieves the burden of making low-level decisions based on the raw sensor data from the user. Such an approach has the additional benefit of introducing an extra degree of freedom to the system. The user does not specify which exact sensor data is used and how it is processed to infer the desired information [75].

Another means of approaching *End-User Programming* is through the composition of existing data and services. This has recently come to the center of research in the realm of web services, where such compositions are referred to as *Mashups*. Several different paradigms of *End-User Programming* have been explored with regards to creating *Mashups*. Among those are simplified domain-specific languages, programming-by-example, and visual programming based on the data flow metaphor. The latter is a rather prominent paradigm that has been employed in several commercial implementations, such as *Microsoft Popfly* or *Yahoo! Pipes*. In this approach, the composition is represented as a graph of components, each of which has input and output parameters. A component transforms the data received in the input parameters and stores the result in the output parameters. By creating connections, the output parameters of one component act as input parameters of another, inspiring the notion of a flow of data through the graph [6, 76].

The *SensorMasher* platform is one specific example for adapting the concept of *Mashups* to sensing environments. It specifies how to publish sensors on the web and access their associated data through URIs. Further, it provides ways of linking these published entities to ontological concepts. This information is used in the visual composer to provide an intuitive interface for navigating and exploring sensor data sources by following semantic links. Once the user finds the relevant sensor data sources, he can use the work flow editor to visually combine the data and apply data processing operators. The resulting *Mashup* can be published and put to work in subsequent compositions [40].

Further research is necessary to explore ways for users to control the ubiquitous computational capabilities that form the basis for *Street Computing*. Chin et al. extend programming-by-example to a *Ubiquitous Computing* context by allowing users to "show" the system the desired behavior "via natural physical interaction with the environment" [13]. The system deduces rules based on the actions that users perform with the physical devices themselves or with the corresponding interface elements. Beyond that, other approaches incorporate tangible user interfaces to represent key programming concepts in the course of the interaction between the user and his surroundings [13, 27].

Contributing and Sharing Resources

Street Computing aims to develop systems that allow the general public to make use of distributed computational resources to gather information about urban environments. At the same time, the public should be encouraged to contribute resources at their disposal. With increased availability of resources, the potential applications of the system become broader. However, resources may be exclusive or constrained, making it necessary to coordinate access and resolve conflicts. For example, if a user chooses to supply his mobile device as a sensor, battery life would be a resource limiting how the sensing is carried out.

In order to design a system that relies on collaboration and sharing, it is essential to understand the factors that motivate people to participate. For certain users, the driving force might be seeing how their data contributes to a larger whole. In other cases it may be necessary to identify and design novel interaction mechanisms to provide incentives for participation [16, 78]. Taking resource constraints into account, one viable approach might be to grant preferred access to those users who also contribute to the system.

The problem of limited shared resources in urban sensing systems has been addressed in different research projects. It becomes especially relevant if the systems provide *End-User Programming* capabilities, which are inherent in *Street Computing*. The *CitySense* project is one such example. In order to tackle this issue, resources are exposed through a custom API, which serves the purpose of restricting the programming model and allowing the underlying framework to perform optimizations such as caching [50].

Another approach for distributing load and resolving resource conflicts is taken by Whitehouse et al. and their *Semantic Streams* framework. The declarative specification language provides the capability to define quality of service trade-offs. Their framework uses this information together with its semantic inference capabilities to dynamically select the best available resources for answering a particular query [75].

Providing Security and Privacy

Sharing and collaboration are fundamental components of *Street Computing*. Its applications can only thrive in an environment where users are encouraged to participate and feel comfortable to exchange data. Individual contributions of local and domain-specific knowledge are key to solving complex problems in urban communities. In order to facilitate such an exchange, it is crucial to have fully-fledged privacy and security mechanisms in place.

The data that drives *Street Computing* poses some unique challenges in terms of privacy. It has a high potential for revealing personal information, "because urban sensing collects information in environments inhabited by and directly connected to human beings" [16]. From mobile devices to cars or smart homes, users carry sensors on their bodies or have them installed on their property. As a result, sensing device custodians are engaging in constant self-surveillance. Furthermore, this monitoring may compromise of privacy of those who are not custodians or primary sensing targets – an effect that Campbell et al. liken to "second-hand smoke". Finally, the collected data may reveal information that goes beyond the original purpose of the sensing system [11, 16]. Adams and Sasse aptly summarize this problem, stating that "most invasions of privacy are not intentional but due to designers inability to anticipate how this data could be used, by whom, and how this might affect users" [2].

Given the sensitivity of the information, it is unlikely that users will participate in *Street Computing* if they do not trust the underlying systems. One aspect of establishing trust is putting the users in charge of their personal data. This includes providing ways to control how personal data is collected, distributed, and processed. When sharing their data, users should have an option to become anonymous or pseudonymous and select the level granularity [16]. Designing usable and effective user-interfaces for controlling privacy and security capabilities is challenging. One major reason is that "users value and want security and privacy functionality, but they regard this functionality as secondary to completing their primary tasks" [33]. Therefore, additional research in the area of *Human-Computer Interaction* is necessary to develop solutions that deliver a better user experience [33].

Different approaches have been proposed that allow people to make contributions to urban data collection efforts while providing strong individual privacy. One such solution is *PriSense*, which is based on data slicing and mixing and supports a wide range of statistical aggregation functions. Instead of directly answering a query, sensor nodes split their data into slices and pass them to randomly chosen cover nodes where they are mixed. From the responses of the different nodes, the initiator of the request can calculate an accurate aggregation result [64]. A different approach is taken by *PoolView*, which is based on data perturbation. The idea is to make it impossible to reconstruct an individual data sequence's items or their trend without large error, while at the same time allowing to estimate the community data distribution and the average community data trend with high accuracy [25]. The *MetroSense* project is addressing privacy issues by "providing sensing device custodians with a notion of anonymity through k-anonymous tasking" [11, 15].

2.2 Information Visualization

The research area of *Information Visualization* can be traced back to its origins long before computers were commonplace. Early visual representations of quantitative data were used in different scientific fields and subsequently started to appear in printed publications. Many basic types of graphs, such as line graphs and bar charts, were invented by the social scientist William Playfair in the late 18th century. In 1983, Edward Tufte broke new ground by outlining key principles for effective visual representations of quantitative information. He identified common mistakes that distort or obfuscate the underlying data and voiced an appeal for clear and honest design [73]. When computers with graphics capabilities became affordable, the field experienced a resurgence of interest in the academic world [22].

In their seminal book, Card et al. defined *Information Visualization* as “the use of computer-supported, interactive, visual representations of abstract data to amplify cognition” [12]. The basic concept was that visual representations can exploit human perceptual skills to increase the bandwidth for processing information. Traditionally, *Information Visualization* has focused on the sciences, because they generate large amounts of structured and often numerical data. Visualizations allow researchers to explore and analyze this data, as well as communicate their findings to others [30].

The commercial potential of *Information Visualization* was first realized in the area of business information systems. The growing size and complexity of such systems generated a need for tools that allowed corporate executives to monitor and analyze the collected data. In an effort to meet this demand, various commercial visualization products emerged, often originating from work that began as academic research. Over time, these products became well established among the business community. The most prevalent type of business intelligence software is based on the metaphor of the vehicle dashboard [22, 44].

Recently, researchers have started to explore new application areas outside the scope of conventional *Information Visualization*. One of these areas is *Ambient Visualization*, which investigates visual representations that “reside in the environment of the user rather than on the screen of a desktop computer” [67]. In many cases, these visualizations are designed for public spaces and therefore have to consider aesthetic appeal and limit distraction. This corresponds with Weiser’s vision for *Ubiquitous Computing*, as technology becomes embedded in the human environment and moves from the center to the periphery of the user’s attention. Rather than trying to amplify cognition, the goal of these visualizations is to communicate information while minimizing the cognitive load on the user [60, 67].

Another noteworthy application area for visualizations is *Persuasive Technology*. In the context of computing, this term refers to “computing devices designed to change human attitudes and behaviors” [36]. Such technologies have found their way into numerous domains, such as marketing, health, safety and environmental conservation. Visualizations hold a lot of potential for these technologies, because they are an effective method for delivering feedback, which is an important factor for motivating behavior change. Often, these visualizations employ iconic or metaphorical representations. In general, such approaches are preferable, because they can evoke emotional attachment [34, 36].

Supporting Different Levels of Skill

Due to its academic origins, *Information Visualization* is still predominantly viewed as a tool for experts in the scientific domain. However, the availability of hardware capable of generating graphically rich interfaces has opened up the field towards a greater audience. The general public is already exposed to advanced visualizations in contemporary online media. If these interfaces are well designed, they can be highly engaging, because of their aesthetic appeal and playful nature. In turn, this leads to growing mainstream interest, as more and more users are becoming familiar with these tools and techniques [30].

It is likely that our limited screen real estate will become a fought-over territory in the quest to deliver information which helps us fulfill our daily needs. Whether it be in our personal or professional lives, we will require effective ways of accessing the data and services that our networked systems provide. Therefore, visualizations are going to play an important role, enabling us “to monitor the proverbial 7 ± 2 key indicators of our lives” [44].

In order to provide everyday users with the ability to construct and compose visualizations of their own data, it is important to gain a better understanding of their goals and motivations. Heer et al. [30] make a distinction between three different kinds of data, based on their target audience. First, *scientific data* is relevant to the traditional audience of specialists, who explore and analyze this data to gain new insights. Second, casual users may be interested in visualizing *personal data*, which is more likely to be an act of self-expression or a catalyst for storytelling. Third, *community data* might be relevant to a broad group of like-minded individuals and serve as a basis for discussion within that community. The success of visualization approaches will likely depend on how well they are tailored towards the intended audience.

However, there are major challenges in bringing *Information Visualization* to a wide variety of users. Novices cannot be expected to perform the tasks that experts routinely do during the design and implementation of visualizations. The process often involves writing database queries and converting data between various formats. Further, many visualization toolkits require a basic programming expertise. Clearly, such a level of involvement cannot be expected from lay users. In order to support these users, researchers have to provide them with tools that "make it easy to create and deploy visualizations of their datasets" [30].

More importantly still, inexperienced users need guidance when choosing visual mappings for their visualizations. The right choice of mappings has long been one of the main research interests in *Information Visualization*. Failure to prevent users from making mistakes in this area would result in poor or even non-sensical visualizations. This would have negative consequences for the perceived usefulness of visualizations in general. Therefore, providing reasonable defaults and predefined templates is key. New advances in research are necessary to create systems capable of supporting novice users and evaluating designs [20, 30].

Novice users are often unaware of the different options for visualizing their datasets. As a result, their approach is not targeted at a specific goal. Instead, they tend to gradually refine and change their designs. Thus, a user-friendly interface should provide continuous visual feedback on the outcome of user choices. Further, the system should ensure that lay users feel confident to experiment with different configurations. In order to encourage such experimentation, it is necessary to include undo functionality, making all user actions reversible. Finally, animations are helpful in letting users track changes during configuration as well as exploration [20, 26].

Finally, due to the focus on scientific analysis, there is a lack of research investigating how *Information Visualization* integrates with the set of information tools people already use today. It is not only necessary to provide lay users with effective visualizations, but it is also important to understand how these visualizations fit into an information ecosystem that helps users make informed decisions [3].

Designing Interfaces for Customization

Historically, many information visualization tools were custom-built with the goal of supporting specific tasks. In particular, many commercial business intelligence solutions were based on the assumption that their configuration would remain constant from an end-user perspective. In order to tailor such solutions to the needs of end-users, multiple actors had to be involved. Dashboard designers implemented the specifications, which were worked out by business analysts based on the feedback from end-users. This process generated a lot of overhead for communication and coordination [20, 46].

In general, it is not possible to anticipate what selection of visualizations will be appropriate for all potential users and how these visualizations should be coordinated. Hence, many research projects in *Information Visualization* aim to provide simple and intuitive interfaces for creating and deploying visualizations. These tools are built to address the visualization needs of users that come from different academic and professional backgrounds. These users are often *Information Visualization* novices, who cannot rely on experts or invest time to train themselves. Therefore, it is necessary develop solutions that enable users to mix and match visualizations without requiring programming expertise [54].

North and Shneiderman [54] classify systems for *Information Visualization* according to their flexibility with regard to three different categories. First, systems can be flexible in terms of *data*, meaning that users can provide their own datasets. Second, systems can offer freedom in the choice of *views*, allowing users select from different visual representations. Finally, the third category is *coordinations*, referring to the ways in which user interaction within one view affects the others and vice versa. Based on these aspects, the authors differentiate between four levels. Level zero offers no flexibility, while each successive level provides flexibility in one additional category as well as the preceding ones. While flexible systems are desirable, Heer et al. point out that there is an apparent fundamental tradeoff between flexibility and accessibility, observing that "increased expressiveness necessitates greater expertise when it comes to data manipulation and visual representation" [30].

According to the previous classification, the *Snap-Together Visualization* [54] system belongs to level three, allowing users to modify *data*, *views* and *coordinations*. It achieves this flexibility by implementing a conceptual model that is based on a relational databases. In this model, *views* correspond to relations and *coordinations* are equivalent to joins. The native application provides an interface for defining such relational schemata and exploring the resulting coordinated visualizations. Individual visualizations can read *data* from files, memory or relational databases. The system was evaluated by a group of statisticians and programmers. After a short training, these data savvy users were able to construct their own coordinated interfaces. Based on their analysis, the authors expressed confidence that "customized information visualization is within the grasp of novice users" [54].

Another attempt to create an accessible *Information Visualization* system is *Dashiki* [46]. It aims to enable the creation of wiki-based dashboards that integrate well with the web information ecosystem. Inspired by wikis, the system consists out of pages that are specified as user-editable markup. The design features low-barrier mechanisms for modifying the content and layout of pages and connecting visualizations to data sources. Data can be directly embedded into these pages or downloaded from an external URL. A WYSIWYG editor is provided to adjust the style and appearance of visualizations. The resulting pages are published on the web and can be embedded into other websites. Selection is automatically coordinated by highlighting the affected item in all visualizations on a page. However, it is not possible for users to specify other *coordinations*, therefore this system would be classified as level two according to North and Shneiderman, as it is only flexible in *data* and *views*.

When developing the *Exploration Views* [20] system, Elias and Bezerianos thoroughly researched and implemented existing guidelines for novice user interaction. The interface is realized as a web client based on Adobe Flex. It makes extensive use of visual templates to encourage low-cost experimentation. Selection of visualizations and templates is supported by icons and thumbnails. Further, the system assists users throughout the dashboard creation process by providing continuous visual feedback. The interface supports drag-and-drop for arranging *views* and the resulting layout adapts intelligently when users toggle the visibility of components or resize them. Users can choose from different sources to load *data* into an in-memory database that provides quick access. As in *Dashiki*, selection is automatically coordinated between the visualizations, placing the system on level two according to North and Shneiderman.

Supporting Extensibility

The creation of entirely new visualizations is a sophisticated task that usually requires programming expertise in order to generate *views*, connect them to *data* and enable *coordinations* with other *views*. Various toolkits support developers by simplifying visualization specific tasks and offering basic functionality in areas such as data management, computer graphics and interaction. Most *Information Visualization* systems provide a collection of *widgets* that encapsulate visualizations into monolithic units. Due to the complexity involved in creating new visualizations, users are generally limited to the suite of *widgets* bundled with the systems. However, many systems offer ways for developers to extend these collections [8, 20].

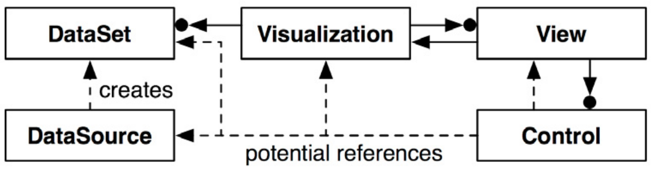


Figure 2.1: The *Reference Model* pattern by Heer and Agrawala [29].

The critical factor for an extensible and reusable software architecture, as pointed out by Heer and Agrawala [29], is finding the right separation of concerns. A common approach to achieve such a separation is based on the *Information Visualization* reference model described by Card et al. [12]. This model breaks up the visualization process into a series of discrete steps, from processing raw data to generating visual abstractions to displaying interactive views. The *Reference Model* pattern divides the system into components (see figure 2.1) supporting the individual steps. This pattern should be regarded as a high-level template for application design. It allows data sets to be used in multiple visualizations and visualizations to be displayed in multiple views.

The `Visualization`, `View` and `Control` classes in the *Reference Model* are analogous to the elements in the standard *Model-View-Controller* pattern of user interface development. As in user interface toolkits, many visualization frameworks bundle `View` and `Controller` functionality into components named *widgets*. Thus, the most common approach for extensibility is to allow developers to create new components from scratch or subclass existing ones. Typically, *widgets* include a `paint` method that is responsible for drawing the visual representation onto the screen. Subclasses can override this method to change the appearance of the *widget*. Most visualization frameworks supporting this form of extensibility require that visualizations are compiled into the applications [20, 29].

The *Snap-Together Visualization* [54] system offers extensibility through a simple API. Visualizations can be entirely independent software programs as long as they provide the methods required by the *Snap API*. This interface was designed with simplicity in mind to make it easy for researchers and developers to extend the system. It defines how the main application communicates with visualizations. The required methods allow the application to load data into the visualizations and coordinate actions such as selection. Other *Information Visualization* systems rely on similar plugin mechanisms for extensibility.

The *Rivet* [7] visualization system follows a different approach to allow the creation of new visualizations. A wrapper generator is used to expose the *Rivet API* to a range of scripting languages. The generator automatically creates the boilerplate code that is necessary to call native functions from these languages. This allows developers to implement new visualizations by writing script code that utilizes the native API. The main application can load these scripts and execute them using an embedded interpreter. A potential downside of this method is the performance cost of interpreting code, which can become a bottleneck for expensive computations.

The *WebCharts Framework* [23] takes the previous approach one step further, relying on an embeddable web browser component to display its visualizations. Rather than implementing a complete *Information Visualization* system, the authors describe a solution that can be plugged in to a variety of host applications. The proposed architecture consists out of two components that handle the communication between the host application and the visualizations – namely *HostLib* and *ChartLib*. The former becomes part of the host application, whereas the latter runs in the embedded web browser. A well-defined *JavaScript* interface is used to exchange data and events. This approach is very dynamic, as it allows new visualizations to be deployed across the web. The authors have tested their concept by integrating *WebCharts* into *Microsoft Excel*, allowing users to create visualizations based on their spreadsheet data.

Motivating Behavior Changes

The basic premise of using *Information Visualization* to display data related to resource consumption is that consumers are lacking information. Making this information available in an intuitive manner allows users to conserve resources and make informed decisions. Feedback helps to expose consumption which was previously invisible, which is often the case for energy use. Most commercially available feedback displays rely on an intrinsic rational-economic model, which assumes that people will be motivated by the prospect of saving money [43, 48].

In order for feedback to be effective, it is important to carefully consider what to present and how. As illustrated in the wisdom hierarchy, *data* by itself is necessary but not sufficient. Instead, it is crucial to provide significant patterns of *data*, i.e. *information*. Feedback becomes even more valuable if it shows significant patterns of *information* with action plans, i.e. *knowledge*. Studies have demonstrated that it is easier to persuade users by addressing specific behaviors rather than general ones. Therefore, it is important to give actionable feedback, highlighting the necessary steps to reach the desired goal. In the case of energy conservation, this requires systems that are capable of analyzing consumption and pointing out potential savings [44, 61, 69].

However, researchers increasingly recognize that the mere availability of information is not sufficient to affect sustained behavior changes. Nisi et al. observe that many users loose interest in monitoring their energy consumption within weeks – and even with the introduction of new visualizations “the novelty effect only [lasts] for less than a week” [53]. Financial incentives can be effective, but many users relapse into old behaviors if the rewards are small in comparison to income and the novelty wears off. A common criticism is that the rational-economic model is not well suited to accurately represent how and why people consume. Strengers illustrates the issue as follows:

> “It is unlikely that most of us, on rising from our slumbers each morning, approach every task ‘rationally’ by consciously weighing up the costs and benefits of a shower, or ensuring we undertake the most efficient load of laundry.” [71]

The author further elaborates that the activities of everyday life are much more guided by social norms, cultural dynamics, institutional rules and technological means. If the consumer does not already hold a strong conviction to conserve energy, then feedback only informs without motivating action [28, 48, 71].

In order to change behavior, *Persuasive Technologies* have to consider personal attitudes and interests. Extrinsic forms of motivation can act as triggers to foster intrinsic motivations. Further, it is possible to differentiate between several “stages of readiness, willingness and ableness to change” [28]. He et al. draw upon the stages of behavior change identified by the *Transtheoretical Model* to develop targeted motivational strategies. Because users have different motivations – and these motivations change over time – the authors assert that it is not feasible to develop a ‘one-size-fits-all’ solution [28, 48].

In their evaluation of *StepGreen.org* – a website designed to encourage energy saving behaviors – Mankoff et al. acknowledge that “no one visualization fits all users and contexts” [43]. The authors plan to address these varying needs by developing a more flexible and adaptive solution. A common approach for achieving this goal is to support end-user customization, allowing users to personalize their interface. Interaction with the application invokes self-reflection and elicits a sense of freedom, which can increase intrinsic motivation. Further, a personalized interface is a more effective motivator than one that displays general information [28, 43, 77].

There is great potential for *Information Visualization* to engage people with their resource consumption when visualizations act as the basis for social interactions. The *Wattsup* application allows users to visualize and share energy consumption data on Facebook. In a study of the application, Foster et al. observe that the socially enabled condition is “more enjoyable and more effective than individual monitoring” [24]. The users engaged in banter and competition, which proved to be very motivating. Further, the social aspect allowed for collaboration and engagement, wherein users gave tips and discussed the success of their actions to reduce energy consumption. A study by Moere et al. on exposing energy consumption on house facades also echoes the beneficial effects of making private information publicly available, such as peer pressure and healthy competition. However, the design of visualizations for these applications requires special care, as users question the fairness of comparisons and voice privacy concerns [24, 48].

2.3 Human-Computer Interaction

The user interface lies at the heart of *Human-Computer Interaction*. The research area studies different aspects of the communication between humans and computers. This requires an understanding of both parties involved, which is supported by a multidisciplinary approach incorporating diverse research fields ranging from behavioral sciences to computer science. In particular, *Human-Computer Interaction* is concerned "with the design, evaluation and implementation of interactive computing systems for human use" [31]. Further, it observes and analyzes phenomena surrounding the interaction [31].

The means by which humans interact with computers are continuously changing. Over time, old technologies become obsolete and new interaction paradigms emerge. For example, the availability of computer graphics capabilities has paved the way for graphical user interfaces that employ novel metaphors. The current state of technology is rapidly evolving. Already today, developers are facing the challenges of designing interfaces for ever-smaller, portable devices with new input technologies. Our interaction with computers will undergo further transformation as we move closer to the vision of *Ubiquitous Computing*.

A recent disruptive development is the adoption of touch-screen based interfaces by mainstream devices such as smartphones or tablet computers. The miniaturization of hardware together with improvements in efficiency and battery life have spawned a new generation of portable devices. Touch screens are common among these mobile computers, because they can save valuable space by eliminating physical input devices such as keyboards. From the early beginnings, the development of touch interfaces was accompanied and guided by *Human-Computer Interaction* research. In practice, the rapid pace of adoption continues to present challenges and foster innovation in user interface design [51].

Touch-screen based input provides a foundation for interfaces that are flexible and tailored to specific tasks. Even more importantly, it opens the door for interactive applications that are easier to use and more intuitive than those relying on traditional input methods [51]. To a large extent, these positive attributes can be explained by a heightened sense of direct manipulation. The core principles of direct manipulation were established by Shneiderman. First, these interfaces feature continuous representation of the objects and actions of interest. Second, operations are triggered using physical actions and graphical interface elements rather than issuing commands with complex syntax. Third, operations are rapid, incremental and reversible with immediate feedback on the objects of interest [65]. Although the concept of direct manipulation is not exclusive to touch interfaces, they can convey a better sense of control. This results from allowing users to manipulate objects on the screen without peripheral input devices.

Direct manipulation is especially well suited for novice users. Since the objects and actions are continuously present, beginners can learn the basic functionality quickly and easily. Due to the incremental operations with immediate feedback, users can quickly see whether their actions are adequate for reaching their goals. Additionally, reversible operations encourage experimentation and allow users to explore the functionality of the system without negative consequences. Finally, these kinds of interfaces rarely require error messages. Overall, users experience less anxiety, because their actions are fail-safe and the effects are clearly visible and comprehensible [65].

When working with portable computers, it is important to recognize their limitations and potential pitfalls. Rather than being universal tools, these devices often compromise on functionality in favor of supporting mobility. Despite the strengths of touch interfaces, developers need to be aware of their weaknesses. Precise selection can be difficult, especially on devices operated with a finger rather than a stylus. Furthermore, parts of the screen are often obstructed by the pointing object. When comparing tablet and laptop computers, Ozok et al. found that users preferred the tablet device for tasks that involve reading or direct manipulation. However, it performed worse in tasks that require a lot of cursor targeting and text entry, such as form filling. As a result, users were hesitant to adopt the touch-screen based device for all their daily computing needs. Nevertheless, most users appreciated the mobility of the tablet computer and noted that it was fun to use [51, 57].

An important aspect of *Human-Computer Interaction* is the study of technical artifacts as part of the user experience within a given context. McClard and Somers use ethnographic and usability methods to observe how tablet devices become integrated into household activities. Although the methodology differs from the task performance evaluation by Ozok et al., both arrive at similar conclusions. The study confirms that tablet computers provide a qualitatively different user experience than desktop computers. The authors acknowledge certain flaws regarding the form and function of tablet devices, especially with regard to text entry. Nonetheless, participants viewed tablet computers as ‘relaxing’, ‘fun’ and well suited for ‘casual’ use, while desktop computers were considered better for ‘work’ and ‘serious’ activities. In particular, users appreciated the portability of tablet devices, using them in places where they did not have access to a desktop computer, often in comfortable positions. The authors further expand on this, concluding that users value mobility, because it supports multi-tasking, allowing them to engage in other household activities while using the tablet device [45].

In order to design successful interventions, it is necessary to understand the human side of *Human-Computer Interaction*. New portable devices are fundamentally shaping our daily activities and media consumption habits. The conventional view of the relationship between traditional and new media is based on competition-based displacement theory. However, observations have shown that this theory is not an accurate description of the way users consume media. A more realistic model is proposed by media complementarity theory. Rather than superseding traditional with new media when seeking out information concerning a particular area of interest, highly motivated users utilize multiple outlets across many media types to gather information in that specific area. The particular medium chosen to retrieve content depends on what is available and convenient [19].

As outlined earlier, the main merit tablet computers lies in their portability, making them convenient to use and well-suited for multi-tasking. Considering the theories proposed by media scholars, it seems probable that tablet devices will occupy a prominent place in the media landscape. Overall media usage is likely to grow with complementary and simultaneous consumption of content targeting specific interests. The observations by McClard and Somers confirm this development, finding that households engage in extensive television viewing while using the tablet device [45]. Some researchers already see modern society moving past the *Information Age* towards an *Age of Interruption*, where we are “afflicted with chronic multi-tasking and continuous partial attention induced by cell phones, email, the Internet, handhelds, and our other many devices” [56].

The development of new interfaces should be guided by human-centered design, acknowledging natural user behaviors and the surrounding context. In particular, future systems should integrate well with our media consumption habits and the complex information landscape. Oviatt points out that limited attention and working memory capacity are increasingly recognized as major bottlenecks for human information processing. Therefore, the author proposes a set of recommendations for the design of human-centered interfaces. Effective systems should accommodate existing work practices and support flexible approaches for problem solving, allowing users to work with familiar representations. Further, these systems should minimize complexity and avoid unnecessary interface features, interruptions or distractions [56]. In conclusion, it is important that future interfaces "minimize cognitive load so users can focus on the intrinsic difficulty of performing well on their real-world tasks" [56].

Researchers like Dourish argue that traditional *Human-Computer Interaction* methods fail to capture important aspects of *Persuasive Technologies*. They suggest that the focus of attention should broaden beyond the user interface, in order to cover the full extent of human interaction with technological interventions. In particular, it should include "the cultural and social processes by which [they come] to be embedded in everyday life" [18]. Dourish acknowledges existing approaches that are modeled around the concepts of 'individual behavior change' and 'citizen science'. He elaborates further on the potential of technologies that facilitate 'political and environmental mobilization'. Such technologies can highlight common concerns and link users "into a broader coalition of concerned citizens, social groups, and organizations" [18].

2.4 The Use Case of Energy Monitoring

The desire to use energy efficiently has already entered the consciousness of early adopters. To get a complete picture of the field, it is necessary to examine commercial technologies for monitoring energy consumption. As *smart meters* have yet to become widespread, consumers interested in tracking their energy use turn to *energy monitors*. Most energy monitoring solutions consist out of two main components, the sensing hardware and an information display.

Two different approaches have emerged for collecting measurements. The first option tracks the energy consumption of the entire household. This is accomplished by a sensor attached to the main electrical line, often in the form of a clamp around the cable. Usually, the installation is performed at the switchboard, which can be on the outside or in the garage. In order to make the information easily accessible, these solutions transmit their measurements wirelessly to a base station inside the house. This type of energy monitor delivers an aggregated measurement, which is suitable to answer pressing concerns such as electricity costs and environmental impact. However, it is often difficult to determine the contribution of individual appliances to the total household consumption. The second option addresses this problem by sensing energy use at the power outlets, allowing users to monitor individual appliances. Generally, these solutions are realized as plug adapters and therefore have a direct connection to the electrical system. They can have integrated displays or wireless transmitters for reporting measurements as in the first case. The disaggregated information can help consumers to better understand where energy is consumed within the household.

Similarly, there are two major ways of presenting the energy consumption data. On the one hand, the display can be realized as dedicated, stand-alone hardware. Often, this display is integrated in the base station that stores and processes measurements from the sensing hardware. Ideally, the device is placed in a central location in the household. These solutions often feature simple, black-and-white LCD displays that show a fixed set of performance indicators. This makes it feasible to leave them always-on, allowing users to glance at their energy consumption whenever they pass by the device. On the other hand, it is increasingly common for energy monitoring solutions to include software for accessing the information. This has the advantage of providing much greater flexibility for how the information is presented. It also introduces the possibility to perform much more elaborate analysis of the data. In the optimal case, these solutions have functions that allow users to export their data and share it with third party applications.

The following sections will briefly introduce concrete examples of hardware and software for energy monitoring. The review will focus on web-enabled solutions, as are particularly relevant in the context of this work.

Sensing Hardware for Energy Monitoring

Sensors for energy monitors can be designed in various ways. A simple and cost-effective solution can be realized based on the Hall effect. If an electric current flows in a conductor, it produces a magnetic field that is proportional to the magnitude of the flow. A Hall effect sensor varies its output voltage depending on the strength of the magnetic field. Due to the physical nature of the Hall effect, measurements taken by the sensor are susceptible to interference from stray magnetic fields. The quality of the measurements can greatly be improved by placing a magnetic circuit around the conductor which concentrates the magnetic flux through the sensor. In the case of an energy monitor, it is convenient to integrate this magnetic circuit into the clamp [59].

The output voltage of the Hall effect sensor is sampled periodically. These samples only represent the sensed quantity of electric current in the conductor. In order to calculate instantaneous power, the energy monitor needs to know the nominal supply voltage in the respective country. Usually, devices are pre-programmed out of the box, which can cause problems if energy monitors are imported. If the supply voltage is known, instantaneous power is given by the simple formula $P = V * I$. The accuracy of the calculated total usage largely depends on the sampling frequency. In order to reduce fluctuations, multiple measurements can be performed and averaged over a short period of time. Even though the results are good indicators for household energy consumption, they are not exact. For this reason, *energy monitors* are not suited for billing purposes, which is one of the key differences to *smart meters*.

After the sensor completes a measurement, it is transmitted to the base station for further processing and storage. Various possibilities exist for enabling the communication between the sensor and the base station. Wireless communication via radio frequency transmitters is arguably the most user friendly solution. Many energy monitor manufacturers implement the *ZigBee* standard for low-power radio networking.

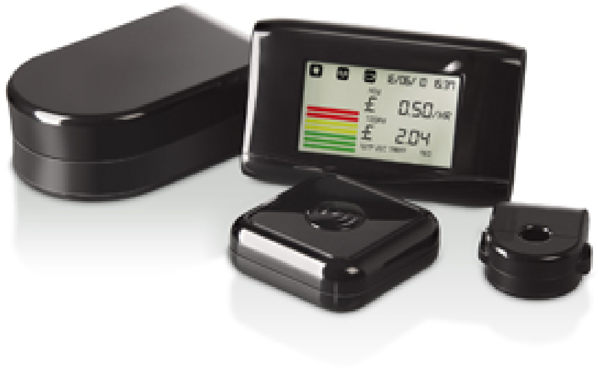


Figure 2.2: AlertMe energy monitoring hardware.

AlertMe[2] is a company offering home monitoring solutions specializing in energy usage and security. Their products are available in the United Kingdom. Their product portfolio features an exhaustive array of energy monitoring hardware (see figure 2.2). The *SmartMeter Reader* attaches to the main electrical line with a clamp, while the *SmartPlug* adapter allows monitoring appliances at individual power plugs. All products communicate with a central *SmartHub* using *ZigBee*. Users can view their energy consumption on a dedicated display or upload data to a proprietary web service, which will be described in detail in the next section.

[2]http://www.alertme.com

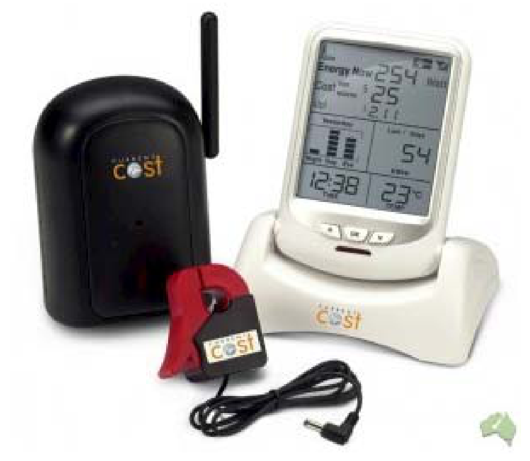


Figure 2.3: Current Cost energy monitoring hardware.

Current Cost[3] is a company focusing purely on energy monitoring. They are also based in the United Kingdom, but they have acquired several international partners for the distribution of their products. In Australia their devices can be purchased through *SmartNow*[4]. Their *Envi* and *EnviR* energy monitors (see figure 2.3) receive data from a sensor at the main electrical line. There is an option to attach a second clamp to monitor photovoltaic solar production separately. Further, the company also offers power socket sensors, which are referred to as *Individual Appliance Monitors*. The communication among devices is wireless and based on *ZigBee*. The system uses *Cosm* as a backend for hosting sensor data online, making it an interesting solution for third party developers.

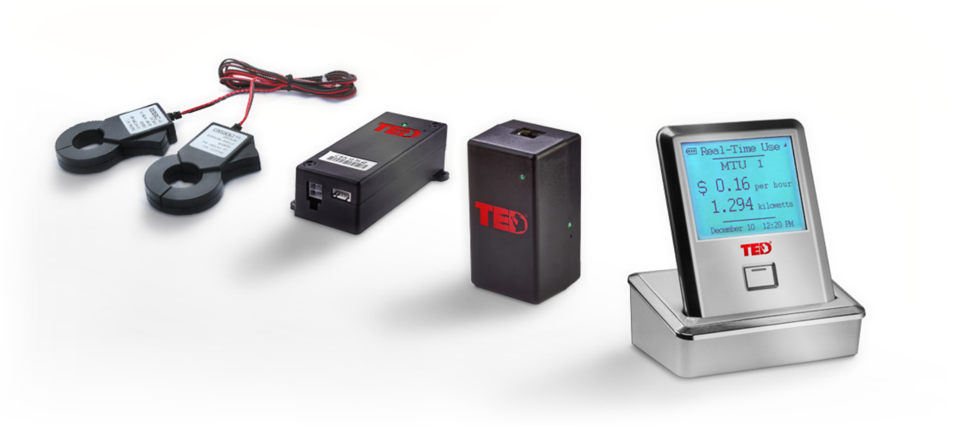


Figure 2.4: The Energy Detective energy monitoring hardware.

The Energy Detective[5] is an American brand of energy monitoring hardware produced by *Energy Inc*. The complete package contains the familiar core components: a sensor featuring clamps for the main electrical line, a base station that acts as a gateway and a dedicated display (see figure 2.4). There is no adapter for monitoring appliances at individual power plugs. As in the previous solutions, the components communicate using *ZigBee*. *The Energy Detective* is ahead of other companies in terms of third party support, allowing developers to access the energy consumption data through the *TED Footprints API*.

[3]http://www.currentcost.com
[4]http://www.smartnow.com.au
[5]http://www.theenergydetective.com

Web Applications for Energy Monitoring

Web applications can add substantial value to an energy monitoring solution. The key prerequisite for using such an application with an energy monitoring setup is having an always-on Internet connection. Further, the energy monitor needs to be connected to the Internet and support an appropriate communication protocol for the desired web application. Generally, a list of supported devices can be found on the respective websites. It is not uncommon for web applications to support multiple devices in order to acquire a greater user base. The functionality that such web applications provide can be summarized as follows.

Tracking The web application is capable of communicating with the energy monitor and receiving its current status. Simple energy monitors only provide a single measurement, whereas more complex setups report much more fine grained data, such as energy usage on a per-appliance level. The timeliness of the data is another differentiating factor and can be limited by both, the energy monitor and the web application. Most web applications support this communication through a HTTP-based and RESTful API.

Aggregating The web application collects and stores a history of all tracked measurements. This history can vary in granularity, depending on the interval at which readings are taken. These intervals are often limited by the web application in order to manage the load on the servers. One technique for collecting fine grained data without causing constant traffic is batch updating. In that case, the energy monitor collects readings over a fixed amount of time and transfers them all at once. Having access to this historical data allows the user gain additional insights about his energy consumption. It exposes usage patterns and makes it easy to track and compare the effects of energy conservation efforts.

Hosting The web application can make the aggregated data available to other parties. In general the user providing the data is also the primary recipient. Through the web application, the user can access their data from virtually any web browser. That enables the user to view their data through an unobtrusive and intuitive interface. Displaying on a computer allows for aesthetically pleasing and elaborate visual representations of the data. The web application can also provide access to the hosted data for other parties. It is becoming increasingly common for web applications to share data about energy consumption in a social context, for example within a selective group of friends or even with the broad public. The resulting competition can be an important incentive to reduce energy usage. Besides that, the data is also valuable for commercial partners that provide energy services tailored to the individual users. Finally, web applications may choose to offer an API for third party developers and therefore act as information brokers.

Analyzing The web application performs higher level analysis on the data it has tracked and aggregated. The goal is to enrich the data and present the user with meaningful and actionable information. The sophistication of the analysis can range from simple conversions between different metrics to complex pattern matching resulting in detailed breakdowns and recommendations. One approach for motivating energy conservation is to provide increasingly ambitious goals for the user and present the data in terms of progress towards these goals. Some web applications are also capable of sending out notifications in the form of emails or text messages based on changes or trends within the data.

As energy monitoring is increasingly gaining traction, several web applications have been developed. The offerings can vary quite substantially with each service trying to provide unique value propositions. This section gives an overview over a number of notable web applications. The only thing that all the following web applications have in common is that they track, aggregate and host the user’s energy consumption data. The following should not be regarded as a complete listing of web applications in this space, but rather as a selection of the most promising or unique ones. In many cases the web applications are offered by the companies that are producing the energy monitoring equipment. Rather than being standalone products they are essentially ways of adding value to the hardware.

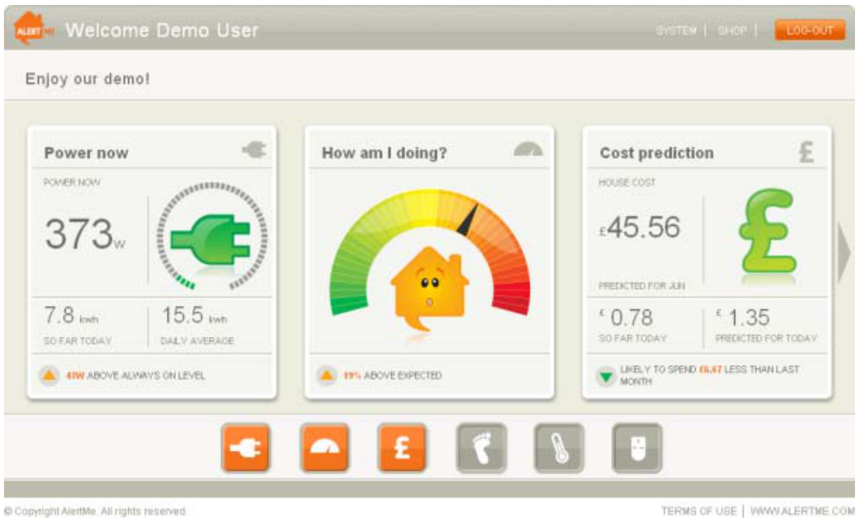


Figure 2.5: AlertMe web interface.

AlertMe offers an online dashboard (see figure 2.5) that works exclusively with their proprietary energy monitoring devices. It features a fixed number of predefined widgets that display the user’s energy consumption using different metrics, such as power drain, cost and carbon footprint. The costs are extrapolated to give a monthly cost prediction. A swing-o-meter widget provides an indicator for the user’s performance compared to other households. Further widgets display temperature readings and individual appliance statuses. By clicking on a widget of interest the user can access more detailed information and historical data. Users can access the dashboard on the go using the mobile version of the website or a native *Apple iPhone* application.

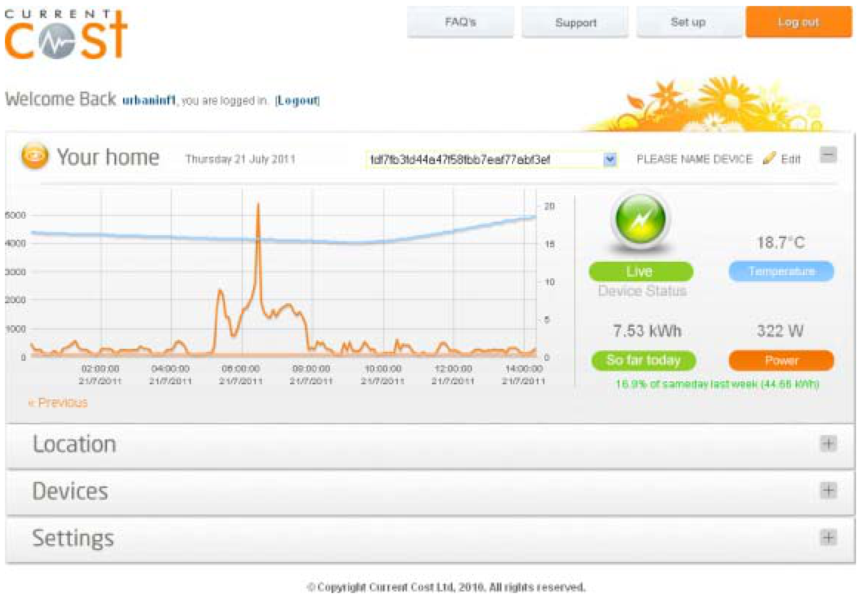


Figure 2.6: Current Cost web interface.

Current Cost provides a web application (see figure 2.6) that only supports *Current Cost* energy monitors. The web portal displays a line chart of the measurements taken by the devices throughout the day, generally power drain and temperature. The latest devices support multiple data streams (IAM – individual appliance monitoring, see above) which can be displayed separately within the interface. Furthermore, the dashboard features a visual representation of the current device status as well as a comparison with the usage on the same day one week ago. A mobile version of the website is also provided.

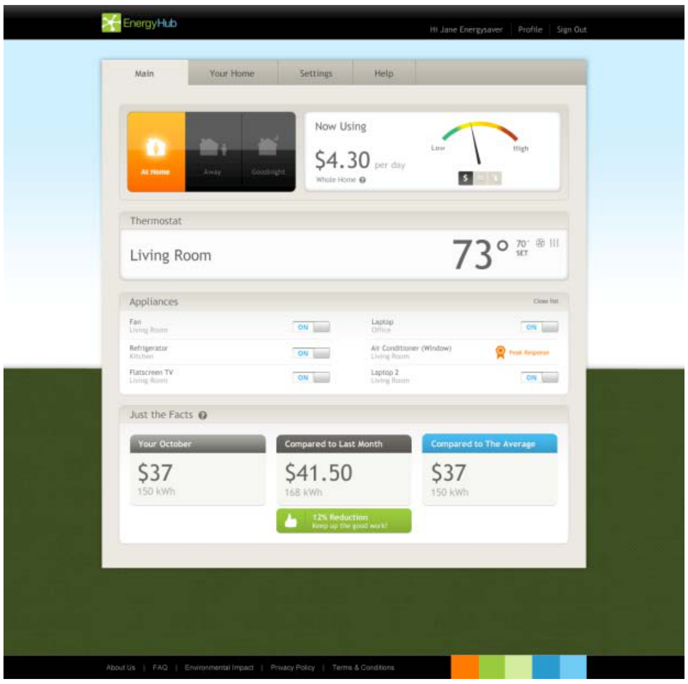


Figure 2.7: EnergyHub web interface.

EnergyHub takes a different approach by catering to both, consumers and utility companies. They are conducting their business in the United States. For tracking energy consumption they access data from smart meters provided by the utility companies. Their *MyHub* web portal (see figure 2.7) can be used with smart meters based on the *Itron OpenWay* architecture. It is limited to a fixed set of widgets. The energy usage can be viewed in terms of power drain, cost or carbon footprint. A swing-o-meter is used to give an indicator of the user's performance. Other elements display temperature readings, individual appliance statuses and monthly cost predictions. For mobile access *EnergyHub* provides a native *Apple iPhone* application.

Wattvision is a software and hardware solution for energy monitoring developed by *Aerodyno Inc*. They are a start-up company based in the United States. They differ from the previously mentioned solutions in that they also offer an open API for communicating with their web application. This means that the website cannot only be used with their device, but also with other energy monitors such as *The Energy Detective*. The dashboard displays a line chart of measurements taken by the sensor. Additionally the current energy usage is displayed in terms of power drain and cost. There are daily cost projections and comparisons with users on the same street as well as all users of the service. The *Wattvision* website also features a ranking of all participating households. The service can be accessed on the go through a mobile version of the website.

Due to increasing public awareness and political backing, environmental sustainability and green technologies are viewed as business areas with considerable future potential. As a result, web applications are also offered by several companies that are not directly involved in the production of energy monitoring equipment but rather focus primarily on web services.

Google, a leading provider of numerous web services, was one of the first to enter the energy monitoring field with *Google PowerMeter*. The project originated from the philanthropic branch of the company. *Google* partnered with utility companies as well as device manufacturers to gain access to energy consumption data. Energy monitors from companies such as *AlertMe*, *Current Cost* or *The Energy Detective* were officially supported and many others were compatible with the service. The web interface displayed a fine grained bar chart of readings taken by the device throughout a day, week or month. Yearly cost predictions were extrapolated from the selected time span. Further, the user could set a target budget and perform comparisons with his past usage. Although the project has now been discontinued, it was influential by providing visibility and validity to the business of home energy monitoring and nurturing several companies that remain active in this field.

Cosm sets itself apart from the other services, because it aims to be a general purpose platform for managing and sharing real-time data from sensors. Nevertheless, it offers all of the basic functionality of a web application for energy monitoring. Energy consumption data can be uploaded through its open API and some energy monitoring devices – such as the latest *Current Cost* bridges – implement this right out of the box. The *Cosm* website features a repository of applications that can be used for displaying and working with the collected sensor data. These applications range from simple widgets that can be embedded into other websites to advanced plugins that perform additional processing and offer higher functions such as alerts. In addition to that, several native mobile applications provide access to the data on the go.

PlotWatt is a fairly recent service that is currently in a beta phase. Its unique value proposition lies in the capability of analyzing a single aggregated stream of energy monitor readings and deriving per-appliance statistics. As of now, the service supports monitors by *The Energy Detective* and *Wattvision* with further device support being planned down the line. The *PlotWatt* dashboard prominently features the current energy usage, the derived per-appliance breakdown and a line graph of the collected measurements over a day, week or month. The design is intentionally striped down and simple, focusing on information that helps the user in reducing his energy consumption. The site also lists recommendations for potential savings in text form that are tailored towards the user.

PVOutput is a service that allows people to cooperate in groups and share their photovoltaic energy output data. The groups are subsequently ranked on a ladder resulting in a friendly competition. Data can be submitted and retrieved through an open API and a wide range of energy monitors is supported through the *PVOutput* integration service, including those manufactured by *Current Cost* and *The Energy Detective*. The website features a number of basic visualizations such as breakdowns of the energy output by geographic locations or bar charts displaying the accumulated amounts of energy produced over time.

Energy monitoring remains an active field and various companies are experimenting with different ways of visualizing energy consumption and helping users to conserve energy. However, it is becoming increasingly clear that the mere display of such information is not enough to reach and effectively engage a significant portion of the population. Most people are already exposed to a large amount of information on a day-to-day basis and are reluctant to spend additional time and effort on energy monitoring. It will be essential to make energy data available through various communication channels in order to maximize its reach.

3 Methodology

This thesis seeks to develop a Dashboard prototype that will be deployed to end-users in the real-world. A central pillar of the concept is a flexible user interface that supports customization. Ultimately, the prototype aims to assist users with their real-world tasks, allowing them to make informed decisions. In order to stand a chance at adoption, the solution needs to be accessible and user friendly. Therefore, the project was guided by a user-centered design approach.

In accordance with the core principles of user-centered design, the development consisted out of several iterations and the results were repeatedly verified with experts and prospective users. The project made use of tools and techniques for rapid development in order to quickly deliver results and create a succession of incremental prototypes. The main iterations of the development process are summarized in figure 3.1. An important goal was to employ adequate evaluation methods at each stage of the project.

A focus group discussion was carried out early on during the design of the Dashboard prototype. Conducting the discussion when the concept was still in an early, flexible stage had significant impact on the following development. The method was chosen, because informal group discussions provide different perspectives on a problem and promote an exchange of ideas. The benefits of focus groups for qualitative research are well documented. Morgan recognizes the potential in “the explicit use of group interaction to produce data and insights that would be less accessible without the interaction found in a group” [49]. The moderator guides the discussion by exercising “mild, unobtrusive control over the group” [38]. In particular, the moderator should be prepared to interject probes and open-ended questions that invigorate the discussion and keep the group focused.

An expert review marked the end of the first round of software development, which yielded an initial software prototype. It served to verify whether the original concept was successfully realized. Expert reviews are widely used, because they can be performed rapidly and require few resources, given that experts are often available on-site within the team. The versatile inspection-based evaluation method is applicable at any stage in the development cycle. Several different variations exist for conducting expert reviews, often based on established usability guidelines and heuristics. In practice, many inspections are informal with experts performing “heuristic evaluation on the basis of their own intuition and common sense” [52]. According to Nielsen and Molich, heuristic evaluation is most effective with a small group of experts, since the number of discovered problems “grows rapidly in the interval from one to five evaluators but reaches the point of diminishing returns around the point of ten evaluators” [52]. In this project, the review took on the shape of a formal usability inspection, where “experts hold a courtroom-style meeting [...] to present the interface and to discuss its merits and weaknesses” [66].

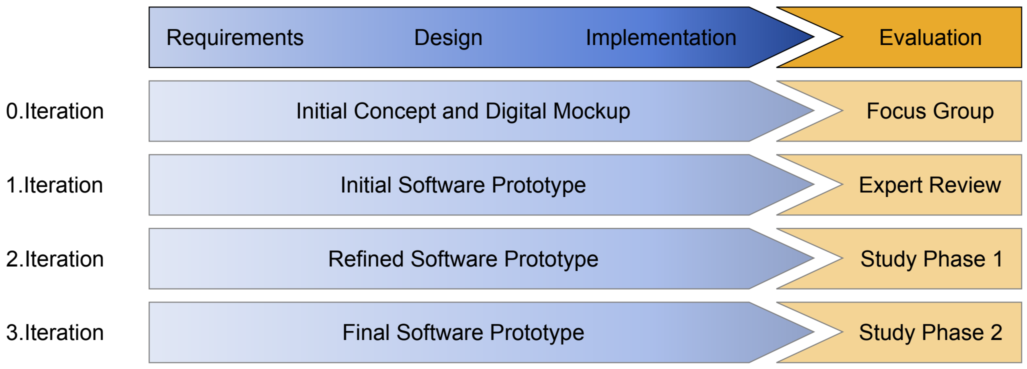


Figure 3.1: Iterations during the development.

The project culminated in an exploratory user study, which served to verify assumptions about the role of the Dashboard prototype in a real-world setting. It was carried out as a field trail, spanning two separate phases. During each phase of the study, the latest version of the prototype was deployed in local households for extended periods of time, lasting up to four weeks. Between the two phases, the prototype was further refined based upon an improved understanding of the user behavior and context. Due to the unique characteristics of the project, a holistic evaluation of the user experience had to take place over time in natural environment rather than a controlled laboratory setting. This reflects Suchman's views on plans and situated actions, who contended that in order to "understand technologies ethnographically, it is required to locate artifacts within the sites and the relations of their everyday use" [72]. Her realization that people's actions are influenced by the context of a specific situation is especially relevant for the evaluation of decision support tools like the Dashboard prototype. The impact of these tools cannot be determined in isolation from other people, devices or the physical environment, because these factors have fundamental influences on a user's course of action. As a result, it was not feasible to implement a randomized experimental design. Instead, the study relied on a mix of observational and quasi-experimental methods to estimate the role of the Dashboard prototype with regard to energy monitoring. Although such methods do not allow to make definitive conclusions about causal relationships, they are accepted as valuable tools for applied research [63]. Quantitative data was collected in the form of empirical usage statistics, tracking various user interactions with the Dashboard system. However, due to the quasi-experimental design, it was essential to supplement this data with qualitative feedback from participants. A wealth of feedback was gathered through semi-structured interviews with a comprehensive question guideline, allowing the interviewer to digress and probe deeper into specific areas of interest [5].

4 Design Process

This section describes the activities that informed the major design decisions and shaped the requirements of the Dashboard system. The development was structured in a way that would repeatedly involve users in the evaluation of the results at the end of each iteration. The main concepts of the prototype were fleshed out during the early iterations. In particular, the focus group and the expert review delivered valuable input for the overall design. In the later phases running up to the user study, the development focus shifted towards refining the design, extending the system and addressing usability issues. At the end of this section, the key learnings from the early phases of the project are summarized in the form of adapted requirements for the final prototype.

4.1 Focus Group

The main goal of the focus group was to gain a better understanding of user needs and expectations. Ultimately, the results of the discussion were analyzed to derive functional requirements for the system. In order to keep the discussion open, participants were given a broad introduction to the topic. The moderation was careful not to impose preconceptions about *Information Visualization* systems and allow participants to express their own ideas. As a backup, a digital mockup was prepared, illustrating several possible Dashboard configurations. However, since participants engaged in a fruitful discussion, it was not necessary to present this additional stimulus.

Structure

Before getting started with the focus group, participants were asked to fill out a brief questionnaire. This was done to gather basic information about the people attending the discussion and learn about their personal experience and background. In particular, participants were asked about their familiarity with commercial products in the central areas: *Energy Monitoring* and *Information Visualization*.

Once all participants had gathered in the meeting room, the moderator opened the session with a general introduction to the topic of discussion. Without going into specifics, the introduction briefly outlined technological developments that contribute to the wealth of real-time data, including data about resource consumption and environmental performance. Further, the moderator advocated the use of *Information Visualization* for communicating this data effectively. Finally, the introduction closed with the general idea of allowing users to compose and visualize information on a single display.

Subsequently, participants had the opportunity to ask questions and give comments about general concept. This led to a first round of broad discussion, which was aimed at collecting opinions and attitudes about the project. After giving the participants chance to speak freely, the session returned to a more structured course. The participants were presented with several tasks. Depending on each task, participants worked collectively or individually.

1. **Gather information that is generally relevant on a regular basis. (Collective)**
   This task served as a warm-up exercise to get an broad overview over the kinds of information that people use on a day-to-day basis. It immersed participants further in the subject and clarified the motivation behind the project. The resulting suggestions were collected on the whiteboard, keeping them visible to all participants throughout the rest of the discussion.

2. **Discuss approaches for presenting the general information. (Collective)**
   This task was aimed at getting participants to think about visual representations of the information. The main interest was to get a feeling for people's expectations with regard to the visualizations. In particular, the moderator inquired about the appropriate levels of abstraction and detail for specific kinds of information. This presented an opportunity to talk

about the role of aesthetics and the different uses for plain representations of raw low-level data versus graphically rich displays of aggregated high-level information. The discussion also touched on aspects of composing different kinds of information in a single display.

3. **Select information that is personally relevant to you. (Individual)**
The goal of this task was to create individual profiles with different information needs. The participants were asked to write down the most relevant kinds of information on separate pieces of paper. They could draw from the previously collected items on the whiteboard or add new ones if they were still missing.

4. **Sketch approaches for presenting your personal information. (Individual)**
This task encouraged participants to get creative and use the provided crafting supplies. They were asked to review the information they had collected and devise a display featuring their favored visual representations. A possible approach was to sketch the desired visualizations on the back of the pieces of paper, which had been labeled in the previous task, and arrange them. However, participants were not required to proceed this way. If they envisioned other approaches for composing the individual visualizations, they were encouraged to sketch them separately.

5. **Present and discuss selected results. (Collective)**
Once a significant portion of participants was ready, they were given a chance to voluntarily present their results. This was done to gain insights into the thought process and rationale behind the different designs. The explanations helped to illustrate some of the more complex and abstract ideas behind the sketches. Participants who were still working on their designs could finalize them in the meantime.

6. **Discuss approaches for allowing users to compose their personal display. (Collective)**
Finally, the focus group closed with a last round of discussion. Based on the experience of creating their own designs during the session, participants discussed the opportunities and challenges for allowing users to compose their personal display. The discussion included aspects such as the appropriate degree of flexibility and the associated effort required for customization.

Setting

The focus group discussion was held in a meeting room at the Urban Informatics Research Lab (see figure 4.1). The participants were seated around a table with moderator sitting at the head of the table. A white board was positioned behind the moderator, which was used for taking notes during the discussion. As part of the session, participants were encouraged to get creative and sketch some of their ideas. To support this, various crafting supplies were placed on the table, including plenty of blank paper in different sizes, colored pens, scissors and tape.

The group had a total of 8 participants, 2 female and 6 male. The participants were aged between 25 and 43 years old. They were recruited from the community surrounding the Urban Informatics Research Lab. Some participants were taking part in the overarching study about domestic energy conservation, meaning that they already had energy monitors installed at home. Furthermore, the group included PhD students from Urban Informatics Research Lab. Therefore, the group was not representative of the general population. Instead, the participants were mostly computer savvy and already had experience with energy monitoring. This was deemed to be a favorable constellation, because it was possible to draw from their experience and discuss advanced topics, such as aspects of the user interface. They were also representative of the early adopters of tools like the Dashboard prototype, since these individuals are likely to be technically adept and sensitized to the issue of energy conservation.

Figure 4.1: Impression from the focus group.

The discussion was documented with photographs and an audio recording. Before starting the recording, a disclaimer was announced to the participants. In total, the focus group lasted roughly 1 hour and 15 minutes. Out of that, approximately 20 minutes were spent on letting individual participants compose and sketch their own designs.

Results

After the general introduction to the project, participants voiced several questions and comments. A particular topic of interest was the usage context of the application. The participants deliberated whether the solution should be fixed or mobile, that is either located at a certain spot in the house or free to move around. One of the participants remarked that both use cases were valid and useful to him. On the one hand, a fixed device could be placed in a visible spot, preferably close to energy intensive appliances, such as the kitchen. On the other hand, a mobile device would have advantage that users could carry it with them while doing everyday tasks and use it in ways that were not anticipated. The group concluded that observing where and how people use the tool could be an interesting outcome of the user study. Throughout the session, participants also brought attention to other notable issues. One topic that came up was privacy concerns regarding certain information. Further, participants expressed the need for multiple profiles. These profiles could support different users as well as different usage scenarios. For example, one configuration could be optimized for tasks at work and another for leisure activities.

During the first group task, participants collected an extensive selection of different kinds of information which can be relevant on a regular basis. While some of the results were expected, whereas others were completely new and original. The broad scope and variety of ideas could be seen as confirmation for the project concept, because users clearly had a wide range of information needs. The participants voiced their ideas freely, resulting in an unstructured collection. Following the focus group, the ideas were organized into categories to provide a better overview (see table 4.1).

Category	Subcategory	Information
Sensor Data	Resource Consumption	*Energy*
		Gas
		Water
		Petrol
Government Data	Meteorology	*Weather Forecast*
		Severe Weather Warnings
	Traffic	*Traffic Jams*
		Accidents
		Parking Spots
	Public Transport	*Timetables*
		Delays
		Electronic Ticket Balance
Social Data	Communication	*Social Media Messages*
		Social Media Statuses
		Email
Business Data	Venues	*Opening Times*
		Occupation Levels
		Queues
		Product Stock
	Promotion	*Sales*
		Events
	Gambling	*Lottery Numbers*
Personal Data	Organization	*Calendar*
	Finances	*Bank Account Balance*
		Credit Card Balance
	Health/Fitness	*Exercise Statistics*
		Nutritional Statistics
	Food Supplies	*Groceries*
		Expiring Food
Media Data	Entertainment	*News*
		Trivia/Fun Facts

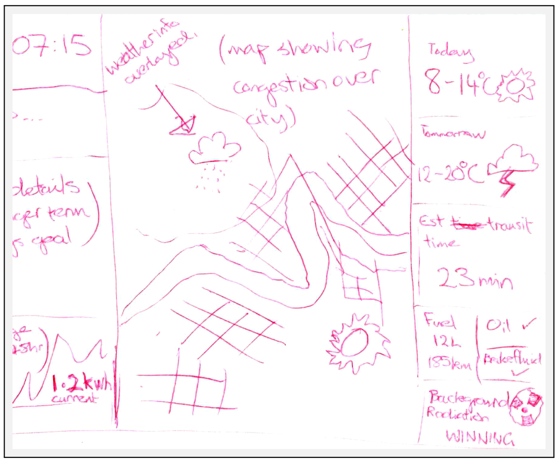
Table 4.1: Information relevant on a regular basis.

Among the more obvious examples was data related to monitoring resource consumption, covering a variety of resources. Other examples relied on data from government and urban services, such as information about weather, traffic and public transportation. Some participants were especially interested in a comprehensive overview information for planning their daily commutes. Their chosen mode of transportation would depend on weather forecasts and obstructive incidents, causing traffic jams or delays. A previously overlooked area mentioned by the participants was business or commercial data. The participants expressed interest for monitoring opening times and occupation levels of their favorite restaurants and shopping venues. Further potential was seen for the promotion of special events or time-limited sales. An original idea by one of the participants involved tracking information about his food supplies and expiry dates in order to reduce food waste.

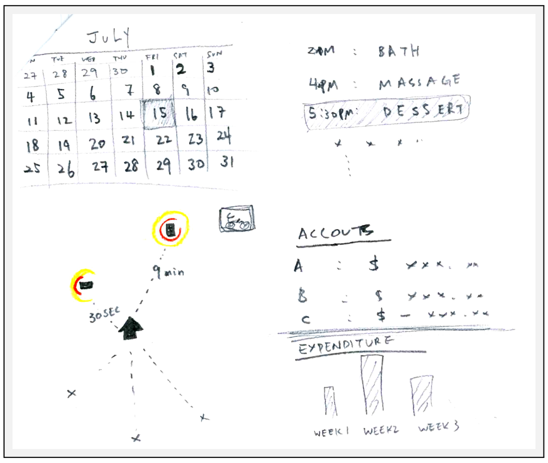
The second group task revolved around possible visual representations of the collected information. Some participants already had very specific ideas in mind regarding the interface. Since many of the participants had a background in computing, their elaborate suggestions were likely influenced by previous experience with comparable software. One proposed interface was reminiscent of the *Apple iOS* home screen, with miniature versions of the visualizations arranged as squares on a grid. The main differentiating factor was that the miniatures would provide dynamic high-level views of the data, whereas home screen icons are mostly static. Users would also be able to tap on them, causing the view to zoom in and present more detailed information. Another idea was to combine visualizations on multiple screens, allowing users to swipe through different groups. Vertical swipes would switch between different categories of information, while horizontal swipes would provide additional information and alternative visualizations within the given category. Finally, one last suggestion imagined that widgets would be arranged on a virtual plane that can be navigated using gestures for scrolling and zooming.

Apart from the interface for arranging and browsing the visualizations, there was a discussion about the visualization widgets themselves. Various participants had different views about the style of the visual representations. One participant advocated a minimalistic, functional design that is consistent with widgets modeled after standard user interface controls. The participant argued that widgets should utilize existing knowledge, opting for a simple, no-frills presentation in favor of elaborate graphical visualizations. Alluding to her familiarity with other software, the participant explained: "I look at my screen all day, I don't want to [have to] try and figure out what is going on". A somewhat contrary position was taken by another participant. He envisioned an abstract, playful interface with highly generic widgets. For example, these visual representations could take on the form of glowing orbs, encoding information using colors, sizes and relative positions.

In the following tasks, participants were given time to individually compose, sketch and present their own designs. Most participants chose to draw their designs on a separate sheet of paper, which gave them greater flexibility in terms of shapes and sizes. Some participants used the labeled pieces of paper to create the initial arrangement. Overall, the sketches demonstrated great diversity in user expectations and many of them were quite detailed, featuring innovative ideas for combining and visualizing information. The participants often made use of the limited space by displaying different kinds of information within a single widget. An example of this was a map that displays traffic congestion and overlays the current weather in the style of a weather map (see figure 4.2a). Another unique sketch included a graph visualization featuring surrounding places of interest and suggesting an appropriate mode of transportation for traveling to the selected venue (see figure 4.2b). Other sketches also featured interesting ideas, such as an virtual plaza with avatars representing friends from a social network. Energy consumption was commonly presented with a line chart displaying data covering a fixed time frame, including one exception which adopted the glowing orb metaphor for current energy usage.



(a) Example 1



(b) Example 2

Figure 4.2: Selected sketches from the focus group.

In the final group task, participants discussed various aspects of allowing users to compose and customize their personal display. One participant made an interesting remark about adapting the interface to reflect the users personality. She speculated that appeal of such customization would vary across different user groups and cultures. There was also discussion about different levels of depth for composing information. The highest level would provide ready-to-use widgets and composition would be limited to their arrangement on the screen. Composing information on a lower level could be done using a visual programming language to create simple programs processing the information and feeding the result to generic widgets. Some participants voiced concerns that the latter approach would be too generic for them.

Some participants also raised doubts whether lay users – given the freedom to compose their own information displays – would arrive at good results. One participant pointed out: “I don’t believe that people by default are great composers. So they might put things together and it looks messy.”. This implied a need to assist and guide during composition. Finally, one participant jokingly stated: “If I wasn’t doing a PhD, I would spend more time on customizing things.”. This light-hearted statement contains a deeper truth, since most of us are living busy lives, leaving little time to learn and customize tools like the Dashboard application. This is a challenge that all *Street Computing* projects will be facing, as they inherently require some level of involvement from users. The value such tools has to be clearly understood in order to motivate users to invest the necessary amount of time.

4.2 Expert Review

Once an initial working prototype was ready, an expert review was conducted to discover potential problems and ensure that the project was heading in the right direction. A total of 8 experts were involved in the evaluation. All of the attending experts were colleagues from the Urban Informatics Research Lab. Therefore, the group was mostly comprised out of PhD students and academics with experience in *Human-Computer Interaction*.

Even though the system was still at an early stage, the core components of the system architecture (see section 6) – the web service and the native tablet application – were already in place, providing all of the basic functionality. The demonstration covered the complete system, including the workflows for authorizing services and creating data streams (see section 5). The participants were encouraged to follow alongside with the presentation and setup their own accounts on the Dashboard web service. However, the subsequent usability evaluation was primarily focused on the native tablet application. For this purpose, the experts received a tablet device which had the Dashboard application installed and running. They were already logged in with a developer account, allowing them to start working directly with the Dashboard display. The gestures for switching between the different modes of the application were briefly explained. The participants were instructed to play with the configuration, giving them a chance add, remove and arrange widgets as they pleased.

Overall, the user interface was met with a positive response. In particular, the use of direct manipulation for arranging widgets was very well-received. Since the version was still in an early stage, the library of widgets was limited. Nevertheless, the participants had no problems configuring and adding those that were already available. This suggested that the chosen screen design was an intuitive match for the respective task flow. Some minor problems were reported, such as an unclear button label and occasional unrecognized gestures. At times, participants were not sure about the correct gesture to switch modes, highlighting the need for instructions within the application to guide first-time users.

Beyond that, the experts proposed several ways in which to extend the prototype. Among the desired improvements was a greater selection of widgets, including better widgets for energy monitoring. Further, the participants suggested a number of enhancements to the graph widget, such as displaying tick marks and values on the axes and the possibility to fix the horizontal axis to a 24 hour time window.

In retrospect, providing a pre-configured developer account and therefore effectively bypassing the Dashboard web service was a shortcoming of the review. The approach successfully saved time, allowing experts to start using the native tablet application immediately. This seemed reasonable, because the interface for composing and visualizing information on the tablet device embodies the key premise of the Dashboard system from an end-user perspective. However, as a result, some usability issues with the website were only identified later on.

The experts also took the opportunity to comment on the overall concept. This reaffirmed some fundamental design choices, which were based on the literature review and the focus group. The participants were enthusiastic about the visualizations that the system enabled. One of them stated: "I think having that information available to you through the different visual mechanisms is a lot more powerful than just seeing a figure.". The idea of using the system to explore more complex visualizations that combine and relate different information was popular among several participants. Furthermore, the implemented sharing functionality was considered to be very useful for social comparison, especially with regard to energy consumption.

There were differing opinions about the premise of combining unrelated information on one display. On the one hand, some were skeptical whether the presence of certain information, like social media, would provide sufficient motivation to view the Dashboard display. This might prove particularly difficult when facing competition from specialized applications. The concern was aptly summarized by one participant as follows: "If I would like to check my tweets I probably would use the Twitter app rather than a new one.". On the other hand, some experts were optimistic about the idea. One participant pointed out: "There are some feeds that I know I should be checking in a kind of peripheral vision and then there are some things that I know I check all the time.". He concluded that: "By combining the things I do all the time with the ones that I know I should, but I never do, you kind of get [the best of both].". During the lively exchange, convincing arguments were put forth by both sides. As a result, it brought attention to the fact that the concept may not appeal to everyone.

Finally, the experts were invited to share their advice for the upcoming user study. They expressed interest in finding out how such a tool would be used in a natural environment – a theme that had already surfaced during the focus group discussion. The study would present an opportunity to observe which household members used the Dashboard display and what situations they used it in. In order to gather this qualitative information, the experts stressed the importance of carrying out interviews with the users. Another aspect that was mentioned was the need for logging functionality, to collect empirical data about the usage. This was a feature that had been neglected during the early development.

4.3 Adapted Requirements

This thesis explores how the notion of *Street Computing* can be utilized in the context of energy monitoring to improve conventional solutions. Rather than displaying data related to energy consumption in a predetermined fashion, it aims to make this information available to users as a building block that can be shared and used in custom tailored solutions. This approach promises to enable new ways for users to interact with their energy data. The review of related research areas and commercial solutions revealed several opportunities for applying the concepts of *Street Computing*. These ideas where further fleshed out in a focus group and verified in an expert review. During this process, a list of functional requirements gradually came into focus. The this section describes the main goals that guided the development of the Dashboard prototype.

The following requirements were determined to be important for end-users of the application. They represent the cornerstones of the strategy to create a tool that is appealing and engaging. Further, they were selected for their potential to increase the stickiness of the application, providing incentives for users to revisit it. This aims to address the problem of users loosing interest in energy monitoring after the novelty wears off.

Seamless Integration with Information Ecosystem A review of relevant literature suggests that the success of the application will depend its integration with existing user habits and workflows. The current portable devices enable new ways for us to access and consume information. If data related to resource consumption is not well-integrated and effectively presented, it runs the risk of being drowned out in an *Age of Interruption*. However, rather than being detrimental, complementary and simultaneous media consumption across different devices can provide new opportunities for communicating information. For example, using a tablet device while watching television in the evening would present a chance to skim through the latest information and check energy usage for the day. As noted during the expert review, a combined information display could potentially create synergistic effects, motivating users to regularly check information which they might otherwise neglect. The convenience of multi-tasking with tablet devices has been well documented, allowing people to use them while performing other household activities. Since most household activities consume energy, this is an excellent occasion to present energy monitoring data, informing people at the time and place of consumption.

Flexible Interface for End-User Customization The basic premise of decision support tools is only viable if they offer enough flexibility to accommodate real-world situations and tasks. From the interaction with prospective users of the application, it became clear that one size cannot fit all. As evident from the results of the focus group, the information needs of individual users vary greatly. In addition to that, users have expressed different preferences regarding how the information should be presented. In order to address these diverse needs and expectations, it is necessary to provide a flexible interface that allows customization. The need for personalized solutions is confirmed by research literature, especially with regard to *Persuasive Technologies* for motivating sustainable energy behavior. Different individuals can be motivated by a wide range of factors, such as environmental concerns, financial savings, benchmarking and gameplay. Furthermore, researchers have identified appropriate motivational strategies for each stage of behavior change in the *Transtheoretical Model*. Once users begin to modify their behavior, it becomes important to foster their intrinsic motivation. This fits the target audience of the application, because they already made the choice of installing an energy monitor at home. By allowing users to compose their own display, the application hopes to engage users and encourage them reflect upon the information.

Networked Solution for Social Interaction Facilitating the sharing of local knowledge is an important part of the *Street Computing* vision. Technology can bring together individuals with similar social backgrounds and comparable living conditions. Within local communities, people can exchange specific recommendations and gain valuable insights. Related research literature confirms the potential of social interaction for *Persuasive Technologies*. The resulting social dynamics can motivate users to achieve their objectives, such as environmentally sustainable behavior. Users often consider socially enabled applications more enjoyable, as they allow them to collaborate and compete with others. Yet, very few commercial energy monitoring solutions exist that support these kinds of social interactions. In most cases, the data is locked away in the devices, only accessible to the respective owners. During the expert review, the attending researchers reaffirmed the importance of social features that enable sharing and comparison. Nevertheless, the implementation has to consider privacy, as pointed out by focus group participants.

Furthermore, additional requirements were defined that are intended to benefit developers. These aspects were identified to have the biggest potential contribution for the developer community. They encourage further development and extension of the application. Additionally, they aim to provide the groundwork for reusable components that will simplify future projects.

Extensible System Architecture An extensible and reusable software architecture is always a desirable goal during development, which is reflected in many of the reviewed *Information Visualization* systems. However, at the time of writing, the only commercial solution actively supporting third party developers is *The Energy Detective*. None of the commercial web applications for energy monitoring made an effort to be extensible. Virtually all commercial products for viewing energy consumption are closed, proprietary applications designed to increase the value of the associated monitoring hardware. This project sets itself apart by making extensibility a priority. For a project that aims to integrate multiple technologies and services, extensibility is indispensable. Without updates and maintenance the application would quickly become obsolete due to the rapid technological progress. There are two main areas for extensions in this project. First, the data stream platform should enable the addition of new services and data sources. For example, once the necessary sensing devices become available, it would make sense to extend the solution to monitor resources such as water, gas and others. Second, the client should support the creation of new visualizations. While the initial selection of widgets will likely be limited due to the constraints of the project, more advanced and inventive visualizations could be developed and tested in the future.

Unified API for Heterogeneous Data Streams As echoed in *Urban Computing* literature, the handling of heterogeneous data sources is a challenge for many projects in this domain. The lack of ready-to-use solutions for integrating different data sources hinders research and development, because teams have to spend resources on custom solutions for their respective projects. While it is favorable that more and more services and organizations open up their data to third party developers, each individual API incurs a learning cost for developers. The Dashboard prototype aims to take a first step towards solving this problem by defining a simple, unified API for managing and accessing data streams. In order to be viable, the solution should be flexible and provide a mechanism to incorporate new services whenever required. If this approach is adopted and extended in subsequent projects, it has the potential to generate an extensive repository of data streams that are accessible through the same API.

5 System Functionality

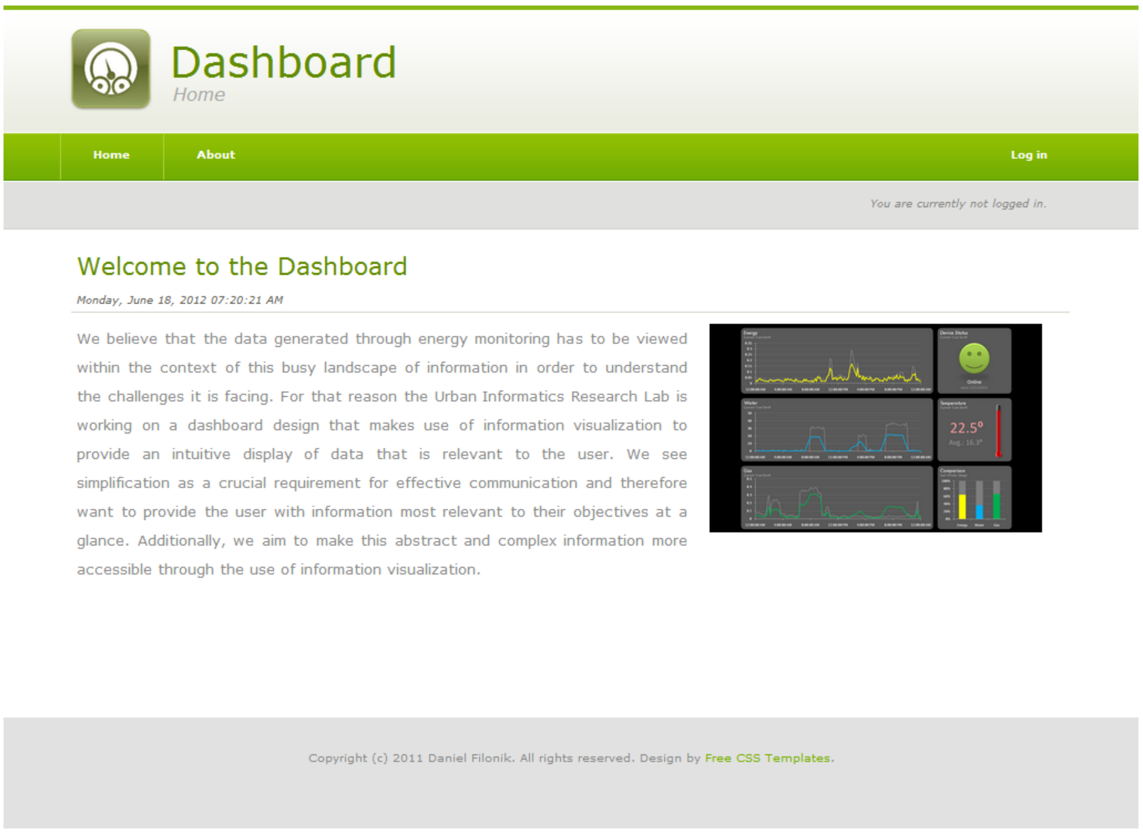
This section serves as an introduction to the Dashboard system developed for this thesis. It outlines the main concepts and features of the prototype from a user's perspective. In order to make the best use from the Dashboard system, users should become familiar with its two main components: the web service and the native tablet application. The technical details of the implementation and rationale behind the chosen system architecture are outlined in section 6.

During development and testing, energy monitoring functionality was provided by *Current Cost* devices. However, the Dashboard system can be used with any energy monitor, as long as it supports a way to upload consumption data to *Cosm*. The reminder of this chapter assumes that a suitable energy monitor is installed at the user's household. For highest possible convenience, automated data upload should be configured, which requires an always-on Internet connection.

5.1 Web Service

As a user, it makes sense to start exploring the Dashboard system by visiting the web service. The aim of this service is to provide a universal data stream brokerage platform. It allows users to access, organize and share data streams from various services. At the same time, the web service provides an API that enables authorized applications to query these streams without having to interact with the heterogeneous underlying services.

Data streams can represent a wide range of dynamic content from across the Internet. Informally speaking, a stream can be described as a series of items. Each item contains multiple attributes that form a self-contained unit of information, such as a reading from an energy monitor or a message on a social service. This definition of a stream is highly generic. Therefore, information from numerous services on the Internet can be modeled as streams. In short, the Dashboard



You are currently not logged in.

Welcome to the Dashboard

Monday, June 18, 2012 07:20:21 AM

We believe that the data generated through energy monitoring has to be viewed within the context of this busy landscape of information in order to understand the challenges it is facing. For that reason the Urban Informatics Research Lab is working on a dashboard design that makes use of information visualization to provide an intuitive display of data that is relevant to the user. We see simplification as a crucial requirement for effective communication and therefore want to provide the user with information most relevant to their objectives at a glance. Additionally, we aim to make this abstract and complex information more accessible through the use of information visualization.

Copyright (c) 2011 Daniel Filonik. All rights reserved. Design by Free CSS Templates.

Figure 5.1: General website layout in the public home section.

web service acts as a single point of contact, offering a uniform way of retrieving data which is published on the web by various services.

A web service can have more than one endpoint providing information that can be exposed as a data stream. In order to model these endpoints, the Dashboard web service introduces the concept of a source. For example, social networking platforms can be powerful tools, hosting many different kinds of dynamic personal information. The plugin for integrating such a hypothetical service could have one source for streams of personal messages and a different source for streams of pictures from photo albums.

Data streams can come in different varieties, containing items representing various kinds of information. The Dashboard web service adopts a loose concept of data stream types, referring to them as signatures. All items in streams belonging to one specific source share the same signature. Different sources can have identical signatures, meaning that their streams are interchangeable from the perspective of a client that consumes data streams.

At the time of writing, the Dashboard web service is hosted in a subdomain[6] of the official Urban Informatics website. The page can be accessed by using one of the major browsers on a desktop computer, tablet or any other Internet-capable device. The public section is comprised out of a static landing page featuring a description of the project and an about page listing credits and basic disclaimers. All functionality resides in the private section of the website, which becomes available after logging in.

User Account and Profile

Authentication is based on the *OpenID* standard, so there is no need to explicitly register an account. Instead, users simply log in using one of their existing accounts with any *OpenID* provider. The login page features a selection of popular providers with dedicated buttons, allowing users to login with a single click. Less common services are supported through a text input field for arbitrary *OpenID* provider URLs. At the first time a user logs in with a certain provider, a user account is automatically generated and populated with personal information retrieved via *OpenID* attribute exchange. Users should log in with the same provider on subsequent visits. Using a different provider will result in the creation of a separate user account on the Dashboard website – unless *OpenID* delegation is in place.

Upon logging in, users are automatically redirected to the private section of the web service. This opens an overview page exposing all the core functionality of the web service, which will be discussed in the next section. If users wish to view or edit their personal information, they can do so by selecting ‘Profile’ from the main navigation menu. The Dashboard web service only stores a minimal set of personal information – including the full name, a contact email and an optional profile image. Besides personal information, the profile page also shows a list of friends. The social features of the web service will be examined more closely in a dedicated section. Finally, there is also a list of all devices that the user has been active on. This list is automatically compiled from the usage logs that are kept to collect empirical data for evaluation.

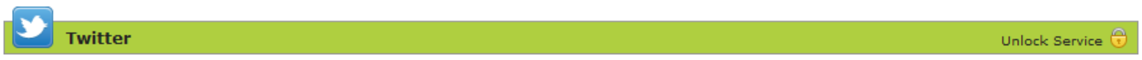
Overview and Stream Management

The core functionality is accessible from an overview page, which is displayed once a user enters the private section of the web service. If a user navigates away from this page, they can always return by selecting ‘Home’ from the main navigation menu. The primary purpose of the web service is to allow users to configure and organize streams, which provide data for the visualizations in the native tablet application.

The three main concepts of the Dashboard web service – services, sources and streams – exhibit a hierarchical structure. The overview page aims to convey this structure by visually grouping

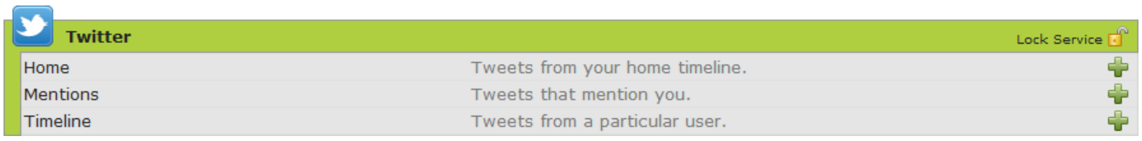
[6]http://dashboard.urbaninformatics.net

the different entities. For good reasons, like privacy or security, certain sensitive information cannot be made available freely to third parties. In order to retrieve data from services dealing with this kind of information, users have to explicitly provide their authorization for the Dashboard web service to gain access. The process of giving and revoking authorization is depicted on the overview page as unlocking and locking a service. When a service is locked the entire group is collapsed, signaling that any associated sources and streams are not accessible.



Twitter — Unlock Service

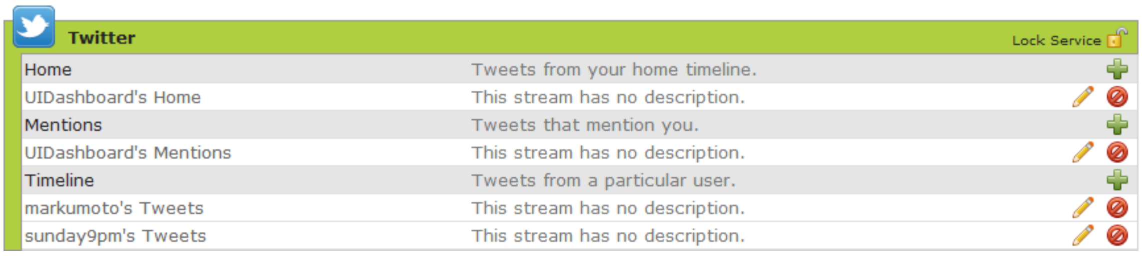
(a) Locked.



Twitter — Lock Service

Home	Tweets from your home timeline.	➕
Mentions	Tweets that mention you.	➕
Timeline	Tweets from a particular user.	➕

(b) Unlocked.



Twitter — Lock Service

Home	Tweets from your home timeline.	➕
UIDashboard's Home	This stream has no description.	✎ ⊘
Mentions	Tweets that mention you.	➕
UIDashboard's Mentions	This stream has no description.	✎ ⊘
Timeline	Tweets from a particular user.	➕
markumoto's Tweets	This stream has no description.	✎ ⊘
sunday9pm's Tweets	This stream has no description.	✎ ⊘

(c) Unlocked with user-created streams.

Figure 5.2: Twitter service on the overview page.

Users can create new data streams by clicking on the green plus icon next to the desired data source. The creator of a certain stream is considered to be its owner. Users can edit or delete their own data streams at any time. Editing is triggered by clicking on the the yellow pen icon, whereas deletion is symbolized by a red circle icon. The icons were chosen in accordance with common conventions regarding shape and color cues.

Stream creation is guided by a simple two step wizard process. In the first step, the user enters general information such as title and description. This step is always the same, regardless of the type of stream being created. Once the user finishes inputting information he proceeds on to the second step by clicking ‘Apply’. In the second step, the user enters information specific to the particular data source and confirms once again with ‘Apply’. Depending on the type of stream, the second step may not be necessary at all. If this is the case, it will be skipped automatically. After completing the final step, the user is redirected back to the overview page. The newly created data stream will appear under the corresponding source featuring the user-defined title and description.

Once a stream has been created, the Dashboard web service offers basic support for querying its contents. Users can access this functionality by clicking on a stream in the overview page. This opens a page with a rudimentary interface for specifying and executing queries. By default, a fixed a number of items is retrieved and displayed. This allows users to take a peak at the stream contents. Advanced queries are also possible, but they have to be entered manually using the correct syntax. Because of this limitation, the feature is currently only practical for expert users or developers. Nevertheless, it is a valuable tool for debugging and the ability to export query results provides a real benefit to expert users.

Supported Services and Sources

The Dashboard web service already integrates a sizable collection of services, which are listed on the overview page. Table 5.1 provides a summary of supported services, including the data sources they provide. Following a brief introduction to each service, this section outlines the individual sources and describes the information that they encompass.

Service	Sources	Signatures
Bureau of Meteorology	*Forecast*	Values, Messages
Cosm	*Datastream*	Values
Facebook	*Home*	Messages
	Photos	Pictures
	Wall	Messages
Flickr	*Photostream*	Pictures
Foursquare	*Checkins*	Locations
	Mayorships	Locations
Google	*Mail*	Messages
	[Calendar]	Events
RSS	*Feed*	Articles
Twitter	*Home*	Messages
	Mentions	Messages
	Timeline	Messages
XKCD	*Comics*	Pictures

Table 5.1: Supported services and sources.

The selection is based on a range of criteria. For one, the choices are based on the ideas that were collected during the focus group discussion. Further, feedback from expert reviews and early test users was taken into account and popular suggestions were added. This user-centered approach is essential for the design of the Dashboard system, because the entire concept is based on the premise on providing personalized and highly relevant information. Furthermore, services that provide dynamic information were favored over those that are mostly static. This was done to promote lively visualizations which are changing over time, as they are more likely to maintain user interest. Finally, the choices were coordinated to provide a diverse, varied mix of information. It is important to give users many options for customization, which brings about a sense of freedom. Ideally, the selection should also showcase capabilities of the Dashboard web service, possibly inspiring further uses.

The *Bureau of Meteorology* is an Australian government agency reporting on weather, climate and water conditions. It plays an important role for locals, helping them to cope with the harsh natural environment. The information published on the website includes regular forecasts and extreme weather warnings. For the time being, the support is limited to a single data source: *forecasts* for selected cities. The streams contain short messages from a set of standard phrases and the predicted temperature ranges for a given date.

The *Cosm* platform enables storing and retrieving real-time sensor data. It promotes openness, allowing users to share and discover sensor feeds. The clear focus on supporting the vision of an Internet of Things has allowed *Cosm* provide a simple and targeted API. Thus, It is a compelling backend for devices that need to exchange data collected from the environment, such as energy monitors. As a result, is an indispensable part of the Dashboard system, providing the *datastream* source that produces streams of energy monitor readings.

Facebook is a prominent representative of social networking services. It supports a large and active user base in many countries around the globe and it is very popular amongst Australians. With first and third party applications, the *Facebook* platform features a powerful set of tools for storing personal data and facilitating social interactions. A lot of this highly dynamic information lends itself to be represented as data streams. The *home* and *wall* sources generate streams with public and private messages. Further, users can create streams containing *photos* from one of their albums.

The *Flickr* service is dedicated to online photo sharing. Services that are specialized on a single task are well suited to illustrate a certain concept. In this case, *Flickr* is a showpiece example of a data source that provides images. The service does have strong competition from the general purpose tool *Facebook*. Nevertheless, it was determined to be a worthwhile addition. Apart from some peculiarities, its API made it fairly easy to integrate the service with the Dashboard system. As a result, the *photostream* source gives users another option for creating data streams from personal photo collections.

Foursquare is a location-based service, which revolves around the concept of checking in to places that are nearby using a GPS enabled mobile device. The service incorporates gaming aspects by letting users score points and earn badges for various actions, most notably check-ins. The *Foursquare* service stands out because of its focus on location based and context sensitive information. Is features an extensive repository of venues and their respective GPS coordinates. Recent check-ins shed light on user activities and presence. Currently, the Dashboard system supports two data sources: *checkins* and *mayorships*. The *checkins* are recently visited venues and the *mayorships* represent venues that a user visited more often than any other user. Streams from both sources contain the names of venues and their GPS coordinates.

Google has cemented its standing as the dominant search engine provider. However, that is far from the only service in its portfolio. Notable services include *Google Mail* and *Google Calendar*, which are popular tools for personal communication and organization. As of now, the Dashboard web service is limited to these two sources, integrating *mail* and partially supporting *calendar*. They provide streams of unread messages from an inbox and calendar events. In the future, there is potential for additional sources, for example *Google Docs* could be used to create data streams from spreadsheets.

RSS stands out from the other examples, because it is a standardized format for publishing website updates rather than a service. The standardization enables automatic content syndication, allowing users to keep track of changes across multiple unrelated websites. Despite its unique position as a standard, it is well suited to be modeled using the concepts of the Dashboard web service. The *feed* source allows users to create streams that contain the latest updates from their favorite websites. However, *RSS* is not just used by content providers for publishing media, such as news. Some Australian government agencies and community organizations also employ this format for distributing time-sensitive information, such as traffic incidents or weather warnings.

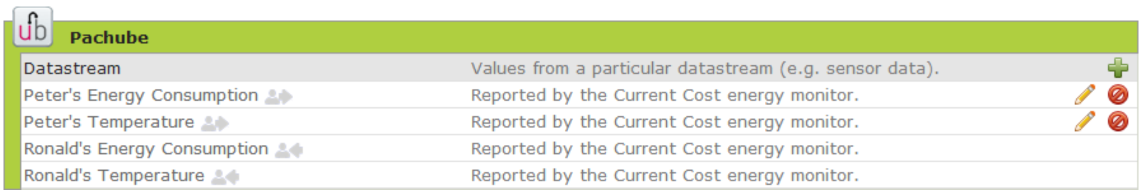
The *Twitter* messaging service is a popular online communication channel. It allows users to publish short messages of up to 140 characters. It supports social interaction by providing a special syntax for addressing them in a message. Further, it is possible to keep up with other users by subscribing to them and thereby following their updates. The Dashboard system currently supports three data sources: *home*, *mentions* and *timelines*. The *home* streams include messages by anyone that the user subscribed to, whereas *timelines* only contain messages from a specific user. The *mentions* source lets users streams with all messages that are addressed to them. Despite the character limitation, *Twitter* messages have proven useful beyond simple gossip and banter. For example, during the Queensland floods in 2011 an active discussion formed under the `#qldfloods` hashtag. Besides voicing reactions to the floods, users organized help and fund raising. The discussion provided timely on-site reports about the flooding, which informed the coordination of emergency response efforts. The official *Twitter* channels of the Queensland Police Service and the Brisbane City Council were actively involved.

Friends and Stream Sharing

Social interaction and cooperative problem solving are key ingredients for *Street Computing*. The ability to exchange and compare data allows users to collaborate and share their expertise. The Dashboard web service is a very natural place for implementing sharing functionality between users. Individual data streams are viewed as resources and their owners control the access to them. Naturally, privacy is an important concern, because streams may contain highly personal information. Therefore, it was important to ensure that users have control over what they share and with whom.

The Dashboard web service implements a friend system to allow users to interact with trusted individuals. Friendships can be managed by selecting 'Friends' from the main navigation menu. As in most social services, the system is based on friend requests. The search feature of the Dashboard web service allows users to look for their acquaintances by name. Users can view and confirm outstanding friend requests on their profile page or on the friends page. When both users agree to become friends, a friendship is established. If one of the two users withdraws their consent at any time, the friendship gets revoked.

As part of the general configuration of a stream, users can choose a privacy setting for it. Currently, the available options are 'private' or 'shared'. Once a friendship is made, each user gains access to the other user's streams, unless they have been marked 'private'. Shared streams are highlighted on the overview page with special icons. The direction of the arrow indicates whether the user is sharing the stream with friends or receiving it from a friend. Streams that are received from friends cannot be modified or deleted, because the user is not considered to be the owner of these streams.


Pachube

Datastream	Values from a particular datastream (e.g. sensor data).
Peter's Energy Consumption	Reported by the Current Cost energy monitor.
Peter's Temperature	Reported by the Current Cost energy monitor.
Ronald's Energy Consumption	Reported by the Current Cost energy monitor.
Ronald's Temperature	Reported by the Current Cost energy monitor.

Figure 5.3: Shared streams on the overview page.

The friend feature is an excellent way to introduce newcomers to the Dashboard web service through somebody who is already familiar with it. If new users establish friendships with existing users, they automatically gain access to their shared streams. This significantly lowers the entry barrier, because their overview is immediately populated with streams that they can play with. They serve as examples to illustrate the possible uses of the Dashboard web service. These streams might give novices ideas on how to use the system to their own benefit and inspire them to try new things.

Administration and Usage Statistics

The Dashboard web service implements a basic system of roles and permissions that determine which functions are accessible to a certain user. The production system defines three roles, which can be ordered by increasing extent of permissions: *user*, *moderator* and *administrator*. The *user* role is assigned to newcomers by default. All previous sections describe functionality that is available to any authenticated user. Contrary to that, the administrative features outlined in this section require *moderator* or *administrator* status.

The *administrator* role has the authority to configure all aspects of the Dashboard web service. Most importantly, this includes the ability to perform basic user management, through the 'Users' and 'Roles' entries in the main navigation menu. Only users with the *administrator* role are permitted to modify existing roles and create new ones. Since the roles are central to the integrity of the system, only selected individuals should be in this position. Further, administrators can manage individual users, allowing them to change their assigned roles. For example, a default *user* can become promoted to a *moderator* by having an *administrator* change his role.

An advanced feature available in the *administrator* role is impersonation. It allows administrators to assume the identity of any member with the default *user* role. This feature is very valuable for troubleshooting problems or performing configuration on behalf of the users. The main benefit is the ability to perform these tasks without asking users to reveal their OpenID credentials. However, it might be necessary to scale back this functionality due to privacy concerns.

Both, the *moderator* and *administrator* roles have full permissions to manage services and sources, which are available to all users of the Dashboard web service. Once a plugin for a new service or source is deployed on the server, it is necessary to create a corresponding entity in the Dashboard web service. Through these entities it is possible to enable individual plugins and modify the presentation on the website. Therefore, moderators and administrators have fine grained control over the kinds of data streams that users can create.

The Dashboard web service is capable of generating reports from the usage logs that are kept for empirical evaluation. The raw data is processed and compiled into a list of meaningful measures. For each active user, these values offer insight into their interactions with the native tablet application. All calculated values are based on the time frame between the first and the last recorded interaction. As part of the analysis, individual user actions are grouped into sessions. This forms the basis for calculating the total number of sessions and their accumulated duration. Additionally, the temporal sequence of sessions is visualized on a timeline. In order to provide more context, the information also includes the total uptime of the tablet device, as well as amount of time during which the device was active or asleep. Further, the report provides indicators for the amount of customization by the user, such as the number of widgets that were created and deleted. Finally, the Dashboard web service automatically collects screenshots from the tablet application, allowing to visually evaluate the progress between different Dashboard configurations.

5.2 Native Application

Once users have configured a couple of data streams on the web service, they can really start benefiting from the Dashboard system. At that point, it makes sense to move on to the the native tablet application. The main purpose of the Dashboard application is to allow users to compose and visualize data streams. The web service is used to discover and query streams, enabling the application to obtain the data that drives its visualizations. In this respect, the Dashboard application is a client of brokerage platform, acting as a consumer of data streams.

The Dashboard application adopts the familiar concept of widgets for representing individual visualization components. Initially, the display is empty, providing users with a blank canvas on which they can arrange their ideal selection of widgets. This supports the key objective of allowing users to tailor the display to their personal needs and interests. Users can choose from a library of widgets that is bundled with the application.

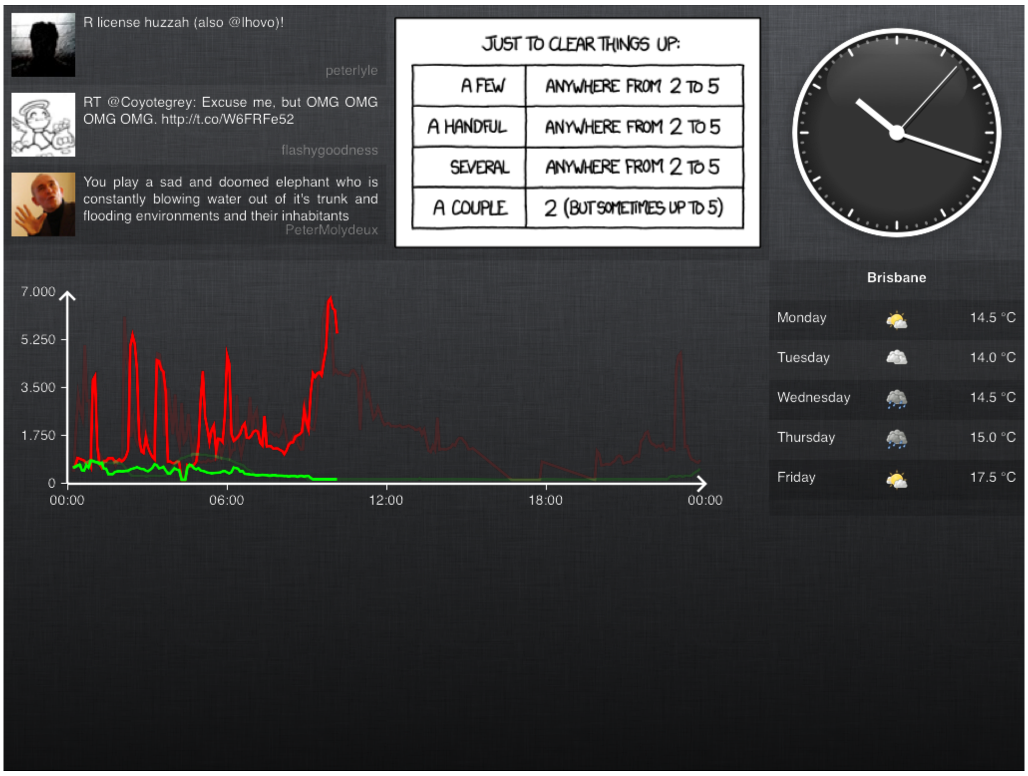
In order to run the Dashboard application, users need a supported tablet device. The initial prototype version was developed for *Apple iPad* tablets. The popularity of these devices amongst consumers was a determining factor, since their proliferation made it easier to recruit a group of potential test users. Since the application was still in prototype stage, it was not released in the official *App Store*. Instead, the deployment was managed by the *TestFlight* service. This service allows over-the-air distribution of pre-release applications to a selected group of test users. After signing up with *TestFlight* and joining the test team, users can simply download and install the latest build onto their device.

Once installed, the application can be launched by touching its icon on the home screen, as customary on *Apple iPad* tablets. The Dashboard application supports multiple user accounts on a single device, which means that users have to authenticate themselves before accessing their personal Dashboard display. The display has two main modes of operation: ‘view’-mode for quick information retrieval and ‘edit’-mode for widget configuration and composition. The user can toggle between these modes using long press and tap gestures. When a user logs in to the application for the first time, his Dashboard display is completely empty. In order to guide novice users that are just getting started, a help message appears that describes how to toggle between the two modes.

User Account and Overview

As of now, the Dashboard application and web service are tightly integrated with regard to user management. Therefore, users perform the same login procedure as they would on the website. This serves two purposes: the user is authenticated and the application is authorized to access his personal data streams. In the long run, it would make sense to increase the separation between the two components. In this case, the Dashboard application would implement its own authentication and user management independently of the web service. A user remains authenticated until he explicitly logs out or the application is terminated. Suspending the application does not invalidate the authentication, allowing users to minimize the application and return later without having to log in again.

Once a user is authenticated, he is presented with an overview of his personal Dashboard display (see figure 5.4). The configurations are stored online, meaning that users will see the same results across different devices. The ‘view’-mode is optimized for information retrieval. This remains true to the original concept of business intelligence dashboards, allowing the user to see all personally relevant information at a glance. By design, the display area is limited to a single screen. Other concepts that would provide additional space for widgets – such as pagination or scrolling – were discarded, as they would violate the principle of having a complete overview on one display. Consequently, screen real estate is a valuable resource. Therefore, special care was


R license huzzah (also @lhovo)!
peterlyle

RT @Coyotegrey: Excuse me, but OMG OMG OMG OMG. http://t.co/W6FRFe52
flashygoodness

You play a sad and doomed elephant who is constantly blowing water out of it's trunk and flooding environments and their inhabitants
PeterMolydeux

JUST TO CLEAR THINGS UP:

A FEW	ANYWHERE FROM 2 TO 5
A HANDFUL	ANYWHERE FROM 2 TO 5
SEVERAL	ANYWHERE FROM 2 TO 5
A COUPLE	2 (BUT SOMETIMES UP TO 5)

Brisbane

Monday	14.5 °C
Tuesday	14.0 °C
Wednesday	14.5 °C
Thursday	15.0 °C
Friday	17.5 °C

Figure 5.4: Basic configuration in ‘view’-mode.

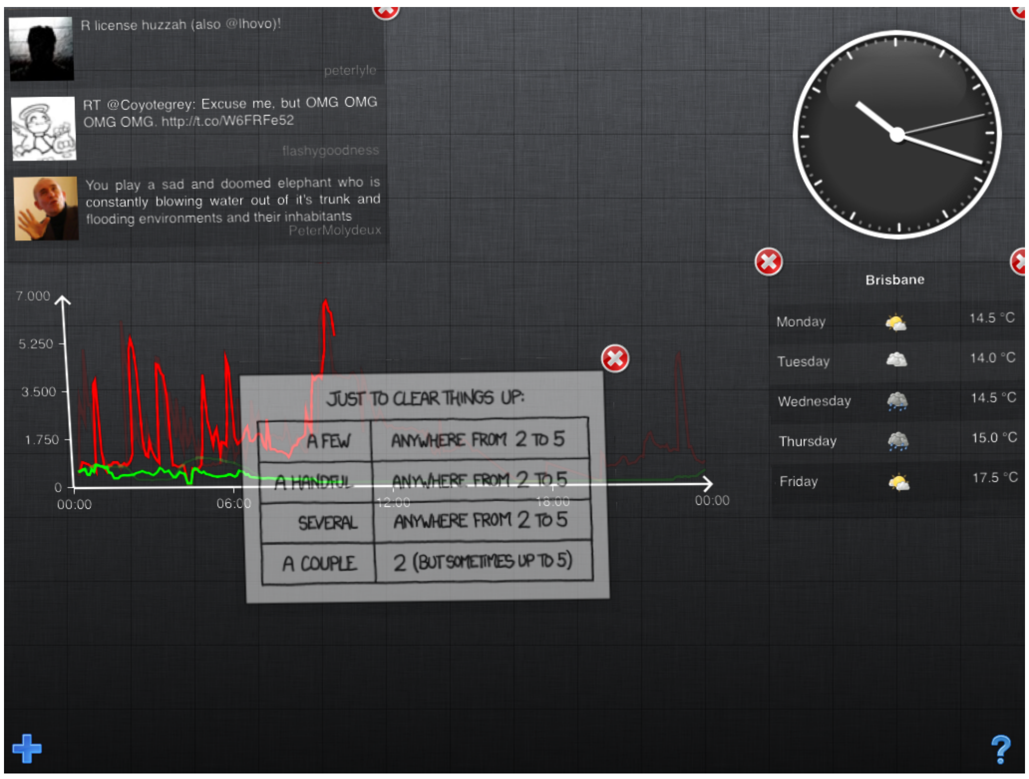
taken to allocate a maximum amount of space to the visualizations. The application refrains from displaying any potentially distracting interface elements, such as the status bar or buttons.

On first and second generation *Apple iPad* tablets, the screen resolution is 1024 × 768 pixels. Assuming a comfortable average widget size of 256 × 256, the Dashboard display provides space for a total of 12 widgets. This number seems sufficient, considering that 7 ± 2 is generally viewed as a rough guideline for the number of items that people are comfortable working with. Naturally, this needs to be taken with a grain of salt, as it is highly dependent on the individual user. Furthermore, it is a simplification to equate a widget with a single item, since different visualizations can vary greatly in complexity. Nevertheless, the Dashboard display should be capable to accommodate a fair share of user types with visualization needs.

In ‘view’-mode, all widgets are locked, meaning that they do not respond to touches. This was done to prevent accidental modification. The only possible action is focusing on a certain widget by performing a double tap gesture. When a widget receives focus, the view translates and zooms, making the visualization fill the entire screen. In this state, widgets can receive touch input, allowing for interactive visualizations. However, the current basic visualizations do not make use of this functionality yet. Users can withdraw focus and return to the overview by repeating the double tap gesture.

Configuration and Composition

In order to apply modifications to their Dashboard display, users enter ‘edit’-mode with a long press gesture. A number of visual cues highlight the transition to a different mode of operation (see figure 5.5). Any existing widgets start moving in a slight wiggle motion, signaling that they can be rearranged. Besides that, a grid which enables user to align widgets fades in. Further, two buttons – previously hidden to minimize visual clutter – appear in the lower corners. The question mark in the lower right corner takes the user to an about screen. That screen displays general


R license huzzah (also @lhovo)!
peterlyle

RT @Coyotegrey: Excuse me, but OMG OMG OMG OMG. http://t.co/W6FRFe52
flashygoodness

You play a sad and doomed elephant who is constantly blowing water out of it's trunk and flooding environments and their inhabitants
PeterMolydeux

Brisbane

Monday	14.5 °C
Tuesday	14.0 °C
Wednesday	14.5 °C
Thursday	15.0 °C
Friday	17.5 °C

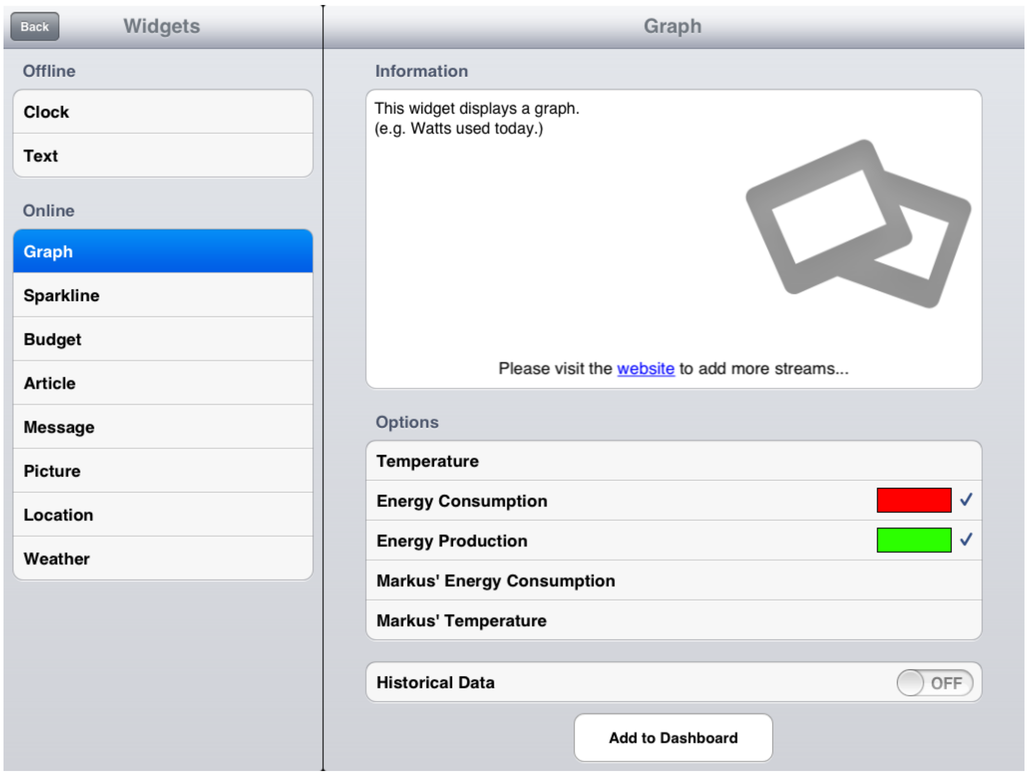
JUST TO CLEAR THINGS UP:

A FEW	ANYWHERE FROM 2 TO 5
A HANDFUL	ANYWHERE FROM 2 TO 5
SEVERAL	ANYWHERE FROM 2 TO 5
A COUPLE	2 (BUT SOMETIMES UP TO 5)

Figure 5.5: Basic configuration in ‘edit’-mode.

information about the application as well as the name of the authenticated user. There is also a button that allows the current user to log out. The plus sign in the lower left corner opens a screen that allows users to browse, configure and add widgets to the Dashboard display.

The screen for configuring widgets is split in two parts. The left side is static and lists all available widget types, whereas the right side offers detailed information and configuration options for the selected type (see figure 5.6). Depending on the visualization method and implementation, widgets provide a range of configuration options. In most cases, users can adjust visual properties of the presentation. Most importantly, widgets may consume a fixed or variable number of data streams. In order to discover compatible data streams, widgets advertise the expected signature to the Dashboard web service. A matching list is compiled by the service, allowing users to choose the data streams that they would like to see visualized. A special case is presented by widgets that do not require any data streams. They are referred to as being 'offline' as opposed to the rest which are 'online'.



This widget displays a graph.
(e.g. Watts used today.)

Please visit the website to add more streams...

Temperature		
Energy Consumption	🟥	✓
Energy Production	🟩	✓
Markus' Energy Consumption		
Markus' Temperature		

Historical Data OFF

Figure 5.6: Configuration of a graph widget.

Offline widgets require no interaction with the Dashboard web service and therefore no Internet connection. They visualize basic information that is available locally on the device, such as the clock widget which displays the current system time. These widgets are a good place to start for new users, allowing them to familiarize themselves with the mechanics of adding, removing and arranging widgets on the Dashboard display, without getting involved with data streams. However, the selection of offline widgets is fairly limited. In order to make the best use of the Dashboard application, users should turn to online widgets. In contrast to their offline counterparts, these widgets need an Internet connection to retrieve content by querying data streams from the Dashboard web service. As part of the configuration of an online widget, the user selects one or more streams of interest from the filtered list supplied by the web service.

Once a users finishes configuring his widget of choice, he completes the process by pressing the 'Add to Dashboard' button. This closes the configuration screen and returns the application to the Dashboard display, which remains in 'edit'-mode. The user can now position the widget by putting his finger down on it and dragging across the screen. Additionally, by performing a pinch gesture with two fingers the user can scale the widget. When the user lets go of the widget by lifting his fingers from the screen, it automatically snaps to the underlying grid. This feature helps the user to neatly align widgets next to each other. Currently, the Dashboard application does not restrict widgets from overlapping. In most cases, it is not sensible to place multiple widgets top of each other. It might be necessary change this behavior, depending on whether it turns out to be useful or distracting.

Supported Widgets

The current version of the Dashboard application comes with a collection of basic widgets. Table 5.2 lists all available widgets and the visualization methods they use to present the information. This section briefly introduces each widget and outlines the intended usage and supported configuration options.

Widget	Visualization	Signature
Clock	*Analog Clock*	—
Note	*Text*	—
Graph	*Line Chart/Graph*	Values
Sparkline	*Sparklines*	Values
Budget	*Progress Bar, Stacked Column*	Values
Article	*List, Long Texts*	Articles
Message	*List, Short Texts*	Messages
Picture	*Slideshow*	Pictures
Location	*Map, Markers*	Locations
Weather	*Table, Icons*	Values, Messages

Table 5.2: Supported widgets.

The selection of widgets was developed to support specific use cases imagined by the focus group. Further, it was desirable to have at least one type of widget for the different groups of data streams with distinctive common signatures that are provided by the Dashboard web service. The chosen visualization techniques are fairly basic. Once the Dashboard system is in a stable and mature state, it would be intriguing to experiment with more advanced visualizations. However, due to the focus on quick information retrieval and composability, there may be limits to the viable complexity of graphical representations. Further, since the main target audience of the Dashboard system are visualization novices, some participants of the focus group and expert review expressed concerns about the adoption of advanced visualization techniques. Nevertheless, the widgets demonstrate the platform independent graphics capabilities of the Dashboard application. Hopefully, this will encourage developers to extend the library of widgets and explore new visualization methods.

The *Graph* widgets displays a classical line chart. Its implementation is very generic, making it suitable for visualizing any continuous series of items with two numerical attributes. It is possible to automatically calculate the ranges for the two axes to fit the data or set them programatically. The version that is exposed through the user interface is specialized for time series. The horizontal axis always represents time and the labels are formatted accordingly. Furthermore, this axis was fixed to a 24 hour time window, whereas the vertical axis automatically scales to fit the data. Based on user feedback, this configuration was determined to be the preferred way of viewing time-based sensor data – and particularly energy monitoring data. As part of the configuration, users select from a list of data streams, where every stream produces a time series of values. Multiple series can be displayed within one chart, each brushed with a customizable color. An auxiliary feature that can be enabled by users is the display of historical data. In addition to the most recent sensor data, this also renders readings taken on the same day in the previous week for comparison. Data collected on the same weekdays was deemed to be most likely to be comparable. However, it could also be interesting to consider averages for historical reference.

The *Sparkline* widget provides a compact variation of the line chart, which saves space by omitting the axes. Despite this omission, it is well suited for communicating the general shape of a graph and spotting trends. In order to maintain readability when dealing with multiple series, each sparkline is rendered separately. The individual plots are arranged vertically in a list. As in the case of the *Graph* widget, the user can customize the brush color for each series. The *Sparkline* widget was implemented as an option for reducing visual clutter on the Dashboard display and making better use of the limited screen real estate.

Budget widgets allow users to track their performance relative to a chosen target goal. Especially in the context of energy monitoring, clear goals have the potential to boost energy savings. This widget also requires a stream from which it queries time-based values in a fixed 24 hour time window. However, unlike the *Graph* or *Sparkline*, individual values are accumulated into one total. The total is then compared to a user-defined goal. The result is presented using a progress bar, which is color-coded in green, yellow or red, depending percentage difference relative to the target value. Additionally, the widget can display a marker by linearly interpolating based on the current time with respect to the time period under consideration. This serves as a rough estimate for how far the progress bar should advance at a given time in order to reach goal by the end of the time window.

The *Article* widget is a general purpose solution for displaying longer texts. The only configuration consists out of choosing the stream providing the article contents. In contrast to the *Message* widgets, the layout can accommodate one or two of short paragraphs of text. Additionally, the items need to contain a title that will be displayed above the text. The current implementation simply renders articles in a list – one underneath the other – and truncates any overflowing text. There is still a lot of room for enhancing the visual appearance and supporting user interaction. For example, it would be possible to render articles as pages in a book or a newspaper. While this could make the widget more appealing, critics of such skeuomorphic designs point out that they put form before function and require more screen space.

Message widgets are an alternative to *Article* widgets – specialized on shorter texts. Analogous to their counterparts, users select a stream providing the message contents. The messages are displayed in a more compact list than articles. Since messages often are part of personal communication, it is important to be able to identify the sender. In order to communicate this information efficiently, the widget renders an avatar next to the text content whenever possible. Additionally, the name of the author is displayed at the end of the message. As with the *Article* widget, the implementation would also benefit from interactivity, for example allowing users to paginate or scroll through long lists of messages. In any case, these textual widgets are currently the primary option for including highly personalized content on the Dashboard display, such as messages from social networks or articles from popular websites.

The *Picture* widget allows users to display a slideshow. It is currently the only widget dedicated to visualizing streams of images. During the configuration, users can choose the desired stream and adjust the display duration. The slideshow cycles through a set of images retrieved by querying a fixed number of the latest items from the stream. This enables users to personalize their Dashboard with their online photo collections or include other image media, such as web comics.

The *Location* widget produces geographical visualizations using maps. It consumes a fixed number of items from a stream containing location-based data. As customary, users can select the stream from a list of compatible candidates during configuration. The widget displays the results on a section of a map. The section is appropriately sized to fit the given geographic locations. Markers are placed on the map in order to highlight the relevant spots. While it is still pretty basic, the visualization can be of interest to certain user groups, such as active users of location-based services. They can use the widget to plot locations which they have recently visited, allowing them to track their activities. Another possible use would be to subscribe to streams which provide locations of events that a user might be interested in.

Weather widgets display weather forecasts in tabular form. Currently, a single stream represents forecasts for a geographic area, such as a city. Therefore, users can configure the *Weather* widget to display forecasts from their local area by selecting the corresponding stream. Each row represents the expected weather for an upcoming day, starting with tomorrow's forecast at the top. The rows include the name of the weekday, an icon representing the weather conditions and the average temperature. Weather forecasts are well suited for the Dashboard display, because many users value this information and have incentives to check it regularly. Further, participants of the focus group pointed out that information about the weather would help them analyze their energy monitoring data.

6 Implementation

This section presents the technology behind the Dashboard system. The development was driven by the objective of delivering a functional prototype for evaluation within a real-world context. Many technological decisions were made to support rapid and agile prototyping. This was considered as an important prerequisite to execute a user-centered design approach. The goal of this section is to provide documentation for the key components of the system, as well as explain the rationale behind the adopted solutions.

6.1 Architecture Overview

The complete Dashboard system is comprised out of two main components, a web service and a native tablet application. An online server is hosting the web service, whose primary client is the native tablet application. Furthermore, users can access specific functionality of the web service through a dedicated website. Figure 6.1 shows a schematic overview of the client-server interaction.

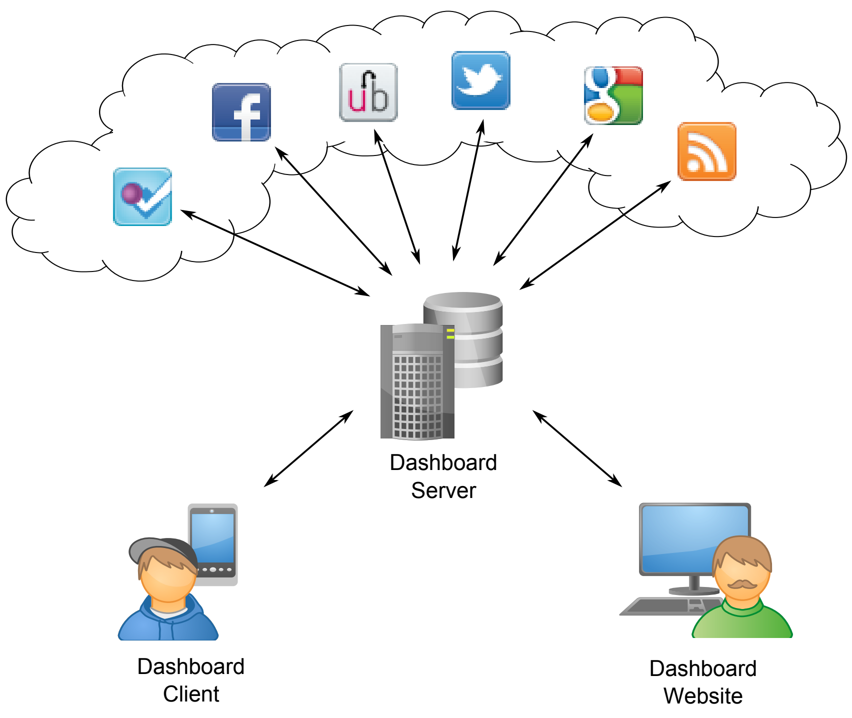


Figure 6.1: External architecture overview.

Each component of the Dashboard system was designed with a distinct goal – trying to address a specific set of problems. It would have been possible to tightly integrate all functionality into one application. However, due to the different characteristics and technical requirements of each problem space, a clear separation was deemed to be preferable. The main reasons can be summed up as follows.

Preserving clear separation of concerns On the one hand, the web service acts as a data stream broker. It adapts heterogeneous data sources from a wide variety of services and exposes them in a uniform manner. On the other hand, the native application is a data stream consumer. It queries the data streams and displays the results in the form of visualizations. These distinct areas of responsibility warrant a clear separation.

Using the right tools for the job Since each component serves a fundamentally different purpose, they invite different solutions. In order to integrate with the information ecosystem of the web, the data stream broker relies on various web technologies. An effective implementation requires comprehensive support of web standards through libraries. In contrast to that, the main focus of the native application is visualization. High quality graphics and smooth interaction pose demands in terms of performance, which – despite considerable advances in web browsers – are more easily met by a native application.

Promoting reuse and extension Drawing a clear separation between the components increases the likelihood that they will be used in subsequent projects. Because the problem areas are clearly defined, the components lend themselves as building blocks for future developers. In particular, the web service is easy to integrate through the API, which has proven its functionality as the basis for the native application.

It would be possible to make the functionality of the web service available exclusively through the API. This would put more burden on the clients, requiring them to provide their own interfaces for managing services and streams. Instead, the decision was made to create a web interface for the web service, reducing the reliance on specific clients.

Stand-alone solution The web service can be deployed as a stand-alone solution for data stream management. It acts as a central place for users wishing to organize their personal data streams. Beyond standard end-user functionality, the web interface offers convenient support for administrative tasks. Further, it provides an easy way for developers to test the web service before committing to its API.

Matching interfaces to tasks The relatively complex task of data stream management involves a fair amount of typing and form input. It makes sense to provide this functionality on a website, giving users the option to perform these tasks on a desktop computer. Contrary to that, interaction with visualizations – and particularly their arrangement – is very well suited for direct manipulation interfaces on a touch-screen device.

Nevertheless, there are reasons against having two separate components. For example, users might resent having to switch between different interfaces. Further, the web service might not deliver a clear and immediate benefit for users, because the real value of data streams comes from the applications that are using them. From a technical perspective, these concerns are relatively minor. They could be addressed by adding more features to the native application and implementing them on top of the API provided by the web service.

Internally, the web service and the native tablet application are realized using a modular software architecture, which consists out of several independent components. Furthermore, the components integrate multiple existing libraries from the open source community to accomplish specific tasks, ranging from the support of web standards on the server to advanced graphics rendering on the client. A particular area of attention was the interface between the server and the client, which takes the form of a simple, well-defined API.

Figure 6.2 provides an overview of the internal software architecture for the Dashboard server and client. The goal to promote reuse and extension is reflected in the design. The current client was developed for *Apple iOS*, but the use of cross-platform libraries opens the door for ports to other platforms, such as *Android*. A detailed discussion of the technical implementation is presented in the following sections. The individual software components – depicted in the diagram – are described in the corresponding server-side and client-side implementation sections.

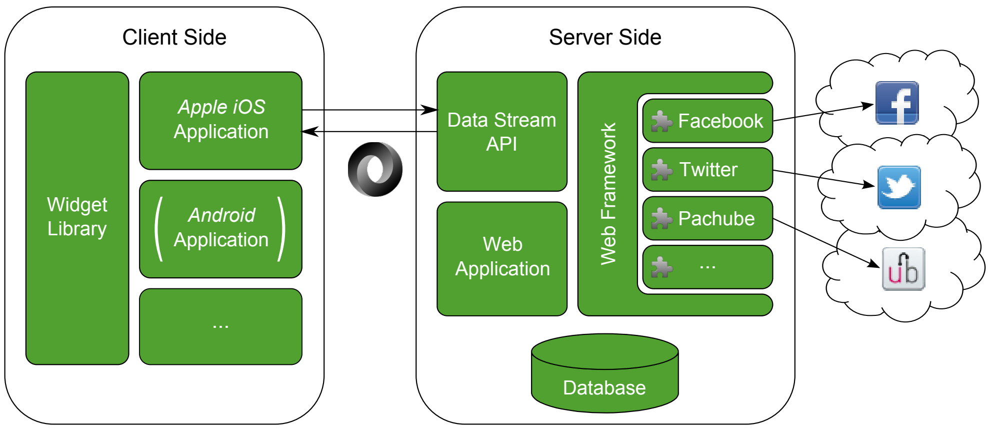


Figure 6.2: Internal architecture overview.

6.2 Server-Side Implementation

The server component of the Dashboard system was implemented in *Python*, using version 2.6 of the popular interpreted programming language. Among the reasons for this decision was the broad platform support and the availability of helpful online resources and documentation. Most importantly, the language is complemented by an extensive ecosystem of libraries that enable simple integration of various web technologies.

The recommended approach for developing *Python* web applications is to conform to the WSGI specification. This specification defines a standard interface between web servers and web applications or frameworks. As a result, any WSGI compliant application can be deployed on any WSGI compliant server. In this project, the application was locally tested using the *Python* reference implementation and deployed on an *Apache HTTP Server* with *mod_wsgi* for production purposes. In order to facilitate adoption by developers of servers and frameworks, the simplicity of implementing the WSGI specification was a key factor in its design. However, this simplicity does not translate into ease of use for a web application authors. Therefore, it is generally advisable to use middleware components that provide more convenient programming interfaces for application developers.

One important goal was keeping the implementation lightweight and remaining agile to respond to user feedback and changing requirements. The goal was to keep the web interface simple and functional, relying on conventional web programming techniques. For the most part, the API was realized as a thin layer on top of the database. For an API like this, it can be useful to drop down to a lower level of abstraction and work directly with the HTTP requests and responses. Therefore, the decision was made against using a monolithic full-stack framework. While these frameworks can significantly increase convenience for developers, they often accomplish this with considerable amounts of code working behind the scenes. As a result, it can be cumbersome to figure out what is happening on a lower level.

Web micro-frameworks present an interesting alternative to their full-featured counterparts. These libraries deliberately refrain from integrating advanced features like template engines or object relational mappings. Instead, they focus on bare essentials, such as request and response objects for easy access to headers, form data, file uploads and cookies. Furthermore, they provide routing mechanisms for dispatching requests to handler functions. The most prominent representatives of these frameworks have straightforward and modular designs. Generally, it is very simple to plug in advanced functionality whenever needed. After a review of the available options, the decision was made to use *Pesto*[7] together with the simple template engine from *Bottle*[8].

[7]http://www.ollycope.com/software/pesto/
[8]http://bottlepy.org

Database Model

Another choice of technology that was motivated by the desire for rapid and flexible development was the use of a *NoSQL* database. *NoSQL* is an umbrella term for a range of database systems that do not adhere to the prevalent relational database model. They were developed to cater to the unique requirements of everyday web applications, namely the persistent storage and retrieval of semi-structured data [62]. The concrete solution used by this project is *MongoDB*[9], which belongs to the family of document-oriented databases. It is based on the premise of storing information as documents with dynamic schemas, which are represented in a JSON-like format. The database is very easy to integrate with *Python* applications using the official *Python* driver. The main benefit of this technology for this project was that it allowed to gradually evolve the schema according to changing requirements.

In the data model of *MongoDB*, multiple documents are grouped into collections. Heterogeneous documents can be stored inside the same collection and there is no explicit way of specifying a schema. However, this does not mean that collections become arbitrary conglomerations of documents. In practice, most documents in a collection will share the same structure, which is shaped by the object-oriented model of the application. Nevertheless, it is possible to deviate from this structure whenever necessary. This was valuable, since modeling heterogeneous data sources requires some flexibility in terms of data. For example, a stream of photos from Facebook requires the unique identifier of an album, whereas a RSS feed needs an URL. From an object-oriented point of view, this can be modeled using inheritance. A dynamic schema permits a very natural way of storing this information within one collection. In conclusion, while the system allows flexibility, there is a conceptual model of how the data is structured. The following paragraphs describe the main entities and relationships of the Dashboard system.

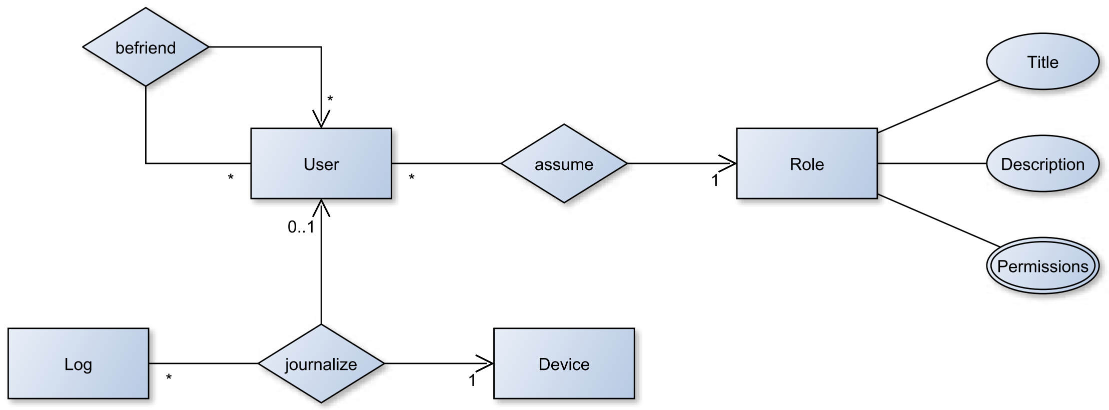


Figure 6.3: Entity-relationship model for user management.

The main entities for managing the users of the Dashboard web service are summarized in figure 6.3. When a user successfully completes *OpenID* authentication, the *OpenID* provider transmits a unique identifier that confirms the user's identity. The web application queries the `User` collection for an existing document with the claimed identifier. If no such document exists, a new `User` entity is created and populated with a minimal set of personal information. This information is automatically retrieved through *OpenID* attribute exchange. User management also provides the basis for the social features of the Dashboard web service by allowing users to form friendships among each other. Friend requests are represented as directed relationships. If both parties befriend one another, a mutual friendship is established, which manifests itself as a bidirectional relationship.

[9]http://www.mongodb.org

The access to restricted features of the Dashboard web service is controlled by `Role` entities. Each `User` assumes a certain `Role`. The exact capabilities of a given `Role` are specified in the multivalued attribute `Permissions`. New users automatically take on a basic default role, which can later be changed by an administrator. As part of the initial deployment of the Dashboard web service, the role system needs to be manually bootstrapped. First, the prospective administrator logs in to provide his *OpenID* identifier and spawn the first `User` entity. Subsequently, it is possible to appoint the first administrator by running a local script on the server. As a result, the newly appointed administrator takes on a `Role` with elevated `Permissions`. All remaining administrative tasks – such as defining and assigning further `Roles` – can now be accomplished through the web interface.

Finally, the Dashboard web service also provides a remote logging interface for gathering empirical data about user interaction with the native tablet application. Individual messages are stored in the database as `Log` entities. At certain times during the application life cycle, there may not be an authenticated user. For example, this is the case when the application launches. Therefore, some messages cannot be traced back directly to a certain `User`. However, all messages are linked to the `Device` that they originated from.

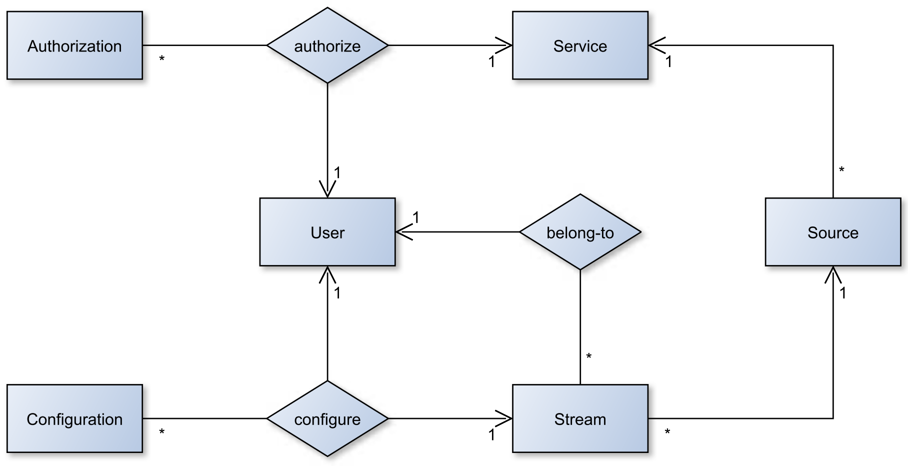


Figure 6.4: Entity-relationship model for services, sources and streams.

The main purpose of the Dashboard web service is data stream management. This functionality is enabled by the entities in figure 6.4. In particular, the very core of the system is represented by `Services`, `Sources` and `Streams`. These entities are arranged in a hierarchical tree structure. Any number of `Streams` can originate from a single `Source` and any number of `Sources` can belong to a single `Service`. The `Stream` entities stand out, as they are owned by the `User` who created them.

Some `Services` require `Authorization` by `Users`. Whenever sensitive personal information is involved, users need ways to control access to their data. The exact procedure for granting authorization depends on the particular service. In most cases, the Dashboard system eventually receives an access token, which is stored in the `Authorization` collection. Similarly, some `Streams` require `Configuration` by `Users`. If additional parameters are required in order to query data, they have to be provided as part of the configuration. Generally, users input these values in a separate step during `Stream` creation. The results are stored as a document in the `Configuration` collection. The `Authorization` and `Configuration` entities function as generic key-value storage, since the concrete data depends on the specific case in question. The implementations are provided by plugins, which will be introduced in the upcoming sections.

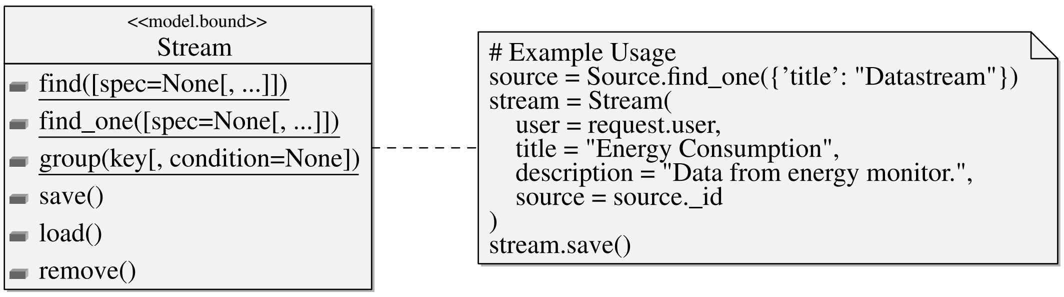
Web Framework

The use of the *Pesto* web micro-framework together with *MongoDB* for document-oriented storage greatly simplified the development of a web application. As development progressed, code was refactored to enable better reuse and several convenience functions were added. Without much effort, some features started to resemble those found in full-stack web frameworks. In particular, the application implements the *Model-View-Controller* pattern in a way that is similar to many popular web development solutions. The following paragraphs aim to provide brief documentation, which may feel familiar to those accustomed to frameworks like *Ruby on Rails*[10].

```
from framework.mongodb import model

from model.user import User
from model.source import Source

@model.bind(collection='streams')
class Stream(model.AttrModel):
    def __init__(self, *args, **kwargs):
        assert kwargs.has_key('user')

        kwargs.setdefault('title', "Untitled")
        kwargs.setdefault('description', "No description.")
        kwargs.setdefault('source', None)

        super(Stream, self).__init__(*args, **kwargs)

    @model.reference(User)
    def user(self): pass

    @model.reference(Source)
    def source(self): pass
```

Listing 6.1: A model in the web application.

By default, the *MongoDB* driver exposes the JSON-like documents from the database as plain *Python* dictionaries. In a complex application with multiple interacting entities, it helps to create specialized data structures that make the underlying model more explicit. In the Dashboard system, this was accomplished using thin wrappers around dictionary objects. An example of this technique is provided in listing 6.1. The code shows a slightly simplified version of the class used to represent `Stream` entities. The `model.AttrModel` base class provides access to values in the dictionary through the dot operator, which is typically used for class member access. This syntax is arguably cleaner and requires less typing. The constructor populates the dictionary with default values and asserts that required fields are present.



```
# Example Usage
source = Source.find_one({'title': "Datastream"})
stream = Stream(
    user = request.user,
    title = "Energy Consumption",
    description = "Data from energy monitor.",
    source = source._id
)
stream.save()
```

Figure 6.5: Methods for retrieving and storing documents.

[10] http://rubyonrails.org

Several *Python* decorators are provided to streamline the code and make it more readable. The `model.bind` decorator associates the class with a certain collection in the database. It injects methods for storing and retrieving objects in the collection, which are listed in figure 6.5. The `model.reference` decorator is used to generate special properties, which are based on the *Python* descriptor protocol. These properties automatically resolve external references to other collections in the database. Therefore, the member access syntax `stream.source` yields an instance of class `Source`, rather than just a foreign key. If the developer wishes to work directly with the object identifiers, he can do so by using the item access syntax `stream['source']`.

```
from pesto import Response

from framework import dispatch, template, utilities

from model.stream import Stream

@dispatch.match('/', 'GET')
@template.render('streams')
def root(request):
    streams = Stream.find({'user': request.user._id})

    return locals()

@dispatch.match('/create', 'GET', 'POST')
@template.render('stream')
def create(request):
    stream = Stream(user=request.user._id)

    if request.request_method == 'POST':
        utilities.convert_objectid(request.form, 'source')

        stream.update(request.form.items())
        stream.save()

        return Response.redirect(request.make_uri(path='/'))

    return locals()

@dispatch.match('/delete/<_id:oid>', 'GET')
def delete(request, _id):
    stream = Stream.find_one(_id)

    if stream is None:
        return Response.not_found()

    if stream['user'] != request.user._id:
        return Response.forbidden()

    stream.remove()

    return Response.redirect(request.make_uri(path='/'))
```

Listing 6.2: A controller in the web application.

The controller is responsible for handling the incoming requests and returning a response to the user. In the general case, it triggers actions in the model and passes the results to an appropriate view. In the Dashboard system, controllers are simple *Python* modules that contain request handlers, as illustrated by listing 6.2. A request handler receives a request object as one of its parameters and returns a response object, both of which are provided by *Pesto*. The actual creation of the response object is often delegated to the `template.render` decorator, which specifies the designated view for the response.

URL	Method	Description
`/<controller>/`	`GET`	Get a list of entities.
`/<controller>/create`	`GET`	Get a form to create an entity.
	`POST`	Create an entity.
`/<controller>/update/<_id:oid>`	`GET`	Get a form to update an entity.
	`POST`	Update an entity.
`/<controller>/delete/<_id:oid>`	`GET`	Delete an entity.

Table 6.1: Main URL patterns in the web application.

A typical controller for managing a collection contains functions to create, update and delete entities. For brevity, the update function was omitted in listing 6.2. By convention, the Dashboard system provides these actions under a standard set of URL patterns, summarized in table 6.1. Apart from these standard actions, there may be additional ones whenever it makes sense. For example, the controller managing user entities has a search action, which allows to search the collection of users by name. Actions that involve form submission are implemented using the *Post/Redirect/Get* pattern, as demonstrated by the create action in listing 6.2. In this example, users are automatically redirected to the list after creating a new entity. This is done to avoid problems when users try refresh the website, which would otherwise cause form resubmission.

```
%rebase layout title='Stream', request=request

%from model.source import Source

<form id="stream" accept-charset="UTF-8" action="" method="POST">
<fieldset>

<label for="title">Title:</label>
<input type="text" name="title" value="{{stream.title}}"/><br/>

<label for="description">Description:</label>
<input type="text" name="description" value="{{stream.description}}"/><br/>

<label for="source">Source:</label>
<select name="source">
%for source in Source.find():
%if stream['source'] == source._id:
<option value="{{source._id}}" selected="selected">{{source.title}}</option>
%else:
<option value="{{source._id}}">{{source.title}}</option>
%end
%end
</select><br/>

<input type="button" value="Apply" onClick="$('#stream').submit();"/>

</fieldset>
</form>
```

Listing 6.3: A view in the web application.

Finally, the view is generated using the simple template engine from *Bottle*, which supports embedding of *Python* code within arbitrarily formatted plain text. This is accomplished using special escape characters and whitespace-agnostic syntax. The engine supplies a `rebase` statement, which inserts the current fragment into a base template containing the HTML layout frame. This feature is used in listing 6.3, which provides a view for creating or updating stream entities.

Web Application

The individual controller modules described in the previous section are organized in *Python* packages, which form a directory structure on the server. They are loaded using a custom module import mechanism, which recursively descends into subdirectories and loads any *Python* modules that it finds. This is different from the standard import mechanism, which only loads the requested module and submodules need to be imported separately on demand. The custom loader automatically adjusts the dispatcher to prepend the path of directories that it descended into. This means that the individual request handlers are registered in the composite web application under routes corresponding to the relative path from the root directory. Once all modules have been processed, the loader returns a ready-to-use WSGI compliant web application.

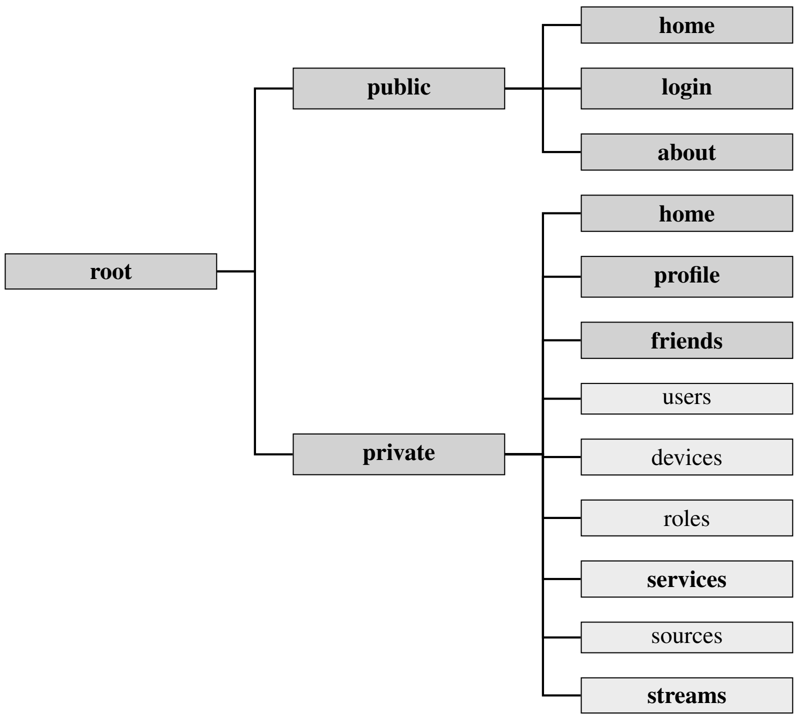


Figure 6.6: Overall web application structure.

The Dashboard web service currently consists out of the modules in figure 6.6. Each node represents a separate controller module, which can contain any number of request handlers that perform various actions. As outlined before, individual actions are accessed via their corresponding URL patterns. In general, the web interface uses the following URL structure:

```
http://<server>/<controller>/<action>?<query_string>
```

For example, let us consider the case of a user wishing to create a new stream. This is performed using the action `create` in the controller `root.private.streams`. The production version is currently deployed on the server `dashboard.urbaninformatics.net`. Therefore, the user should be directed to the following URL:

```
http://dashboard.urbaninformatics.net/private/streams/create
```

A session cookie is automatically created when users visit the website. Users are redirected to the `public` or `private` sections, depending on whether they are authenticated in the current session. The `login` module contains the *OpenID* consumer implementation, allowing users to authenticate with existing *OpenID* accounts. The sections printed in bold can be accessed by all users, whereas the normal ones are only available to moderators and administrators.

The lighter nodes in figure 6.6 represent controllers that serve the purpose of managing certain entities. For example, the `users`, `devices` and `roles` modules correspond directly to the entities for user management. Likewise, the `services`, `sources` and `streams` provide access to the entities for data stream management. The darker nodes symbolize controllers that are geared towards more specific tasks and integrate different parts of the system. For example, the `home` module in the `private` section provides an overview that combines all of the essential end-user functionality. As the web interface matures, the lighter sections become less relevant. Already now, the `services` and `streams` are only available to users for legacy reasons. Eventually, they will disappear from the website, since the `home` module provides a superior interface.

Data Stream API

One of the main technical contributions of this project is the universal data stream API. It defines a single interface for managing and accessing heterogeneous data streams. The implementation follows an approach that is very similar to the web interface. In fact, much of the code is shared, including all data structures for representing the model. The main difference to the web interface is that the API returns responses in a machine readable format, rather than HTML. The controller modules for handling requests are located in a separate directory on the server. Internally, the directory structure closely resembles that of the previously discussed web application. In general, the API uses the following URL structure:

```
http://<server>/api/<controller>/<action>.<format>?<query_string>
```

The main differences are the addition of the `api` directory and the format specifier. Currently, the API only supports the common JSON format, which is a perfect fit for the JSON-like documents in *MongoDB*. The predominant use of this format maintains a degree of consistency throughout the API. It is convenient to use and easy to parse using standard library functions in *Python*. Nevertheless, other formats like XML could easily be added, should there be sufficient demand.

The API responses take on the typical JSON structure, characterized by unordered sets of name/value pairs and ordered lists of values. The top-level message object contains a header and an optional body. The header is stored in the `meta` field of the message object. It contains meta information, such as status codes for signaling errors. If no errors occurred while processing the request, the actual response payload can be found in the `data` field. The nesting allows to provide rich meta information in the header, like descriptive error messages. This information is usually handled by a separate layer in the system dealing directly with the API requests. If the status indicates success, the payload is extracted and passed on. Alternatively, it would have been possible to send the payload directly and use existing HTTP status codes for communicating errors. Both approaches have their proponents.

URL	Parameters	Description
`/private/services.json`	—	Get a list of service entities.
`/private/services/<_id:oid>.json`	—	Get a specific service entity.
`/private/sources.json`	—	Get a list of source entities.
`/private/sources/<_id:oid>.json`	—	Get a specific source entity.
`/private/streams.json`	—	Get a list of stream entities.
`/private/streams/<_id:oid>.json`	—	Get a specific stream entity.
`/private/streams/search.json`	`signature`	Find matching stream entities.
`/private/streams/query/<_id:oid>.json`	`statement`	Query a specific stream entity.

Table 6.2: Main data stream API methods.

An important goal in the design of the API was to keep it simple, making it easy for developers to implement. The main methods of the API are summarized in table 6.2. They can be grouped into three separate areas of responsibility. First, a set of RESTful methods for managing the key entities of the Dashboard web service. Second, a way to discover data streams that match certain requirements based on their signature. Third, a method to query data streams in order to retrieve their contents.

Managing Data Streams This part of the API closely follows the principles of RESTful web service design. The server simply manages collections of resources. Every resource on the server is addressed through a unique identifier. There is no additional logic performed on the server-side. All of the intelligence is implemented on the client-side. The clients use different HTTP methods to access and modify the collections of resources. Currently, the only existing client based on the data stream API is the native tablet application, which requires very limited functionality in this particular area. It only retrieves resources using the `GET` method. The actual data stream management is performed through the web interface. However, it would be easy to implement clients that provide their own user interfaces for creating, updating and deleting streams.

URL: `.../private/streams/4fbd191af4f9da6977000000.json`

```
{
  "meta": {"status": 0, "message": "OK"},
  "data": {
    "_id": "4fbd191af4f9da6977000000",
    "title": "Melbourne",
    "description": "Weather forecast for Melbourne.",
    "source": "4f458d90f4f9da2ffd000000",
    "user": "4f4de8d6f4f9da2ffd000003"
  }
}
```

Listing 6.4: Response containing a stream entity.

Listing 6.4 shows the result of requesting a single stream entity using the `GET` method. As previously described, the JSON object is nested in a message wrapper. The payload contains the fields prescribed by the data model presented in listing 6.1. Following the RESTful approach, it would be possible to update the stream entity using the same URL and the `POST` method.

Discovering Data Streams The current usage of the API rarely requires to retrieve single stream entities. Instead, the client is primarily concerned with discovering all data streams that are compatible with a certain widget. In order to accomplish this task, the Dashboard system introduces the concept of stream signatures. Essentially, signatures provide a simple type system for streams. A signature describes the data contained in each individual item provided by the stream, including an identifier and a basic data type for each field. Generally, all streams from a single data source share the same signature. The data stream consumer usually requires certain fields to be present within the items queried form the stream. Therefore, a special API request enables the consumer to retrieve a list of streams that match the expected signature.

This approach is reminiscent of the type system in the *Go*[11] programming language. The language abandons traditional type hierarchies and promotes the use of interfaces to provide type safety. There is no need to explicitly specify which interfaces are implemented by a certain type. Instead, any type that contains all methods specified in a given interface automatically conforms to that interface. As a result, instances of a matching type can safely be passed to code that expects this interface.

[11] http://golang.org

Similarly, there is no need to explicitly specify the type of a stream. The signature describes the interface of individual items within the stream. Streams matching a certain signature are guaranteed to provide items that contain the specified fields. This is a flexible system that promotes separation of concerns, because new interfaces can be defined and used without modifying existing code. If a new widget expects items with a very specific set of fields, it merely needs to advertise this requirement in the requested signature. There is no need to introduce new types. Likewise, if a new data source provides streams with items containing rich sets of data, that stream can automatically be used by any widget that consumes a subset of these fields.

URL: `.../private/streams/search.json?signature={"title":"String","image":"String"}`

```
{
  "meta": {"status": 0, "message": "OK"},
  "data": [
    {
      "_id": "4fbd1c16f4f9da274e00000b",
      "title": "Comics",
      "description": "A webcomic of romance, sarcasm, math, and language.",
      "source": "4f549b20f4f9da1c53000007",
      "user": "4f4de8d6f4f9da2ffd000003"

    }, {
      "_id": "4fd583aaf4f9da5d4c000000",
      "title": "Pretty Pictures",
      "description": "This stream has no description.",
      "source": "4f91f02df4f9da15f6000000",
      "user": "4f34a171f4f9da14c8000000"
    }
  ]
}
```

Listing 6.5: Response containing stream entities with matching signature.

Listing 6.5 shows the result of searching for data streams that match a certain signature. The required signature is specified as a parameter in the query string. The parameter value is provided in JSON format, as an unordered set of name/value pairs. Each pair represents a required field, where the name is a field identifier and the value is the expected data type. The example searches for streams that supply images. In the current implementation, items representing images contain the fields `title` and `image`, which provide a textual description and a URL of the actual image file. It should be noted that the two matching streams originate from different sources and belong to different users. The stream of comics is shared by a friend of the authenticated user. During discovery, all accessible streams are considered as candidates.

This system for discovering data streams works sufficiently well for the purposes of the Dashboard prototype. However, some issues could arise if the Dashboard web service grows, featuring a bigger selection of sources. In that case, discovering streams based on matching signatures can become problematic, because the type system is too permissive. The stream signature only contains the field identifiers and basic data types. There is no consideration of semantics, that is the actual meaning of the values. Sometimes it is desirable to be more specific when searching for certain data streams.

For example, a source of weather forecasts can have the same signature like a source of product reviews, namely a message and a value with data types `String` and `Number`. The weather widget would expect the message to be a description of weather conditions and the value to represent temperature. However, a product review might contain a message with a verdict and a value representing the rating on a numeric scale. Even though the signatures would match, it would make little sense to display product reviews using the weather widget. A quick solution would be to use less generic field names, encoding the semantics in the field identifiers.

In the long run, a cleaner solution might require a richer type system that allows annotation of data types. A simple example would be annotating the `Number` type with physical units. This would reduce the potential for non-sensical matchings of streams and widgets. In particular, this would be useful for widgets that perform calculations on physical quantities, such as the budget widget, which derives kilowatt hours from a time series of energy monitor readings in watts. Going one step further, it would be possible to integrate ontologies – like the one used by the *Open Directory Project*[12] – to associate streams with semantic categories.

Querying Data Streams The ultimate purpose of data streams is to provide data. The retrieval of items from a stream is supported by the query functionality of the API. It presents a unified approach for retrieving information from different services and sources.

URL: `.../private/streams/query/4fbd191af4f9da6977000000.json?statement={"limit":{"count":5}}`

```
{
  "meta": {"status": 0, "message": "OK"},
  "data": {
    "data": [
      ["2012-07-19T00:00:00", "Mostly sunny.", 9.5],
      ["2012-07-20T00:00:00", "Possible shower.", 11.5],
      ["2012-07-21T00:00:00", "Partly cloudy.", 11.5],
      ["2012-07-22T00:00:00", "Sunny.", 10.0],
      ["2012-07-23T00:00:00", "Mostly sunny.", 10.0]
    ],
    "signature": [
      ["time", "Date"], ["message", "String"], ["value", "Number"]
    ]
  }
}
```

Listing 6.6: Response containing a stream query result.

Listing 6.6 shows the result of querying the contents of a data stream. The message payload contains a list of items. Furthermore, each individual item takes on the form of an ordered list of values. The order of values corresponds with the order of fields in the stream signature, which is also included in the response. This enables the mapping of field identifiers to indices into the lists of values.

As common throughout the API, query statements are also specified using the JSON format. To be specific, a statement is a JSON object that may contain three separate clauses, each of which is available under a distinct key. These clauses are `select`, `where` and `limit`. Their functionality is loosely based on the counterparts in LINQ and – to a certain extent – SQL. However, it is important to point out that despite their similarity they do not implement a full-fledged relational algebra. Instead, they offer limited support for useful operations when querying data streams. The `select` and `where` allow to specify functions that are applied locally to individual fields of items. The `select` clause acts like the higher-order function `map`, applying a function that transforms field values. The `where` clause corresponds to the higher-order function `filter`, applying a predicate function that determines whether an item should be included in the result set. Finally, the `limit` clause allows to specify global constraints on the query, such as the desired total `count` of items or the maximum `duration` allowed for processing on the server.

All clauses in a query statement are optional. An empty statement would attempt to retrieve all items from a stream. Generally, this is not a very useful query, because data streams often contain large numbers of items. Conceptually, streams can be viewed as infinite sequences of items. Therefore, the execution of an unrestricted query would never terminate. In practice, the server would implement a timeout to cancel requests that exceed the maximum processing time. In

[12]http://www.dmoz.org

order to prevent requests from being forcibly canceled by the server, the query statements should specify reasonable constraints on the requested set of items.

The API introduces a custom syntax to represent expression trees. It uses JSON objects with a single name/value pair to denote function invocation. The name acts as the unique function identifier, which is always prefixed with the ‘$’ character in order to make it stand out from normal dictionary keys. The corresponding value provides a list of arguments that will be passed to the function. This syntax is inspired by the advanced query operators in *MongoDB*. In this case, a custom expression syntax would most likely be cleaner and more readable. However, using the well-supported JSON format was beneficial, since it eliminated the effort of writing a custom parser. For the sake of faster prototyping, this approach was chosen in favor of a terse syntax.

Expression	**Semantics**
Placeholders	
`'$0'` ⇔ `'$_'`	$(x) \mapsto x$
`'$1'`	$(x,y) \mapsto y$
`'$2'`	$(x,y,z) \mapsto z$
…	
Arithmetic Operators	
`{'$+':[2,1]}`	$() \mapsto 2+1$
`{'$+':['$0',1]}` ⇔ `{'$+':[1]}` ⇔ `{'$+':1}`	$(x) \mapsto x+1$
`{'$+':['$0','$1']}` ⇔ `{'$+':[]}`	$(x,y) \mapsto x+y$
`{'$-':[2,1]}`	$() \mapsto 2-1$
`{'$-':['$0',1]}` ⇔ `{'$-':[1]}` ⇔ `{'$-':1}`	$(x) \mapsto x-1$
`{'$-':['$0','$1']}` ⇔ `{'$-':[]}`	$(x,y) \mapsto x-y$
…	
Logical Operators	
`{'$and':[{'$>':0},{'$<':5}]}`	$(x) \mapsto (x>0) \wedge (x<5)$
`{'$not':{'$or':[{'$<=':0},{'$>=':5}]}}`	$(x) \mapsto \neg((x \leqslant 0) \vee (x \geqslant 5))$
…	

Table 6.3: Exemplary JSON expressions.

Some exemplary expressions are listed in table 6.3. As demonstrated by the examples, function invocations can be nested to create arbitrary expression trees. If an argument list contains fewer arguments than expected by the function, it is padded with placeholders. The resulting expression tree is compiled into a *Python* function with the appropriate semantics.

```
{
  'select': {
    'value': {'$/':4.0}
  },
  'where': {
    'time': {'$and':[{'$>':"2012-07-29"},{'$<':"2012-07-30"}]}
  },
  'limit': {
    'duration': 5.0
  }
}
```

Listing 6.7: Advanced query statement.

Listing 6.7 shows a possible use of the advanced query functionality provided by the API. The listed statement could be applied to a stream of energy monitor readings. Assuming a four person household, the query result could be used to estimate the per-person energy consumption over the course of a specific day. Furthermore, if the query takes more than five seconds to execute on the server, it should abort and return a partial result.

Plugin Mechanism

The Dashboard web service is built around a simple plugin mechanism that allows easy integration with the information ecosystem of the web. Plugins are realized in the form of regular *Python* packages that are deployed on the server. All packages follow the same general structure, as outlined in figure 6.7. The modules inside a package should adhere to a well-defined interface, implementing a number of required functions. Since these modules can contain arbitrary *Python* code, only trusted plugins should be installed.

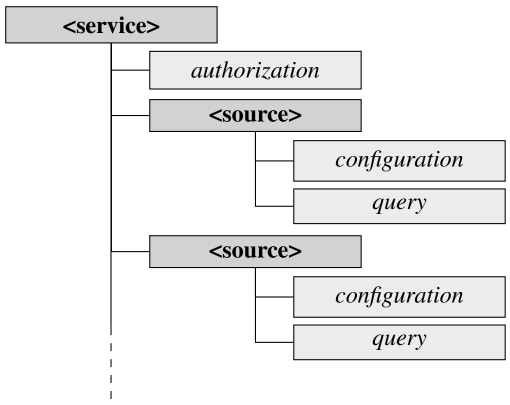


Figure 6.7: Overall plugin structure.

The `authorization` and `configuration` modules extend the Dashboard web interface with custom functionality. They provide specialized implementations of the controllers and views that manage the `Authorization` and `Configuration` entities. For example, if a service requires authorization based on the common *OAuth* standard, the `authorization` module initiates the three-legged token exchange to retrieve an access token. Likewise, if a stream requires additional configuration parameters, the corresponding `configuration` module typically generates suitable HTML forms to collect the data.

Function	Description
authorization	
`create(request, authorization)`	Returns a `Response` that handles creation.
`update(request, authorization)`	Returns a `Response` that handles update.
`delete(request, authorization)`	Returns a `Response` that handles deletion.
configuration	
`create(request, configuration)`	Returns a `Response` that handles creation.
`update(request, configuration)`	Returns a `Response` that handles update.
`delete(request, configuration)`	Returns a `Response` that handles deletion.
query	
`create(authorization, configuration)`	Returns a `Queryable` object.

Table 6.4: Required functions for plugin modules.

The `query` module is the central component of a plugin. Its responsibility is to supply adapter objects that support the retrieval information from heterogeneous data sources in a uniform manner. The `query` module only contains a single factory function which creates objects conforming to the `Queryable` interface. The only method prescribed by this interface is `execute`, which takes a compiled query statement and returns the query result. The Dashboard web service includes several base classes that simplify the implementation of the `Queryable` interface, which are illustrated in the following example.

For the purpose of illustration, let us consider extending the Dashboard web service with a simple source that provides natural numbers. Listing 6.8 shows the creation of the `Queryable` stream object in the factory function of the `query` module. The classes utilize the dependency injection pattern, assembling the final object by passing its dependencies into the constructor.

```
import json

from plugin import base

from . import NumberSource

def create(authorization, configuration):
    service = base.Service(authorization)
    source = NumberSource(service)
    stream = base.Stream(source, configuration)

    return base.Stream(source)
```

Listing 6.8: Basic `query` module of an example plugin.

All that is needed in order to complete this simple example is to create the `NumberSource`. The skeleton provided by the `plugin.base` module minimizes the complexity of this task, allowing the implementation to take the form of the simple generator function in listing 6.9. The `execute` method in the `Stream` class continuously retrieves items from the `NumberSource` and processes them according to the query statement.

```
import sys

MIN, MAX = 0, sys.maxint

from plugin import base, signature

@signature.sign(('value', signature.Number))
class NumberSource(base.Source):
    def iterate(self, configuration, where):
        for value in xrange(MIN, MAX):
            yield {'value': value}
```

Listing 6.9: Naive implementation of a `NumberSource`.

However, there is one mayor problem with this implementation: It is very inefficient, because every item is processed in isolation. There is no intelligence about the underlying nature of the data source. In particular, the naive solution does not exploit the order of items or the ability to skip directly to specific items. This becomes apparent when attempting to select certain intervals of numbers with the help of the `where` clause. In practice, many data sources are indexable by certain fields, which makes it possible to quickly retrieve individual items or item ranges. However, this requires knowledge about the specific data source. Therefore, such optimizations have to be performed in the concrete implementation. In order to make it easy for developers to apply these optimizations, the `jsonscript` module provides convenience functions for analyzing the expression trees in the `where` clause.

```
import sys

MIN, MAX = 0, sys.maxint

from plugin import base, signature

import jsonscript

@signature.sign(('value', signature.Number))
class NumberSource(base.Source):
    def iterate(self, configuration, where):
        interval = jsonscript.interval(where.get('value'))
        for bounds in interval:
            min = bounds.min.value or MIN
            max = bounds.max.value or MAX
            for value in xrange(min, max):
                yield {'value': value}
```

Listing 6.10: Improved implementation of a `NumberSource`.

Listing 6.8 shows an improved implementation of the `NumberSource`. This version checks the `where` clause for expressions that denote ranges. These expressions are represented as intervals, which can be combined according to interval arithmetic. The expression tree is traversed and manipulated according to the theorems of boolean algebra. The result is a finite union of possibly open bounds. The `NumberSource` only yields items that have values lying within each of these bounds.

6.3 Client-Side Implementation

The client component of the Dashboard system was realized as a native tablet application for *Apple iPad* devices. As customary on the *Apple iOS* platform, the majority of the development work was carried out with the *Objective-C* programming language. To be exact, the project relies on the *Objective-C++* dialect, which allows the combined use of *Objective-C* and *C++* syntax. This enabled certain parts of the application to be implemented in pure *C++* code. The main motivation for this approach was to keep selected components portable. Platform independence is preferable from an extensibility point-of-view, because it increases the potential reach of the software, encouraging other developers to contribute regardless of their target platform.

Furthermore, during early development, the target platform for the Dashboard client was not set in stone. There was some discussion among the team at the Urban Informatics Research Lab whether *Android* devices would be a better match for this project. Eventually, *Apple iOS* was favored due to the better availability of hardware, which allowed to maximize the potential audience of the user study. In the meantime, *Cairo*[13] and *OpenGL*[14] were chosen to allow early prototyping of Dashboard widgets and to ensure that they would be portable to the final platform once a decision was reached. A platform independent widget library would greatly increase the possibilities for further development and reuse. Existing widgets could be used in other software and on other devices. Ideally, developers on other platforms would create visualizations tailored to their specific needs and add them to the library. Therefore, much effort was invested to keep the ties with *Apple iOS* to a minimum.

The decision to develop a native application was largely influenced by the requirements of *Information Visualization*, particularly in terms of graphics performance. In addition to that, the application aims to provide a fluid and responsive direct manipulation interface that allows users to compose different widgets on one display. Reducing latency when handling user interaction was a

[13]http://www.cairographics.org
[14]http://www.opengl.org

key priority, especially with regard to touch events. For this purpose, the application adopts a task-based threading model. The use of multithreading allows to offload expensive computations from the main thread. The implementation is based on the cross-platform threading library included in *Boost*[15] – an essential collection of free, peer-reviewed *C++* libraries.

Application Modes and Interaction Flow

In the *Apple iOS* model, applications are represented by an icon on the home screen. At any time, an application is either not running, running or suspended. The operating system can move applications from state to state in response to certain actions. There is only one active application running in the foreground at any given time. In order to bring an application to the foreground, the user taps its icon on the home screen. If the user presses the home button or an interruption occurs, the application is suspended. The Dashboard application conforms to this model. In order to access the Dashboard display, users have to bring the application to the foreground. While running, the application can be in a number of different internal states, as illustrated in figure 6.8. If the Dashboard application was not running, it starts in the initial state. Otherwise, if the application was suspended, it resumes in its previous state.

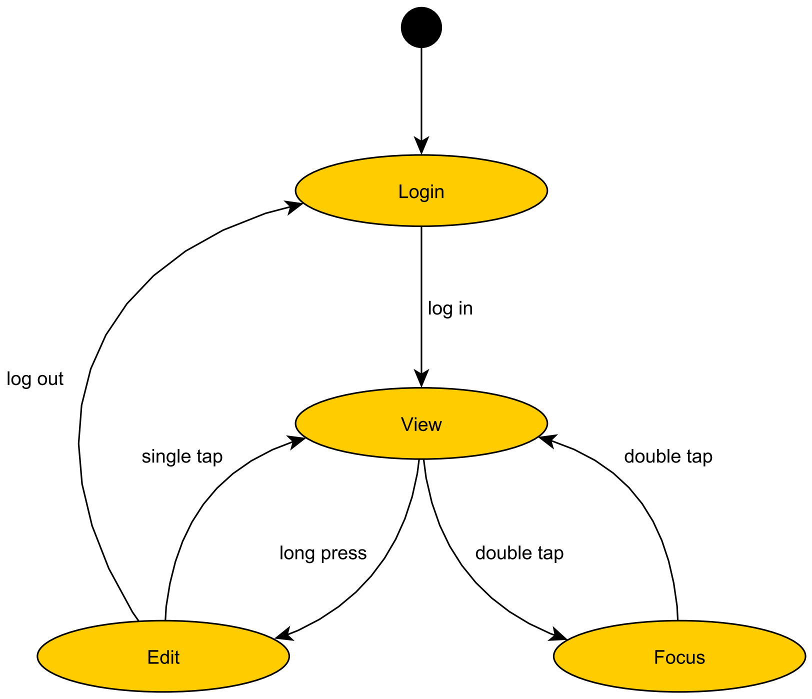


Figure 6.8: Modes of the application.

Initially, when launching the Dashboard application, users have to authenticate themselves. This is required, because the widgets need to have access to their personal data streams. Furthermore, the application utilizes cloud storage for Dashboard configurations. Since user profiles are stored online, users will see the same information display if they log in on different devices. Currently, the application relies on the Dashboard web service for user management. As a result, the interface for the `Login` state consists out of a web view that displays the Dashboard website. The use of an embedded web browser is unavoidable, because the Dashboard web service depends on *OpenID* for authentication. Once the user confirms his identity with the chosen *OpenID* provider, a session is created and the application leaves the `Login` state.

[15]http://www.boost.org

After logging in, the web view disappears, revealing an *OpenGL* view of the Dashboard display. This view acts as the basis for all subsequent modes of operation. At first, the application starts out in `View` mode, which presents an unobstructed overview of all visualization widgets. Recognizing that the primary purpose of the Dashboard display is efficient communication of information, the entire screen space is dedicated exclusively to the widgets. All user interface elements are hidden to avoid visual clutter and distraction. Therefore, user interaction is carried out with the help of touch gestures, which are detected using custom recognizers that offer improved multi-touch support. In particular, the long press gesture initiates the transition into `Edit` mode, whereas the double tap gesture is used to enter `Focus` mode. The gestures were modeled after existing conventions in prevalent *Apple iOS* components, such as the home screen itself and the bundled *Safari* web browser.

Once the user executes a long press gesture, the application switches into `Edit` mode, which is indicated by the visual cues described in section 5. This mode can be used to change the arrangement and composition of widgets. Two buttons appear, which open subviews that are built from standard user interface components provided by the *UIKit* framework. The `Configuration` subview allows to configure and add widgets to the Dashboard display. Meanwhile, the `About` subview displays information about the application and allows the current authenticated user to log out. Users may return to `View` mode by performing a single tap. Finally, `Focus` mode allows users to single out individual widgets with a double tap gesture. The view zooms in on the selected widget, making it fill the entire screen. In order to ensure a sharp and appealing presentation, the content is vector scaled. This mode was also intended for interactive visualizations, however the current widgets do not make use of this functionality.

Task-based Multithreading

The Dashboard display incorporates a composition of multiple widgets that visualize information from various data streams. In order to present a real-time view of the data, widgets periodically query data and render updated visual representations. Both operations, updating the model and painting the view, may take up considerable periods of time. Therefore, it would not be acceptable to perform them on the main thread, as they would interfere with the processing of user interface events. Furthermore, carrying out these operations for multiple separate widgets is an embarrassingly parallel problem. Therefore, it makes sense to execute them in parallel via multithreading.

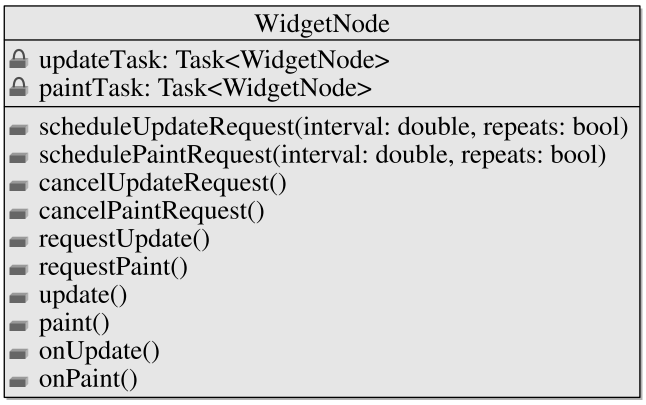

WidgetNode
updateTask: Task<WidgetNode>
paintTask: Task<WidgetNode>
scheduleUpdateRequest(interval: double, repeats: bool)
schedulePaintRequest(interval: double, repeats: bool)
cancelUpdateRequest()
cancelPaintRequest()
requestUpdate()
requestPaint()
update()
paint()
onUpdate()
onPaint()

Figure 6.9: Relevant methods for `update` and `paint` operations.

The two operations are realized as methods in the `WidgetNode` base class (see figure 6.9). The `update` method is mainly responsible for network communication. Besides that, it may also be used for any other long running computations that drive the visualizations. Typically, it is the slower operation, taking up to several seconds to complete. The `paint` method generates the visual representation using vector graphics. For complex widgets, this may require several hundred milliseconds. Therefore, it is not feasible to perform painting in the main rendering loop, which is synchronized to the refresh rate of the screen – usually at 60 hertz. In principle, both operations are completely independent. A widget can change its visual representation without retrieving new data. Similarly, querying data may not necessarily lead to an updated view. In practice, an `update` often triggers the `paint` operation, because new data becomes available.

The Dashboard application utilizes two separate thread pools, each dedicating its worker threads to one of the two operations. While it would be possible to handle `update` and `paint` tasks in the same thread pool, the unequal nature of the tasks made it beneficial to keep them separate, allocating different numbers of worker threads. Upon application launch, the needed worker threads are spawned for each thread pool. The `Task` template generates wrapper classes that hold pointers to an object and a function. This provides a convenient way to invoke a certain member function on a given object instance.

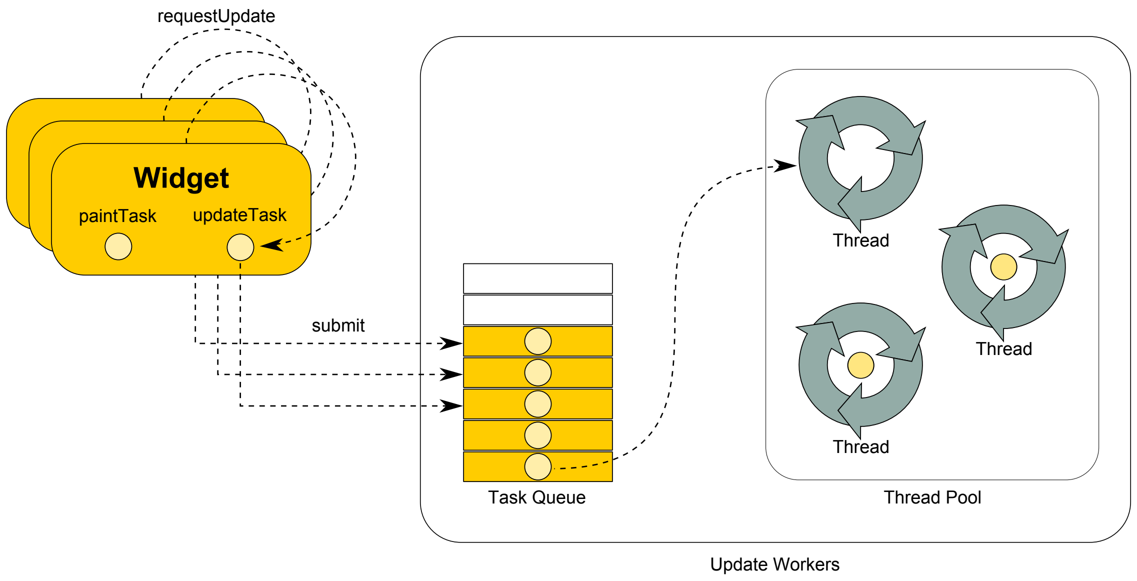


Figure 6.10: Multithreaded updates using a thread pool.

Figure 6.10 shows a schematic representation of performing an `update` operation using the task-based threading model. The `requestUpdate` method is called to signal that a widget needs updating. This causes the `updateTask` to be submitted to the synchronized task queue. As soon as one of the worker threads becomes idle, it attempts to dequeue the next task. If a task is available, the worker immediately starts processing it. Consequently, the `update` method is invoked on the respective widget. After the task completes, it schedules the `onUpdate` method to be executed in the main thread. This can be used to display an indicator in the user interface, which informs users that the widget has retrieved the latest data and is up-to-date.

The `WidgetNode` base class provides convenience methods for scheduling timers, which periodically invoke the `requestUpdate` or `requestPaint` methods. These timers can be stopped with the corresponding cancel methods.

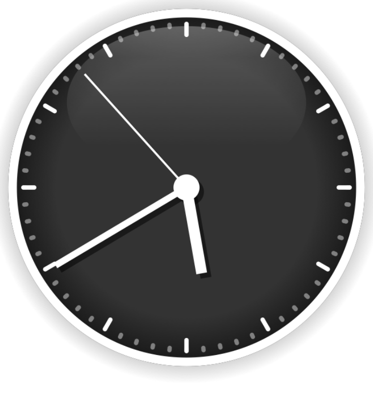
Graphics Rendering

The Dashboard application implements a two-tiered approach for creating the graphical presentation. This was done to address the unique requirements of a composable information display that contains multiple widgets. The task of generating the desired visual output can be broken down into two distinct problem areas. On the one hand, the Dashboard application aims to provide high quality *Information Visualization*. The visualization widgets should be able to make use of advanced 2D drawing techniques, such as anti-aliased paths, shapes and text rendering. On the other hand, the direct manipulation interface requires highly dynamic and interactive graphics. It is important for the user interface to stay responsive, especially when relying on touch input to transform and arrange widgets. Furthermore, it is desirable to support smooth animations, which can be used to guide users and to maintain flow.

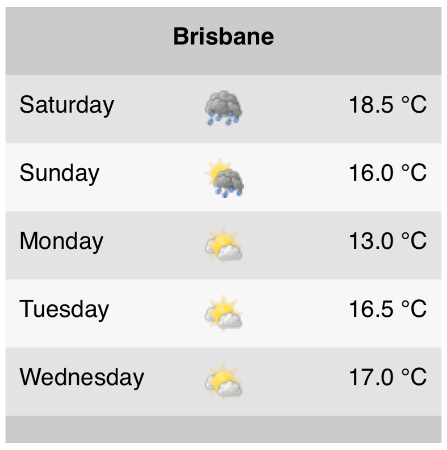
Widget Rendering with *Cairo* The Dashboard widgets paint their visual representations with *Cairo* – a cross-platform library for rendering 2D vector graphics. A key feature of the library is the support for multiple backends. This versatile architecture allows using the *Cairo* API to output graphics to a range of different targets, including several file formats and other rendering systems. It is possible to compile the library for the *Apple iOS* platform with just a few minor modifications. The build relies the *Quartz* backend. This backend maps drawing operations to the *Quartz* API, which is part of the native graphics layer in *Apple iOS*.

As a result, it is possible to develop widgets in a platform independent manner, avoiding *Apple*-specific programming interfaces. However, in some cases, it turned out to be impractical to maintain platform independence. In particular, *Cairo* offers limited support for typesetting and does not perform sophisticated text layouting. If needed by a certain widget, these features were provided by the *Core Text* API. On other platforms, it would be necessary to substitute the corresponding sections with the help of another library like *Pango*[16].

The functionality exposed by the *Cairo* API is a great match for generating visualizations. It is easy to draw complex paths and shapes, applying affine transformations whenever necessary. These basic components can be used to assemble complex charts and diagrams. Since the rendering is vector-based, it is possible to scale the resulting visualizations without loss of clarity. The Dashboard application uses feature when zooming in on a single widget. A scale factor is applied to generate a sharp image with appropriate dimensions. Ultimately, *Cairo* produces raster images, which are uploaded to *OpenGL* textures for display.



(a) Example 1



Brisbane		
Saturday		18.5 °C
Sunday		16.0 °C
Monday		13.0 °C
Tuesday		16.5 °C
Wednesday		17.0 °C

(b) Example 2

Figure 6.11: Rendering of exemplary visualization widgets.

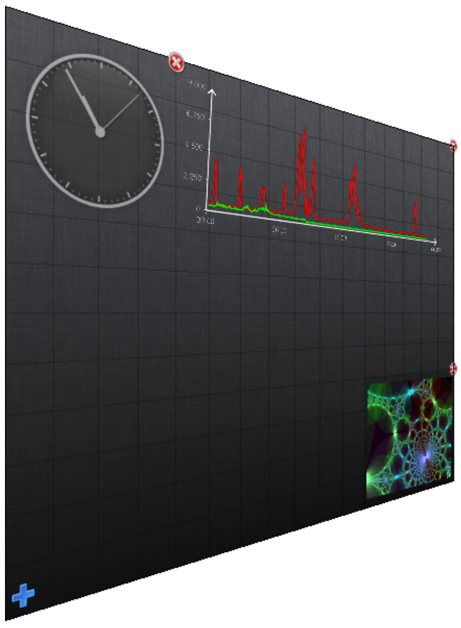
Figure 6.11 depicts two exemplary Dashboard widgets. The widget in figure 6.11a illustrates the stroking and filling of different paths and shapes. It applies radial gradients to simulate shading, improving the visual appearance. The widget in figure 6.11b uses advanced functionality for rendering anti-aliased text and compositing translucent icons.

[16]http://www.pango.org

Display Rendering with *OpenGL* The Dashboard display is rendered using *OpenGL* – a cross-platform library for high performance 3D graphics. The application contains convenience wrappers for interfacing with *OpenGL*, as well as a basic scene graph implementation. The scene graph allows to structure complex scenes with a transformation hierarchy. It is possible to freely position cameras in the fully three dimensional scene, which may use perspective projection to generate realistic views. The Dashboard display makes use of this functionality when zooming in on certain widgets. The camera is translated to an appropriate distance from the widget, making it fill the entire screen.

All elements of the Dashboard interface are rendered using textured rectangles, which can be freely positioned in three dimensional space. However, in practice they are all aligned perfectly flat in the xy-plane. In order to render multiple planar surfaces on the same plane, depth testing needs to be disabled. Rendering with depth testing enabled results in unpleasant z-fighting artifacts. If it ever becomes necessary to integrate the Dashboard interface in a fully three dimensional scene, it would have to be rendered in a separate pass – either beforehand into a texture or at the end with depth testing enabled and depth writes disabled.

The interface elements have an additional z-index to control their rendering order. Each frame, the elements are ordered by their indices and drawn in back-to-front order. This process is illustrated in figure 6.12. Conceptually, elements with the same z-index are grouped into a layer. Among elements within the same layer the rendering order is undefined and could vary between frames. Therefore, elements within the same layer should not overlap, as this could lead to flicker. In practice, it tends to be stable if the scene graph does not change. Currently, the application uses three main layers (see figure 6.12b). The background and grid belong to the bottom layer, whereas widgets usually belong to the middle layer. The only exception are widgets that are actively being transformed by the user, they are temporarily placed in the top layer to appear floating above the passive widgets.


(a) Collapsed for actual use.

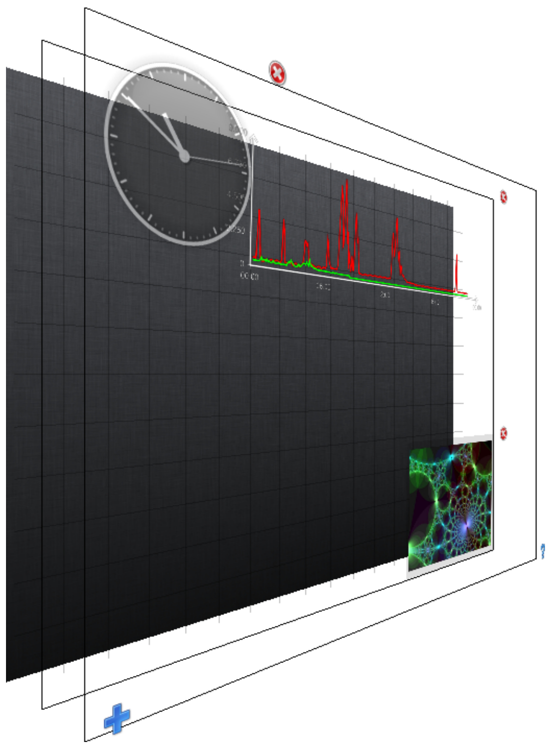

(b) Expanded for demonstration.

Figure 6.12: Rendering with layers collapsed and expanded.

In addition to the scene graph, the graphics component provides a simple system for performing time-based animations. The solution is implemented using blocks, which are the *Objective-C* specific equivalent of closures. Blocks provide a syntax for writing functions inline with the rest of the code, capturing variables from the enclosing scope. The animation system allows to register blocks as callbacks, which are invoked during each frame. When registering a callback, the developer specifies the time interval when the animation should play, as well as looping behavior and optional easing functions. At each invocation, the callbacks receive a single parameter, representing the current progress of the animation. Based on this parameter, it is possible to interpolate various visual properties of any object instance captured by the closure. This animation system is used to smoothly transition between the different modes application, fading interface elements in and out. Furthermore, an *OpenGL* shader-based wiggle animation provides visual motion cues about possible interactions with the widgets, signaling when they can be arranged and modified.

7 Evaluation

This thesis is part of a larger body of research at the Urban Informatics Research Lab, which explores the use of *Ubiquitous Computing* for bringing real-time environmental data into the homes of Queensland residents. The stated goal of this research is to deliver usable and useful prototypes, which offer individual and collective visualizations of ecological impact, as well as opportunities for engagement and collaboration. The shared vision for these technologies is to foster a participatory and sustainable culture of life in Australia. For this purpose, the Urban Informatics Research Lab has received funding from the Queensland State Government to conduct a three year study. The Dashboard system was one of the prototypes that were evaluated during the study. The evaluation was carried out in coordination with local PhD students, with particular involvement from Richard Medland, whose dissertation aims to inform the development of *Persuasive Technologies* for helping individuals to conserve resources.

The following study design focuses on the application of the Dashboard prototype to the problem of energy conservation. Participants were introduced to the prototype as tool for monitoring and managing their household energy consumption. A major research question was whether the ability to tailor the Dashboard interface to personal needs and preferences would result in a more enjoyable and effective information display. The expectation was that these desirable properties would manifest themselves through the occurrence of regular usage patterns. The study plan proposed to observe the natural use of the system, tracking key measures about different user interactions. This delivered insights about the integration of Dashboard prototype into daily routines and the potential for sustained use. Furthermore, the study provided recommendations for future development.

In order to find potential participants, it was possible to tap into the pool of users that were involved in the overarching study at the Urban Informatics Research Lab. For the most part, they were Brisbane residents recruited by the PhD students conducting related research preceding the Dashboard project. After being accepted into the pool, the users were payed a visit and received energy monitoring hardware, which was provided courtesy of *Smart Now Pty Ltd.* – the Australian distributor of *Current Cost* devices. During the visit – several weeks before the deployment of the Dashboard system – a first interview was held, asking general questions about their lifestyle and environmental attitudes. In addition to that, participants filled out a short survey covering demographic data and basic information about their household.

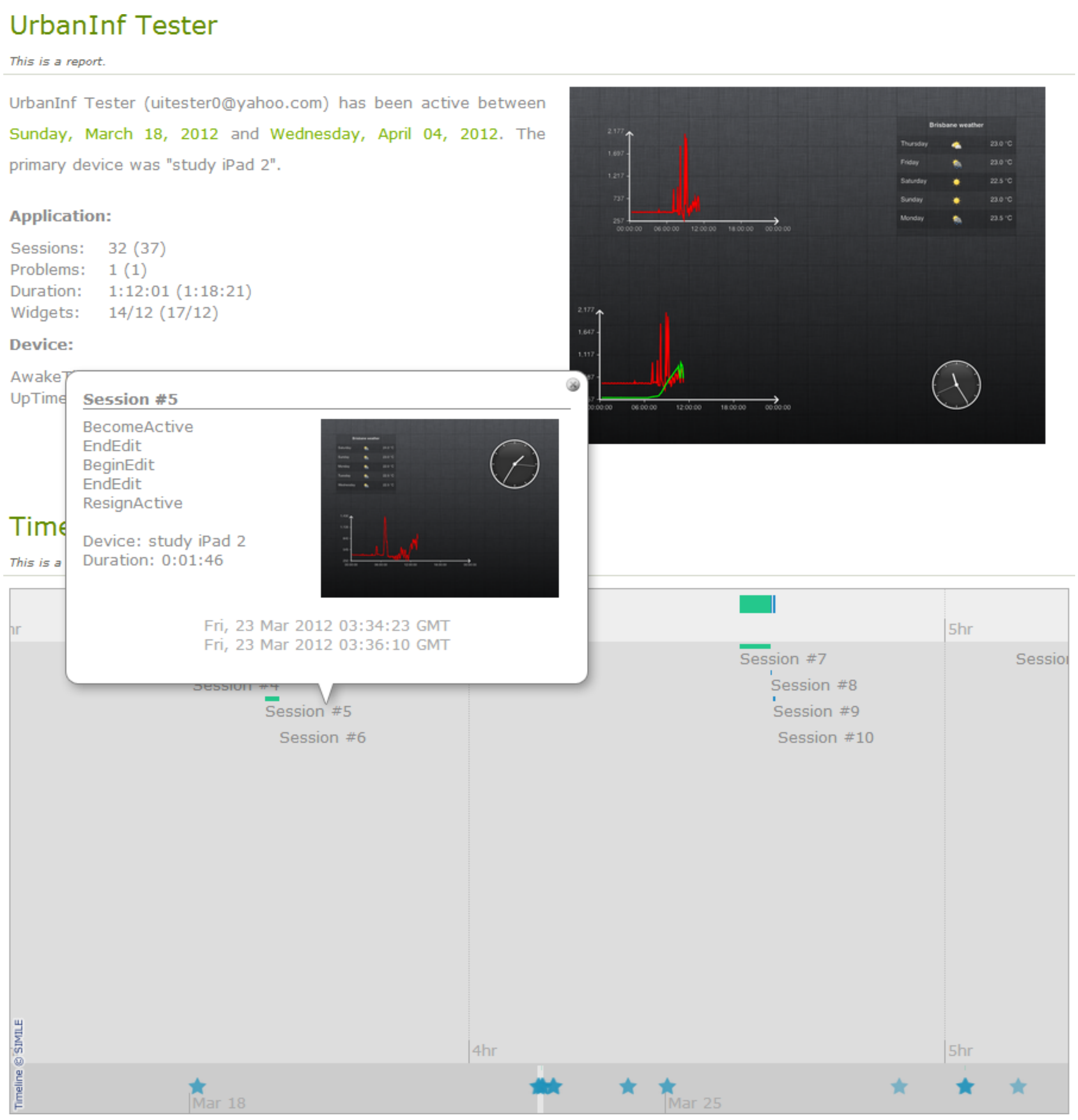
The complete Dashboard study spanned two separate phases, which were carried out sequentially. A time buffer was factored in between the two study runs, allowing to address any pressing issues that would arise in the meantime. The first phase was limited to a smaller number of participants, including several experts to help identify usability issues. It served as a dry run before the second phase, which included a larger group of test users. This study setup allowed to fine-tune the evaluation methods and ensure smooth operation of the Dashboard system. The following sections outline the course of events for each phase and summarize the main findings.

7.1 Study Phase 1

This phase marked the first time that the Dashboard prototype would be used outside of the lab. While the on-site expert review yielded encouraging results, open questions still remained about performance of the system in a real-world environment without supervision. In order to make the Dashboard functionality accessible from the outside world, a public server had to be configured, hosting the production deployment of the web service. Furthermore, *TestFlight* was integrated to allow over-the-air distribution of the native tablet application, as well as remote logging and debugging of potential problems.

Aside from serving as a field test for the different components of the Dashboard system, the first phase allowed to gather early empirical usage data. In addition to this quantitative data, an interview guideline was developed to collect qualitative feedback from the participants. Since this phase included several subject-matter experts in order to assess the usability of the system, the guideline was aligned accordingly. For the second phase, the interview questions were modified to probe deeper, evaluating the core Dashboard concepts. Most importantly, the first phase played an important role for informing the plan for the second phase. Some parts of the original study design had to be revised in response to the observations.

The administrative tools of the Dashboard web service enabled timely tracking of user activity. A detailed usage report was automatically generated for each user (see figure 7.1). The generated reports allowed to verify the correct functioning of the Dashboard application and provided a first glimpse at usage patterns. For example, the integrated timeline often showed bursts of activity, consisting out of multiple sessions carried out in succession.



UrbanInf Tester

This is a report.

UrbanInf Tester (uitester0@yahoo.com) has been active between Sunday, March 18, 2012 and Wednesday, April 04, 2012. The primary device was "study iPad 2".

Application:

Sessions:	32 (37)
Problems:	1 (1)
Duration:	1:12:01 (1:18:21)
Widgets:	14/12 (17/12)

Device:

AwakeT

UpTime

Brisbane weather

Thursday	23.0 °C
Friday	23.0 °C
Saturday	22.5 °C
Sunday	23.0 °C
Monday	23.5 °C

Session #5

BecomeActive
EndEdit
BeginEdit
EndEdit
ResignActive

Device: study iPad 2
Duration: 0:01:46

Fri, 23 Mar 2012 03:34:23 GMT
Fri, 23 Mar 2012 03:36:10 GMT

Time

This is a

Figure 7.1: Usage report provided by Dashboard web service.

Setting

The first phase of the user study was carried out over the course of two weeks. A total of 5 participants were selected for the initial round of evaluation, including 2 female and 3 male. They were aged between 27 and 37 years old (see table 7.1). Unfortunately, in both phases of the study, a small number participants dropped out for various personal reasons, such as intervening business travel arrangements on short notice. As a result, the first iteration of the study was completed by a total of 4 active participants.

	Age	Gender	Living Situation	Housing	Members
#0	36	M	Couple, Kids	Private, Owned	3
#1	27	F	Couple, Flatmates	Private, Rented	3
#2	30	M	Couple, Flatmates	Private, Owned	4
#3	37	M	Couple	Private, Owned	2
#4	32	F	Single, Flatmates	Private, Rented	2

Table 7.1: Participants for study phase one.

In an effort to make it as easy as possible for study participants to use the Dashboard application, pre-configured dummy accounts were created for each user. Primarily, this was done to set up the energy monitoring streams in advance. The configuration process for adding data streams from the *Cosm* service is slightly involved and requires manual input of numerical feed identifiers. Since this task is not very user friendly and demands some technical expertise, it was not meant to be performed by lay users without assistance. Instead, users were given the login credentials to the dummy accounts, which already contained the streams from their personal energy monitor.

In order to be included in the first phase, participants did not need to own compatible tablet devices. Instead, they were outfitted with ready-to-use devices at the beginning of the study. To make this possible, the Apple University Consortium provided a short-term loan of five *Apple iPad 2* kits as part of their seeding equipment program for member universities. Once the accounts were configured and the application was installed, the tablet devices were distributed to the participants. As part of the initial visit, the test users received a brief introduction to the Dashboard system, assisting them in the composition of a first set of widgets. After answering any remaining questions, the participants were left to explore the software at their own convenience. At the end of the two week timespan, they received another visit from Richard Medland, who conducted the post-study interview. Finally, the tablet devices were collected and returned to the Apple University Consortium.

Quantitative Data

Several measures were identified to track different aspects of the interaction with the Dashboard system. The fine-grained raw data about individual actions was processed and accumulated into these measures. In addition to the frequency and duration of use, particular attention was paid to quantify the amount of viewing and editing performed by a certain user. This information was deemed to provide potential insights about how customization affects overall Dashboard usage. Similarly, the collected data included the overall amount of tablet use, providing clues about the extent of integration with the users media consumption habits.

Due to the small sample size, the statistical significance of the quantitative data was limited. Therefore, the primary focus was on the qualitative feedback from the interviews. Nevertheless, a fair amount of quantitative data was collected and analyzed. Over the course of the two weeks, the usage statistics showed a lot of fluctuation between the different participants. In particular, the overall usage of both, the tablet device and the Dashboard application, varied strongly. Table 7.2 provides a high-level overview of the activity by listing the main aggregated statistics for each participant.

	Tablet (Time Active)	**Dashboard** (Time Active)	**Sessions** (View/Edit)	**Widgets** (Created/Deleted)
#0	04:47:59	01:12:02	32 (16/15)	2 (14/12)
#1	04:48:27	00:51:47	23 (14/8)	4 (8/4)
#2	10:28:42	00:26:56	19 (9/10)	2 (8/6)
#3	00:40:26	00:05:39	6 (2/4)	4 (4/0)
#4	00:01:52	00:01:52	1 (0/1)	0 (0/0)

Table 7.2: Usage statistics for study phase one.

In total, the more active users spent roughly between 5 and 10 hours on the provided tablet device. Approximately 5-25% of this time was spent with the Dashboard application. While this could be interpreted as a positive indicator, it is safe to say that in the long term this percentage would likely decrease. A large share of the activity took place during the first and second day of the study. In particular, most of the editing happened towards the beginning, whereas later on the Dashboard application was used for viewing. A considerable portion of time was invested in the initial configuration sessions, which were significantly longer than the subsequent viewing sessions. While it is not possible to draw conclusions about long term usage from this short trial, viewing seemed to occur periodically – daily or every other day – for most participants. Sometimes, there was another spike of activity at the end, which can be explained by the interviews, when participants used the application for demonstration purposes.

The ratio of viewing versus editing was between 50/50 and 30/70, meaning that more than half of the time was spent on customization. Furthermore, the overall tablet usage was fairly low, which can be attributed to the somewhat artificial treatment of the foreign tablet devices, since they did not contain personally relevant applications or data. The interviews showed that neither the tablet nor the Dashboard application were factored in to any daily routines.

The tracked data also contained information about the Dashboard configurations. In the first phase, the majority of compositions was very basic. The most commonly used widgets were the clock, weather forecast and energy graph. This suggested that the creation of new streams via the Dashboard web service was a hurdle for many participants. The feedback confirmed this suspicion and identified possible areas for improvement.

In addition to that, depending on availability, up to three months of energy monitoring data were collected from each participant. However, it was not possible to identify meaningful trends or changes. The interviews proved to be a more valuable resource, providing compelling and critical testimonials about the perceived role of the Dashboard application with regard to monitoring energy consumption.

Qualitative Data

The end of the first phase was marked by the post-study interviews, which were conducted in person at a date that was convenient for the participants. The semi-structured interviews primarily sought to identify usability issues and to explore the use of the Dashboard application. During the interview, the participants were encouraged to provide unstructured feedback and to suggest improvements. Furthermore, they were invited use the Dashboard application for demonstration whenever appropriate. Finally, the interviews also served to assess the impact of the application with regard to energy monitoring, if any. On average, the interviews took about 25-35 minutes, following a comprehensive question base and including follow-up questions. The key findings from the feedback are grouped and outlined in the following paragraphs.

Study Design and Execution The participants highlighted several issues about the overall study design, which were taken into account when planning the second phase. The feedback made it clear that there was a certain learning curve when working with the Dashboard system. Some users felt that they were not given enough instructions or guidance. Still, they eventually became comfortable with the system, even though it took them a bit to familiarize themselves with the functionality. Although the Dashboard application was designed as a generic tool, the participants felt that it was primarily intended for energy monitoring. While this was generally true in the context of the study, it unfortunately led to certain widgets being neglected.

Most participants thought that the *Apple iPad* was an acceptable device for this style of interface, noting that a mobile phone would also be ideal for some of the information, but rather restricted due to its small screen size. However, one participant noted that it felt somewhat artificial to be issued with a tablet device that was not their own. This highlighted an experimental limitation, since the device was not integrated into the workflows of the users.

It seemed logical that this issue would be exacerbated by the pre-configured dummy accounts. However, when asked explicitly about the accounts, participants did not perceive them as much of an issue. Nonetheless, the users were not inclined to grant authorization to the Dashboard web service, stifling the chances of adding personal data to their Dashboard display. One participant stated that he did not participate in social media due to privacy concerns. Another participant noted that he preferred to visit the respective websites as they offer more features.

Feedback on some aspects, such as social features and sharing, was limited due to participants not having used them. This is also a limitation, since it precluded online interactions with other users. As the group was small, the lack of social interaction relates to the isolation of the participants on the Dashboard web service, since none of their friends were using the application. Furthermore, the experts were chosen for their expertise rather than their heavy participation in social media, although this was a desirable attribute.

Application Strengths All participants provided constructive feedback, including positive and negative aspects of the Dashboard system. The common themes among the answers revealed some of the most enjoyable features of the Dashboard application. One of the main perceived strengths was in the visual presentation of the data. Participants felt that the visualizations were an effective way of communicating the information.

In particular, participants favored the widgets for displaying the energy graph and weather forecasts. They praised the convenience of accessing information about their energy consumption from the tablet device. The combination of energy graph and weather was especially interesting, because the participants felt that the two pieces of information were connected. Information about the weather conditions was used to reason about energy consumption, exploring the relationship between the two. One participant had photovoltaic cells mounted on his roof. He complimented the configurability of the graph widget, which allowed him to combine and visualize two data streams – one for energy consumption and one for production – within one widget. Therefore, the Dashboard display became the preferred method for checking this information, at a high level, defaulting to the website for more detailed interrogation.

Several participants pointed out that they liked the direct manipulation using touch input. Moving and arranging widgets considered to be simple and intuitive. The overall user experience on the *Apple iPad* device received a positive verdict from participants. The main criticism was related to the Dashboard website and the need to transition between the web service and the native tablet application.

Application Weaknesses A key objective of the first study phase was to identify potential obstacles and usability issues. The participants were asked to not hold back any criticism, since their feedback would help to improve the Dashboard application for the second phase. The collected answers allowed to examine key aspects of how participants interacted with the system. Overall, when asked about the difficulty of specific tasks, the users stated that most of them where easy, albeit not necessarily convenient.

A major area of complaints was the Dashboard website. For most participants, the configuration and organization of data streams was something of a chore. The use of the website by itself was not very rewarding, as it did not provide any direct, tangible benefit. Furthermore, the website did not clearly communicate the roles of the different entities and how they relate to the widgets on the Dashboard display. The transition between the two different interfaces to conduct certain activities was not well-advertised and sometimes unintuitive.

Another area where the participants felt that the application was lacking, was in the relevance and range of widgets available to them during the trial. Ultimately, the constraints of the project put a limit on the absolute size of the widget library. On the flip side, these comments illustrated the ease with which participants were able to interpret the intended function of the Dashboard application. Rather than being overwhelmed by the information that was present, participants wanted more of it, specifically tailored to their interests.

Participants put a lot of emphasis on getting their information quickly. Expiring sessions, prompting users to log in again, as well as other loading times due to communication with server were criticized. The expectations for getting information instantly were quite high, meaning that speed is an important factor for any application aiming to deliver timely information. Additionally, one participant disliked the required constant network connectivity, suggesting that the last retrieved information should be cached locally.

Some of the touch gestures were not immediately clear to certain users. There was confusion about how to switch between ‘edit’-mode and ‘focus’-mode, probably since the application did not support a direct transition between the two. Scaling widgets was not always considered intuitive, because some users had different expectations for how the content should scale. Also, users expressed the need to restrict the sizes certain widgets could have – or to restrict their aspect ratios.

The picture widget was implemented with photos in mind. However, as something of an extra, a data source of web-comics was created to provide some entertaining content. The comics in general were well-received, but it turned out that the existing slideshow presentation was not very well suited for displaying comics. Users were frustrated when the images changed while they were reading and requested a pause or flip function.

Energy Monitoring Impact The participants were also queried about the perceived impact of the Dashboard application on their personal energy consumption. There was consensus among the participants that – given the context of the study – there were some inherent limitations for using the Dashboard system to motivate energy savings. This was linked to the fact that participants had already received energy monitoring hardware at the beginning of the study, including an in-home energy display and access to a device-specific website with similar graphs. Most participants felt that they had already reached an optimal level of energy efficiency, which still allowed them to maintain their desired lifestyle. Achieving further savings would require them to sacrifice everyday conveniences, which they considered non-negotiable.

Apart from that, some participants expressed their desire for deeper analysis features, stating that the current graph widget was too limited for their needs. The missing functionality included on-the-fly adjustments to the scale of the axes in order to zoom in on certain areas of the graph. Although the graph representation was well-received, participants requested the option to extract hard numbers for instantaneous energy usage or total energy used over a given time. In addition to that, several participants were interested in a widget that would compile periodical energy consumption reports.

Nevertheless, several participants found the Dashboard system useful for supporting their maintenance behaviors. One participant mentioned that frequent checking of the Dashboard display allowed him to accurately track usage spikes. When noticing something unusual, he would analyze the data more closely on the website. Participants were able to generally sketch their energy usage on a typical day, depicting the major spikes due to breakfast, dinner and after hours entertainment. Two of the participants were able to give extremely detailed pictures of their daily usage, providing explanations while sketching. Seeing these familiar patterns of the graph on the Dashboard display reassured users that everything was in working order.

7.2 Study Phase 2

The development leading up to the second phase was informed by the findings of the first study run, addressing many of the issues that had been raised by the initial group of participants. As a result, the final version of the Dashboard prototype offered several improvements relating to usability and also functionality. Likewise, the overall study design was altered to account for the learnings from phase one.

One of the main identified problem areas was the Dashboard website. Several steps were taken to improve the overall user experience of this component. The most dramatic change was the complete redesign of the private overview page, arriving at the final result described in section 5. In particular, the visual grouping was introduced to highlight the hierarchical structure of services, sources and streams. This was done in an effort to clearly communicate the conceptual model of the Dashboard web service. Apart from that, basic functionality was added for querying data stream contents through the web interface, allowing interested users to become accustomed with this concept through experimentation. While still very basic, it represents the first step of adding value to the website by making direct use of the data streams. Finally, the link to the website was featured more prominently within the native tablet application and formatted according to established conventions for displaying hyperlinks.

In response to the most prominent requests from the first group of testers, new data sources and widgets were added. For example, generic RSS feeds were previously overlooked as a potential data source. Their inclusion allowed to cover a wide variety of desired content from diverse websites, including news, bargains, promotions and classifieds. Beyond that, participants expressed their interest in using the Dashboard display to boost personal productivity and organization. Therefore, another plugin was added with initial support for *Google* web services.

The original study design for the second phase included A/B testing between subjects. The designated independent variables were the available sets of widgets and the initial configurations. However, this design was discarded, since the first phase demonstrated the drawbacks of executing such a design with a total of 5 participants per group. If the usage statistics would exhibit similar fluctuations, any results would likely be statistical noise. Besides, with a limited library of widgets it would be counterproductive to restrict access to certain subsets. Instead, the chosen course of action was to reevaluate the final prototype with a more targeted approach, focusing on aspects of the Dashboard system that were unattended for during the first phase.

One goal was to facilitate different quality of Dashboard usage by defining tasks for participants to accomplish as part of the experiment. For instance, the first phase failed to fully evaluate the concept of a rich, personalized information display, because participants were under the impression that the main focus was energy monitoring. The tasks were designed to broaden the perceived scope of the project, encouraging participants to try a wider range of widgets. Furthermore, the tasks promoted the use of social sharing features among participants.

In order to cope with the observed learning curve of the Dashboard system, a written user manual was prepared for the second phase. It contained illustrated descriptions of the relevant system functionality for end-users. Moreover, very brief how-to guides provided step-by-step instructions for adding social media data to the Dashboard display.

Setting

The second phase of the user study took place over the course of four weeks. This time, it was possible to recruit a total of 10 participants for evaluation, which were coincidentally all male. They were aged between 26 and 43 years old (see table 7.3). Two of the included participants had already taken part in the first iteration of the study. Like in the first phase, there were a small number of drop outs, resulting in a total of 8 active participants.

	Age	Gender	Living Situation	Housing	Members
#0	26	M	Single, Family	Private, Owned	9
#1	34	M	Single, Flatmates	Private, Owned	3
#2	26	M	Couple, Flatmates	Private, Rented	3
#3	43	M	Couple, Kids	Private, Owned	4
#4	37	M	Couple	Private, Owned	2
#5	30	M	Couple	Private, Rented	2
#6	34	M	Couple	Private, Owned	2
#7	36	M	Couple, Kids	Private, Owned	3
#8	28	M	Single, Flatmates	Private, Rented	3
#9	26	M	Couple, Flatmates	Private, Rented	3

Table 7.3: Participants for study phase two.

Like in the first round, some of the configuration work was done in advance to ensure a smooth start with the Dashboard application. However, this time the study design abandoned the generic dummy accounts in favor of real accounts to make the interaction with the system feel more personal. Several days ahead of the study, participants received an email, prompting them to visit the Dashboard website and log in with an *OpenID* account of their choice. In order to still be able to set up energy monitoring streams on behalf of other users, the impersonation feature was introduced for administrators of the Dashboard web service. Apart from creating the data streams, a friendship was established between the participants and the study supervisor. As a result, it was possible to use the sharing feature to give participants access to a range of miscellaneous content, such as weather forecasts or news feeds, right from the start.

In this phase, one criteria for the selection of participants was that they would own *Apple iPad* devices. The use of personal devices promised to bring forward a more natural tablet usage. Once their accounts were configured, users could download the Dashboard application directly onto the personal tablet device from the *TestFlight* service. Since the conditions were more restrictive than at the first time, some members of the Urban Informatics Research Lab were included to reach a more substantial sample size. These participants were only considered if they were in a position to give objective, unbiased feedback. The second phase was accompanied by two surveys, which aimed to collect more background information about the individual media usage habits of the participants. At the beginning of the study period, the test subjects were contacted by Richard Medland, who notified them about the commencement of the study and provided a link to the online pre-study survey. Initially, the users were free to explore the software on their own volition. After a couple of days, all participants had familiarized themselves with the basic functions of the Dashboard application. Subsequently, several stimuli were introduced throughout the study period, including specific tasks with how-to guides whenever appropriate. The voluntary tasks included trying out certain widgets and establishing friendships with other participants. During the final visit, participants completed an extensive interview and a short post-study survey. The guideline for the final interview is included in appendix A.

Quantitative Data

The tracking functionality of the Dashboard system was not modified and continued to record the same empirical usage data as in the first phase. However, the analysis of the data was improved and an option to export the data was introduced. This made it possible to process the data in external applications and to generate diagrams.

Even though the sample size doubled with respect to the first run, the results still have to be viewed with some reservation. In order to recruit sample sizes that could provide results with a high level of statistical significance, it would be necessary to release the Dashboard application to the general public. Unfortunately, this was not possible due to the time constraints of the project and the complexity of the *Apple* approval process. As in the first phase, there was a lot of variance in the usage statistics of the different participants. Overall, the amount of activity did not increase since the previous phase, although two of the new subjects seemed to respond particularly well. Table 7.4 provides a high-level overview of the activity by listing the main aggregated statistics for each participant.

	Tablet (Time Active)	**Dashboard** (Time Active)	**Sessions** (View/Edit)	**Widgets** (Created/Deleted)
#0	10 days, 01:21:11	03:50:11	62 (34/24)	9 (18/9)
#1	18:57:06	01:30:16	19 (8/10)	6 (15/9)
#2	2 days, 04:27:11	00:37:19	15 (6/8)	8 (10/2)
#3	22:49:16	00:29:56	11 (6/4)	2 (10/8)
#4	04:09:36	00:20:52	5 (3/2)	4 (11/7)
#5	15:33:07	00:19:06	12 (5/6)	5 (11/6)
#6	9 days, 15:29:49	00:15:41	5 (1/4)	4 (6/2)
#7	5 days, 22:02:50	00:13:53	9 (4/4)	5 (9/4)
#8	00:02:56	00:02:56	1 (0/1)	2 (3/1)
#9	00:00:00	00:00:00	0 (0/0)	0 (0/0)

Table 7.4: Usage statistics for study phase two.

As expected, the percentage of time spent with the Dashboard application was significantly lower than in the first phase, amounting to approximately 1-10% of the overall tablet usage. This was likely due to the more natural use of the personal tablet devices for other activities. The distribution of activity was much more spread out accross the study period, mainly due to the tasks that were presented at different points in time. Peaks of activity coincided with the introduction of external stimuli, making the data unfit to reason about long term usage.

The overall tablet usage varied strongly between participants. For most participants, it was considerably higher than during first phase, even when accounting for the longer study period. This suggests that the interaction with the personal tablet devices was indeed more natural and better integrated into existing workflows. Furthermore, it should be noted that a high amount of tablet usage did not necessarily correlate with a high amount of Dashboard usage. This makes sense, since the main purpose of the Dashboard display is quick information retrieval. The short viewing sessions are usually dwarfed by other media consumption on the tablet device.

As during the first phase, customization took up the majority of the time, with viewing to editing ratios between 50/50 and 10/90. Again, this was mostly due to the fact that configuration sessions were significantly longer than viewing sessions. Averaging across all participants, one viewing took about 2 minutes, whereas editing lasted for approximately 5 minutes. However, these averages are not very representative, because the data series had high standard deviations.

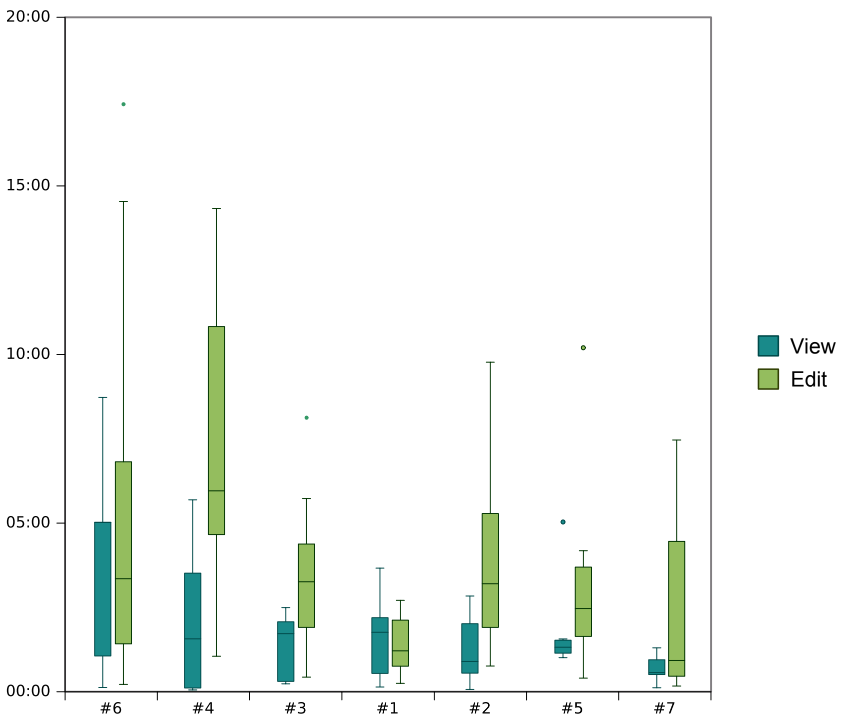


Figure 7.2: Distribution of session durations per participant.

Figure 7.2 shows a box plot of the respective session durations for each active participant. As illustrated by the chart, the values fluctuate between participants. Most 'view'-sessions are concentrated the lower end of the 0-5 minute range, whereas most 'edit'-sessions lie within a 1-10 minute time span with outliers of up to 17 minutes.

Furthermore, the collected data allowed to analyze user activity over the course of a day. The histogram in figure 7.3 displays the total number of sessions at a certain time of day. Overall, the available data generates an uneven distribution, showing increased activity around noon and in the evening. The contributions of individual participants were color-coded in an effort to reveal different usage patterns.

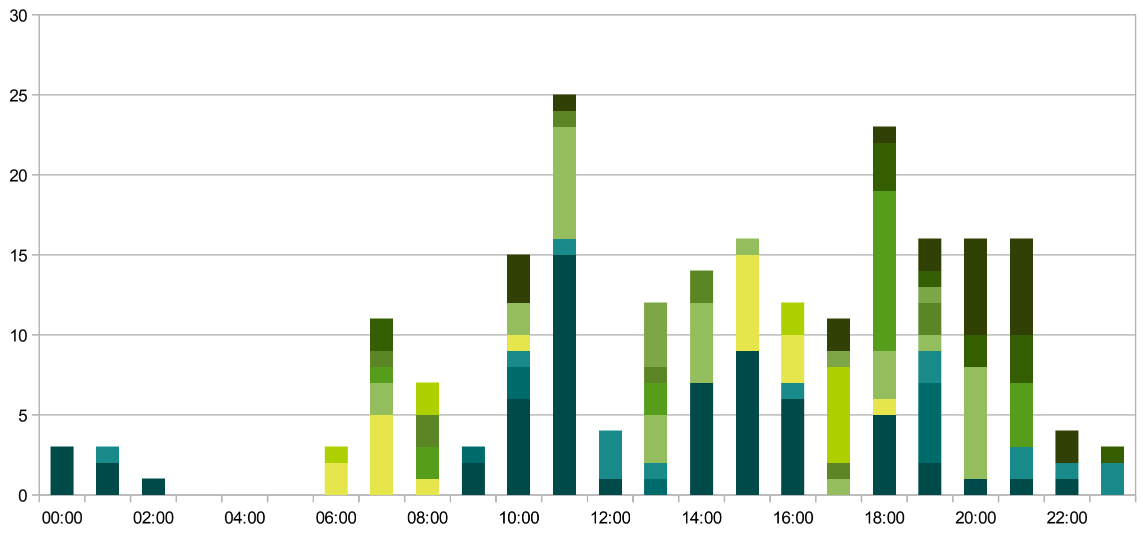


Figure 7.3: Dashboard usage over the course of a day.

The user marked in yellow stands out with clusters of activity in the early morning and in the afternoon. Meanwhile, the users colored in darker shades of green were more active during the evening or at night. However, the coloring also highlights how sensitive the data is, showing major spikes generated by one or two regular users.

The tracked data also confirms that participants created more sophisticated Dashboard configurations than in the first phase, trying out more of the widgets. Above anything else, this can be attributed to the new study design. Interestingly, the most commonly used widgets were still the weather forecast and energy graph. However, they were closely followed by personal messages from social media, as well as the new article widget for viewing RSS feeds.

Qualitative Data

The second phase closed with the post-study interviews, which were again conducted in person at a previously agreed upon date. The semi-structured interviews aimed to gather feedback about core concepts of the Dashboard system, as well as its potential role with regard to energy monitoring. On average, the interviews took about 25-40 minutes, following the interview guideline listed in appendix A. The key findings from the feedback are grouped and outlined in the following paragraphs.

Usage Patterns and Context Participants almost exclusively used the Dashboard application while at home, choosing to take a look during their spare time. One participant also found it useful to check the energy production from his photovoltaic cells remotely, while being out of town. Several participants stated that they were most likely to view the Dashboard display early in the morning or late in the afternoon, after getting home from work. Some participants also described using the application at nighttime. The tablet device was often located in the living room, sometimes kept in reach when watching television. Multiple participants also stated that they would often place their tablet device next to their personal computer. As a result, users would often check the Dashboard display from the sofa in the living room or at the desk in their personal office. One participant also noted using the device at the dining table.

A number of participants appreciated the mobility of the tablet device, allowing it to be carried around the house. However, the usefulness was limited by the rate at which the energy monitoring device transmits measurements to the *Cosm* service. The five minute intervals between updates prohibited users from seeing the immediate effects of their actions, such as switching off lights. Therefore, a truly real-time display would be preferable to most users.

The majority of participants would open the Dashboard application to view a specific widget. In particular, the primary motivation for checking the Dashboard display was related to their wish to review energy consumption. Several participants reported using the Dashboard application and the device-specific website interchangeably, depending on which was more convenient at the time. One participant stated that he enjoyed reading the web comics, sometimes opening the application for this purpose.

Integration with Workflows Only two of the participants were able to say that they regularly used the *Apple iPad*. For the rest, the tablet device was not factored in to any everyday routines or workflows. Many stated that they primarily resort to their desktop or laptop computers for work, spending extensive time with them. One participant noted that keeping several devices in sync was too cumbersome, leading him to favor his laptop computer for most intents and purposes.

With regard to media consumption, one participant described that he commonly used the device to browse the Internet while lying in bed or on a couch. Several participants noted that they only used the tablet device for very specific applications, giving preference to their personal computer for general purpose tasks. The most prominent uses of the *Apple iPad* were reading and playing games. Therefore, its usage was irregular, often during extended periods of spare time.

The feedback made it clear that the Dashboard application fell short of the goal of being integrated with existing workflows. It became evident that supporting a single device would not be enough to accomplish this goal, even if it was as prevalent as the *Apple iPad*. The small group of participants already covered a large spectrum of working habits and device preferences, suggesting numerous alternative platforms for the Dashboard display. In order to meet the diverse expectations, it will be necessary to support multiple devices.

At that stage, the mobility of widgets would become an interesting concept, allowing users to copy or move them across different devices. A possible scenario would include a user who transfers the energy monitoring widget from a stationary in-home display to a smart watch, keeping an eye on energy consumption on-the-go.

Mental Model and Task Flow The extended probing about the Dashboard system revealed potential stumbling blocks with regard to the mental model of the native tablet application. It became obvious that the model-view separation was difficult to grasp for non-technical users. Several participants did not fully understand the connection between data streams on the website and visualization widgets in the application. The two concepts were often believed to be identical, prompting participants to complain about inconsistent terminology.

However, more fundamentally, there was a mismatch between the mental model that was assumed during development and the actual user expectations. The development of the Dashboard application was always centered around visualizations, which led to the choice of visual representation taking precedence over the selection of data streams when configuring widgets. Yet, most users formed a mental model that was centered around the data, primarily thinking about *what* they would like to see and secondarily worrying about *how* it should be presented. This was particularly confusing after the creation of a new data stream on the website, because users were not sure where it would appear in the native tablet application. The application did not communicate clearly which widgets would be able to make use of which data streams.

This should be taken into account in future iterations of the Dashboard application by turning the widget creation task flow on its head. Users should immediately see their data streams in the application. Upon selecting the desired streams, the application should suggest a list of compatible visualizations.

Composability and Customization When asked about the ability to compose their own personal Dashboard display, most participants showed a cautiously optimistic attitude. One participant pointed out that the barriers for going back to customize the display were pretty high once users settled for a certain configuration. At that point, users were reluctant to modify the Dashboard display, unless they discovered something truly new and enticing. This is a reasonable observation, given the human tendency to resist change. Furthermore, it is consistent with the empirical usage statistics. Ideally, users should eventually arrive at a stable configuration that works for them. However, this underlines the importance of guiding users early on, while they are still experimenting with the system.

The basic assumption that individual users have very diverse information needs was confirmed. This was reflected in the varying complexity of different Dashboard configurations. While certain users imagined entire portfolios of different profiles around certain themes, others were only looking for very specific information. In particular, one participant was exclusively interested in energy consumption. His Dashboard display contained several versions of the graph widget, as well as sparklines. While it would be reasonable to expect that the composability had limited value to this participant, the interview answers proved otherwise. Although the options were limited, the participant used multiple widgets to get different perspectives on the data – similar to multiple coordinated views in professional *Information Visualization* systems.

It was easy for most participants to come up with additional information that was missing from their Dashboard display. However, very few reported making an attempt to add this information via the Dashboard web service. Generally, when coming across interesting information, most users had no idea how they might add it. Sometimes this would already be possible using workarounds that require basic technical expertise. In other cases it would be necessary to develop new plugins for the Dashboard web service. In order to make the system accessible to lay users, it will be necessary to streamline the process of collecting data streams from the environment and integrating them into their compositions.

Nevertheless, the different answers highlighted some reservations about the concept of visualizing different types information on a single display. Many participants felt that it was necessary to differentiate between related and unrelated information. The usefulness of combining different pieces of related information was generally undisputed. However, the benefits of viewing unrelated information in combination were less clear. Some participants stated that they liked the exposure to diverse information, whereas others were skeptical about it. Several users expressed the desire to create different profiles, tailored around specific topics of their interest. These profiles were especially popular among the participants who felt the need to organize and group related information. Furthermore, such profiles could help alleviate concerns about limited screen real estate. One participant suggested that the Dashboard application should automatically switch to certain profiles at different times of day. For example, in the morning the display might contain news and weather forecasts, whereas in the evening it could provide a report about the daily energy consumption.

Social Interaction and Sharing About half of the participants completed a voluntary task that invited them to use the social features of the Dashboard web service. Those who took part established friendships among each other and shared their energy consumption data. Overall, the feedback on this aspect was very positive. Several participants described how they compared each others energy consumption and exchanged advice. One participant noted that the sharing aspect was a unique feature of the Dashboard system, which his previous solutions did not provide. Some participants had already manually compared their data with others prior to the introduction of the Dashboard application. They highlighted the convenience of having this capability supported directly by the system.

Participants made suggestions for widgets that would go beyond the simple comparison of energy consumption patterns over fixed periods of time. In particular, one participant expressed his desire for more competitive comparisons, using accumulated totals – normalized to the size of the household – as scores in a contest to reach the lowest energy consumption.

The users who did not engage in social interaction provided various reasons. Two of them experienced technical difficulties with their energy monitoring devices, which limited their ability to participate. One user explained that he was very conscious about his privacy. Therefore, he preferred not to share any personal data, maintaining a clear separation between his social activities and his media consumption.

Energy Monitoring Impact Similar to the first phase, most participants had already gathered experience with energy monitoring, using an in-home energy display and a device-specific website ahead of the deployment of the Dashboard prototype. As a result, they had exhausted most easy to achieve lifestyle adjustments to conserve energy. Several participants noted that convenient access to energy monitoring data did make them more conscious about their consumption, but they had not been able to identify any new savings since the introduction of the Dashboard application. One participant pointed out that he was planning to move into a new home, which discouraged him from making investments to improve energy efficiency at his current place of residence.

Furthermore, two additional factors were identified by participants, which prevented the Dashboard display from being used to identify new energy savings. First, the participants desired to see more detailed breakdowns of the energy usage, possibly per room or even per appliance. Second, the lack of true real-time updates ruled out the possibility of getting instant feedback, for example when turning off lights or appliances. As mentioned before, those were limitations of the current energy monitoring hardware in conjunction with the *Cosm* service. One participant stated that having a more timely energy graph in the Dashboard application would make the dedicated in-home display obsolete.

Like during the first phase, most participants primarily viewed the Dashboard application as another alternative for checking their energy monitoring data. Having this information available through an additional channel increased the convenience of accessing it, which supported existing maintenance behaviors. One participant reported using the *Apple iPad* to check how the current day compared to his typical consumption. At times, this allowed him to diagnose that his energy monitoring device had crashed – a hardware problem affecting the study on several occasions.

The introduction social interaction generated interesting dynamics between participants. One participant reported that he noticed unusual, reoccurring spikes in another participant's energy consumption graph. Together, they investigated the problem and came to the conclusion that it was probably caused by the hot water system. Moreover, the affected participant was persuaded to look into the issue, because the system was active "for quite a long time as well, and in the middle of the night, when you're not using any water. It should keep the heat – that means its loosing the heat". This turned out to be an accurate analysis, because the household turned out to be equipped with a poorly insulated hot water cylinder.

8 Discussion

This section takes a look back at the project and discusses the obtained results. It starts by taking a critical look at the current Dashboard prototype, examining the extent to which it fulfills the original requirements and the areas where it falls short. Subsequently, the study results are revisited, yielding recommendations for increasing the impact of the Dashboard system. Finally, the section concludes with a deliberation of how this concept could develop in the long term, proposing extensions and avenues for future research.

8.1 Critical Reflection

The Dashboard system set out to deliver value for two distinct audiences, namely real-world users and future developers. Therefore, requirements were defined, which would specifically cater to each of these target groups. In particular, the application should offer an appealing and engaging experience for its users, whereas the software architecture should provide a lasting contribution to the developer community. Viewing the problem from both perspectives benefited the project, because the resulting system was tried and tested in a real-world setting rather than being developed in a vacuum. The input collected throughout the user-centered design process helped shape the system functionality, which was in turn reflected in the technical decisions regarding the system architecture. The following paragraphs discuss how the final prototype measures up to the main requirements, which were summarized in section 4.

The system architecture was designed according to best practices in software engineering, using loosely coupled components and well-defined interfaces. To a large extent, it was possible to realize this design within the current prototype, but some compromises had to be made due to the time constraints of the project. The following paragraphs outline the main accomplishments and shortcomings of the current server and client components.

Extensible System Architecture The server-side implementation is already quite mature and refined. In particular, the adopted approach of representing data streams is highly generic and versatile. It is easy to adapt a wide range of data sources from around the web and expose them through a single API. The plugin mechanism places no constraints on how the dynamic data is obtained. For example, it can be collected by scraping HTML pages, parsing RSS feeds or interacting with third party APIs. The *Python* ecosystem provides abundant libraries for processing data in various formats, such as CSV, XML or JSON. Therefore, the Dashboard web service is very easy to extend with new services and data sources. Nevertheless, at the current stage, the server-side implementation also has a few areas where it is still lacking. By far the most pressing issue is the absence of a standard mechanism for authorizing third party applications. Currently, the data stream API relies on the basic session mechanism of the Dashboard web service. This means that users have to authenticate themselves from a web view within the third party application, which generates a session cookie that provides access to the data streams of the authenticated user. This is clearly not an ideal solution, because users have little control over what information third parties can access. Instead, it would make sense to support the *OAuth* standard, allowing users to revoke access and giving them fine gained control over their data streams. Furthermore, the native tablet application is closely tied to the web service with regard to user management. It would make sense to fully separate the two components, implementing the Dashboard application as a truly independent third party client of the Dashboard web service. Finally, it was originally planned to maintain a temporary cache of data stream contents on the server-side. This feature was omitted, since it was not necessary for the scope of the current deployment. However, eventually this functionality should be added in order to reduce latency and support a greater number of simultaneous clients.

Although slightly less mature, the client-side implementation also lays the groundwork for future extension through its widget library. Due to the reliance on cross-platform libraries, a majority of the widget rendering code is easily portable. This opens the door for providing Dashboard displays on other platforms, allowing users to take their favorite widgets onto their preferred devices. However, some concessions had to be made on the client-side in order to save development time. It was not possible to completely separate the widget library code from the rest of the client code as originally intended. The main problem areas were the lack of advanced typesetting in *Cairo* and the handling of user interaction, which requires interfacing with *Apple iOS* specific data structures and thread management. Nevertheless, given sufficient time, it would be possible to untangle the two and provide a truly stand-alone, cross-platform widget library.

Unified API for Heterogeneous Data Streams The data stream API of the Dashboard web service was specifically designed to fulfill this requirement. It exposes heterogeneous data streams in a uniform manner, using a loose type system to communicate the structure of their contents. This structure is defined by so-called signatures, which can be used to discover stream entities that match the requirements of a data stream consumer. Furthermore, the data stream API features a custom query syntax that allows to retrieve data stream contents by executing advanced queries. Most importantly, the data stream API has proven its worth in the Dashboard application, which has been deployed in real-world settings. What remains to be determined is how well this system can scale. The current user base did not generate a noteworthy load on the Dashboard web service. However, once the platform is open to the public, the cost of evaluating complex queries could become significant. Additional performance optimizations may be necessary in order to support multiple third party developers.

The user interaction with the system is split across two different interfaces. Each interface was developed to allow users to perform a specific set of tasks and to do them well. In particular, the design was careful to utilize the strengths of the respective target devices, taking into consideration their form factors and input options. The following paragraphs discuss where the solution succeeded and where it failed.

Seamless Integration with Information Ecosystem The task of data stream management is performed through a web interface. This interface acts as a central hub for establishing connections to the information ecosystem of the web. It allows users to organize and configure data streams from different services. Whenever necessary and desired, users can authorize the brokerage platform to access personal data hosted by external services. While data stream management is essential for the proper working of the Dashboard system, it also turned out to be widely unpopular among users. Even though the underlying technology works well, its value is not explained sufficiently and users do not perceive any immediate benefits. However, this is not the only challenge with respect to seamless integration, which turned out to have two separate dimensions. First, the integration with the information ecosystem is one dimension, which suffers the problem of motivating users to visit the website. Second, the integration with the media consumption habits of individual users turned out to be even more difficult to achieve. Initially, these two aspects were combined into one requirement, because it seemed logical that successful integration with the information ecosystem would automatically lead to better integration with media consumption habits. However, this was clearly not the case, because even after personalizing their Dashboard display with social media, users did not feel incentivised to open the application more often. Instead, they generally preferred to use specialized applications or visit the service directly in order to cater to specific information needs. Finally, for many users, the tablet device was not part of any regular routines or workflows, which further reduced the likelihood of using the Dashboard display. Therefore, it is necessary to concede that the current solution failed to meet the requirement of seamless integration.

Flexible Interface for End-User Customization The composition and visualization of data streams is performed within the native tablet application. It offers a highly flexible interface that allows users to create rich, personalized information displays. The built-in widget library provides different options for visualizing a range of different data streams. Further, a touch-based direct manipulation interface allows users to arrange various widgets side-by-side. Some of the later feedback showed, that the experience could be improved with regard to the widget configuration task flow. Nevertheless, apart from minor usability issues, the interface was well-received and most users were able to customize their Dashboard displays with ease. However, an area where the application is still lacking was in the range and sophistication of available widgets. Due to the limited development time, only a basic set of widgets was created. As a result, some of the more creative ideas for composing data from different streams within one visualization were left unexplored. In particular, the system would benefit from specialized widgets that support social interaction or employ targeted motivational strategies to encourage energy savings. In addition to that, it should be noted that the current Dashboard application is not suited as a tool for professional *Information Visualization*, lacking many basic features of such systems. Instead, the goal of supporting novice users took precedence. The resulting interface allows users to compose visualizations without the need for manually gathering data, writing parsers or converting between formats. Nonetheless, it would make sense to extend the existing capabilities by incorporating more powerful visualization techniques, such as coordination between different visualizations and deeper data analysis support.

Networked Solution for Social Interaction The social interaction is supported through the web interface, which implements a rudimentary friend system. Upon establishing friendships on the brokerage platform, users gain access to each other's data streams, unless they are explicitly marked as private. While this feature worked well, it could potentially be improved with the introduction of more fine grained control over the sharing of streams. Currently, streams are either shared or private, whereas future users may wish share different streams with different groups of friends. Nonetheless, for those who participated in social interaction, the ability to exchange knowledge provided many of the anticipated benefits. However, since the prototype was not released to the general public, the chances of finding acquaintances and friends to interact with were low. This is disadvantageous, since there are strong indicators that social comparison and benchmarking work best with a close circle of friends that share past experiences and common attributes.

8.2 Recommendations

The quasi-experimental study design aimed to investigate correlations between customization of the Dashboard display and repeated, long-term use. In order to quantify the amount of customization, usage statistics were processed to extract meaningful measures, like the number of widgets created as well as the total number and duration of 'edit'-sessions. In order to quantify the longevity of the application, the examination turned to the total number of days during which the Dashboard display had been used. Although these data series did have correlation coefficients upward of 0.8, there were strong doubts about the significance of the result, because the amount of data was limited and the data exhibited questionable characteristics. For example, participants rarely used the application just for viewing, which suggested that they were playing around with it rather than using it to retrieve information regularly. This meant that the correlation between the total number of days and total number of 'edit'-sessions was trivial. Furthermore, the tasks introduced in the second phase distorted the usage statistics. As a result, editing was no longer concentrated at the beginning, but rather spread out across the entire study duration. Therefore, based on the quantitative data, it was not possible to reason whether customization was related to long-term usage.

The qualitative data further called into question if the current approach to customization could provide an incentive for revisiting the application. Participants rarely mentioned their desire to get an overview of information as the motivation for opening the Dashboard application. Instead, they were generally looking for very specific information. Some participants were hesitant to put different pieces of information on the same display, especially if they perceived them to be unrelated. Other participants were not fully aware of their own information needs or the capabilities of the Dashboard system. This meant that some of the anticipated synergy effects did not take place. Furthermore, the desire for specific information often led participants to visit the respective service directly, as it offered more functionality than the widgets on the Dashboard display. It became clear that a high amount of customization – and in particular the mere inclusion of personal information, such as social media – would not warrant sustained use. Instead, in order to entice users to view the display, it has to offer a truly unique experience. The basic concept of a highly flexible, customizable information display has the potential to deliver such an experience. However, in order to make this happen, the compositions have to offer more than just the sum of their parts. Unfortunately, for most users, the current prototype could not accomplish this goal. The following paragraphs outline the main recommendations for bringing the Dashboard display closer to achieving its purpose. They represent the main insights that were gained from the user study.

Guide and Inspire Users Many participants were lost when presented with a blank canvas. Sometimes, they were lacking ideas for what to put on the Dashboard display. At other times, the capabilities of the system were not fully understood. This became evident when participants showed their configurations to each other, often triggering remarks like “I didn’t think of that” or “I didn’t know you could do that – I want that too!”. This illustrated that it is absolutely crucial to guide users, providing them with examples and inspiration. For example, predefined templates would enable users to try different compositions without much effort, showcasing the available capabilities. Extending upon that, users could draw inspiration from each other, if the system allowed them to share and show off their personal Dashboard designs.

Make Widgets Interactive and Proactive Several participants felt that the mere retrieval of information was not gratifying enough, expressing their wish to actively engage with the data. In particular, a recurring theme was the need for deeper, more advanced functionality to analyze the presented data. However, as this was not the focus of the project, the existing widgets were severely limited in interactivity. The design of sophisticated, interactive visualization widgets for use on a tablet device is a challenge in itself. Nevertheless, the interviews showed that finding an adequate solution would add great value to the Dashboard system. Going beyond that, the Dashboard application would further benefit from being proactive. For example, the application should notify the user about new content, such as messages or articles. Likewise, it would make sense to trigger alerts when values in a certain data stream cross a user-defined threshold.

Support Diverse User Preferences Another insight from the study was the wide variety of user habits and preferences, even within the relatively small sample. The participants had highly diverse workflows and used different technologies to support them in their everyday tasks. Beforehand, all indicators suggested that a tablet device would be a good fit for this project. However, in practice, the tablet use was often limited, which precluded the Dashboard display from being integrated with media consumption habits and day-to-day activities. Instead, participants were eager to suggest alternative platforms and devices, which they believed to be better suited for this concept. While it may never be possible to satisfy all users, supporting a broader range of usage habits will be crucial in order to stand a chance at truly seamless integration.

Emphasize Social Interaction The most interesting dynamics with respect to energy conservation arose during the very end of the study, among the subjects that engaged in social interaction. Several participants complimented the ease with which they could exchange and compare energy monitoring information, once they had established friendships on the Dashboard web service. The following exchange of expertise helped to identify problems and put a spotlight on wasteful behaviors. Still, there is much room for improvement, facilitating more frequent and complex social interactions. Without external stimulus, participants were not likely to discover or use the social functionality of the Dashboard system. Again, it is necessary to guide users and prominently advertise these capabilities, making them a central part of the experience. Furthermore, social interaction could be encouraged with specialized widgets. For example, participants showed strong interest in normative and competitive comparisons, delivering more direct assessments and definitive ratings of their environmental performance.

8.3 Future Work

The preceding parts of this section presented several key areas for improving the current Dashboard prototype. From the very beginning, the project was framed as a first step towards exploring the notion of *Street Computing*. It highlighted some of the challenges for building tools that allow people to access and leverage the increasingly complex technological landscape. Reaching the end of this thesis, it is clear that there is no shortage of work left to be done within this research space. Apart from smoothing out the rough edges and implementing the recommendations, future research could explore various new directions.

All of the following projects can build upon the current Dashboard system, bringing the ideas of *Street Computing* closer to fruition. In particular, they embrace core values of its philosophy, assisting the creation of personalized decision support tools through physical and situated end-user composition.

One interesting theme – inspired by user feedback during the study – would revolve around multi platform support with cross-device interaction. As a first step, this would require completely separating the widget library from the remaining client code. Based on that, it would be possible to build specialized Dashboard interfaces for new devices and platforms. At that stage, novel interaction concepts could be explored, such as the ability to move widgets between devices or to interact with different views on separate screens. Apart from enabling completely new usage scenarios, this would offer a whole new level of flexibility, potentially improving integration with media usage habits and workflows.

Another interesting avenue to explore would address one of the main shortfalls of the current prototype. As outlined before, users often lack ideas, initiative or technical know-how to add new data streams on the brokerage platform. Further work on providing simpler, more enjoyable interfaces for data stream management would tremendously benefit the Dashboard system. In particular, it would be worth researching better interface metaphors for the purpose of collecting data streams from virtual and physical environments. Ultimately, adding a data stream to your personal collection should be as easy as placing a bookmark or scanning a code. In the virtual space, this could be enabled through browser plugins that display a simple icon when the user visits a service that provides data streams, much like existing RSS feed icons. This way, adding a new stream should not require more than a single click. In the physical domain, it would be interesting to explore technologies such as QR codes or NFC. This could allow users to interact with data sources in-situ, picking up streams directly from the environment. For example, creating a data stream of bus departure times could be as easy as scanning a code at a bus stop. Such advances would offer exciting new possibilities, while implementing them based on the existing RESTful data stream API would be very simple.

Finally, another extension of the brokerage platform could allow users to program with data streams based on the data flow model. This might be realized through a visual programming interface, taking cues from existing web mashup tools like *Yahoo! Pipes*[17]. Even more cutting-edge research may explore supporting this through concepts like the aforementioned cross-device capabilities or tangible user interfaces. The main contribution from pursuing this direction would lie in the potential for sophisticated processing and mashing up of data stream contents. This would support the goal of building more intelligent systems, that have the ability to extract knowledge from the information contained within data streams.

[17]http://pipes.yahoo.com

9 Conclusion

This thesis chronicles the design, implementation and evaluation of the Dashboard system. It is an early exploration of the *Street Computing* vision, which strives to empower everyday users, allowing them to tap into the layers of digital information that permeate our physical environments. Although efforts exist to make public data accessible and share privately generated data, many sources languish in isolation. Access is often restricted to specialized tools or proprietary technologies. In contrast to this, *Street Computing* aims to foster an open, participatory culture, helping users to help themselves by enabling them to create customized decision support tools. Even though the Dashboard system offers just a faint glimpse of this grand vision, it was a highly compelling and educational project to pursue.

The design of the system was motivated by an extensive review of research literature in the areas of *Ubiquitous Computing*, *Information Visualization* and *Human-Computer Interaction*. Furthermore, it was repeatedly refined through involvement of potential end-users, deploying a diversified repertoire of evaluation methods, including focus group, expert review and field trial. The result incorporates learnings from active research and proposes novel approaches to address open research problems. In particular, it makes the following contributions.

Conceptual Model The Dashboard web service introduces an original way of modeling data streams, driven by the desire to manage and interact with heterogeneous sources. Rather than hardwiring data streams and exposing their contents in a predetermined fashion, a dedicated brokerage platform makes them available as building blocks that can be combined and shared. The platform features a simple API, including mechanisms for discovering streams and querying their data. The Dashboard application is a client of brokerage platform, providing a blank canvas for users to compose and visualize data streams.

System Architecture The Dashboard web service realizes the conceptual model in the form of a lightweight, agile implementation. Instead of pursuing a rigorous formal definition of data streams, the project presents a workable solution that integrates with the existing information ecosystem. The server-side plugin mechanism allows to expose any iterable collection as a data source, placing no constraints on how the dynamic content is obtained. Furthermore, the framework includes convenience methods for working with the custom query language, allowing plugin developers to perform optimizations based on the specific characteristics of the underlying data source. The Dashboard application makes extensive use of cross-platform libraries to support its task-based multithreaded rendering engine. As a result, large portions of the client-side widget library are easily portable to other platforms, promoting reuse of existing widgets and extension with custom ones.

User Interface The Dashboard application provides a flexible interface that allows users to compose rich, personalized information displays. The touch-based interface follows best practices in human interface design, making prominent use of direct manipulation as well as fluid animations that guide users and maintain flow. It is mindful of the weaknesses of tablet devices, avoiding tasks that involve extended typing and form input. The interface supports novice and casual users by automatically matching visual representations and data streams, using the discovery mechanism of Dashboard web service. Detailed feedback from end-users helped to identify usability issues and provided clues about their mental model, revealing further opportunities to improve the user experience.

Evaluation Despite the limited time frame of the project and a relatively small group of study participants, the deployment of the Dashboard system in real-world settings provided valuable insights regarding its potential role for energy monitoring. The results showed that the current prototype was effective for supporting maintenance behaviors of intrinsically motivated users. Certain features of the system, such as the ability to interact with other users and share data streams, successfully brought attention to problems and facilitated the exchange of expertise. However, true integration in everyday routines proved difficult to achieve and a high degree of customization did not necessarily warrant sustained usage. This led to reconsider some of the initial assumptions, resulting in recommendations for increasing the impact on domestic energy consumption.

In its present form, the Dashboard application fulfills a very specific role with regard to energy monitoring. It is effective in the advanced stages of change described by the *Transtheoretical Model*, supporting intrinsically motivated users and making it easier for them to maintain their targeted level of energy consumption. The current prototype is not suited to motivate behavior changes. If a person has already exhausted all feasible energy savings or if they have no intention to conserve energy, providing them with a Dashboard display is not going to have an impact on their energy usage. This confirms the notion that merely informing people about their consumption is insufficient to affect behavior change. However, with ongoing work, the Dashboard system can easily surpass these limitations. Building on what has been accomplished, the following steps promise to reap increasingly tangible rewards, as the results become closer approximations of the *Street Computing* vision. This project merely lays the initial foundation for further research.

List of Abbreviations

API	Application Programming Interface
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LINQ	Language INtegrated Query
NFC	Near Field Communication
QR Code	Quick Response Code
REST	REpresentational State Transfer
RSS	Really Simple Syndication
SQL	Structured Query Language
URL	Uniform Resource Locator
WSGI	Web Server Gateway Interface
WYSIWYG	What You See Is What You Get
XML	eXtensible Markup Language

A Interview Guideline

Usage and Context

- Did you have noteworthy experiences using the system? (Open question. Collect stories.)
- When did you use the system? (Part of daily routine./During spare time. Etc.)
- Where did you use the system? (At home, living room, kitchen, bed. At work. Etc.)
- How often did you use the Dashboard on average? (x times a day/week)
- How long did you use the Dashboard on average? (x minutes)
- What would lead you to use the system more?
  Under what conditions would your usage increase?
- Do you see benefit in the system being customizable/flexible? Would you prefer a specialized system, tailored to specific information (e.g. only energy consumption)? Would your usage differ?
- Were you using the system by yourself or with other household members?
  (Which household members used the system? How did you interact?)
- Was it useful/appropriate to have the Dashboard display on a tablet device?
  (Would you prefer to see this information displayed in a different way?)
- Did you use the iPad in general (not the Dashboard)? Did that influence your Dashboard usage?

System Components

- Were the roles of the Dashboard website and the app clear?
  (Was it clear how they are connected?)
- Did you understand the key concepts: services, streams, widgets?
  (Was it clear how they interact?)
- Do you like the premise of managing streams from different services on a single website?
  What do you like/dislike about it? (Did you find it useful?)
- Do you like the premise of visualizing data from different streams on a single display?
  What do you like/dislike about it? (Did you find it useful?)
- Did you feel confident to make changes to the Dashboard system?

Dashboard Website

- Did you create streams on the Dashboard website? If no, why not?
- Did you authorize services (Twitter, Facebook, Flickr, Foursquare) to access your personal information? If no, why not?
- Did the Dashboard website provide enough services/sources? Did you miss any?
- Which services/sources did you find most useful/interesting?
- Which services/sources did you find least useful/interesting?
- If you had hardware/software to aggregate/track any information, what streams would you add?
  (e.g. water/gas/petrol consumption, personal health/fitness, anything you can imagine)
- Technical barriers aside, if you locate/discover information that you would like to keep up with (e.g. departure times of a bus, stock quotes), would you consider adding it on the Dashboard website?

- Have you come across something that you wanted to add?
  Did you have an idea how you might add it?
- What would be a convenient way for you to manage streams that you are interested in?
- Did you interact with other users online (become friends and share streams)?
  Which streams did/didn't you share? Why?

Dashboard Application

- What was your final configuration of widgets? Can you list the widgets? (From memory!)
- Did the Dashboard app provide enough widgets? Did you miss any?
- Which widgets did you find most useful/interesting?
- Which widgets did you find least useful/interesting?
- Generally, when opening the Dashboard display: What was your intention?
  Looking up a specific piece of information or getting a general overview?
- Generally, when looking at your Dashboard display: Did you focus on a single widget?
  Did you pay attention to a certain set of widgets? Did you scan over all of them?
- Did you find it useful to view multiple streams of information at once?
  (e.g. did you want to check one thing and notice something else/did you see connections etc.)
- Did you find it distracting to view multiple streams of information at once?
  (e.g. did not pay attention to certain widgets etc.)
- What portion of time did you spend viewing (information retrieval) versus editing (configuration/customization)? (Roughly, X% viewing/Y% editing)
- What kind of interactivity should the widgets provide?
  What actions would you like to perform directly with a Dashboard widget?
- Where do you see the difference between Dashboard widgets and apps? What makes sense as a widget/what makes sense as an app? How are widgets different from apps?

Energy Monitoring

- Was the system useful for monitoring energy data? (Was the provided visualization understandable? Was the information sufficient/useful for reasoning about your energy consumption?)
- Can you estimate your last weeks energy consumption?
- Can you sketch your energy usage on a typical day?
- Were you able to identify potential energy savings? (How did the system help?)
- Do you know more about your households energy consumption? (What did you learn?)
- Did the interaction with the system differ from your previous energy monitoring solutions?/How?
- Apart from energy monitoring, what else did you use the Dashboard for?
  Can you imagine other uses?
- Would a stand-alone energy monitoring device be better?

Suggestions

- How would you improve the system? What would you like to see improved?
- Are there any streams/widgets you would like to see added?

List of Figures

List of Listings

List of Tables

Content of the DVD-ROM

Project

- Thesis ... Written Thesis
  - Build ... Print Version
  - Source ... LaTeX Sources
- Implementation ... Practical Part
  - Server ... Web Service
    - WebApplication ... Web Application
    - WebFramework ... Web Framework
  - Client ... Native Application
    - IOSApplication ... *Apple iOS* Application
  - Documentation ... Guides and How-Tos
  - Libraries ... External Libraries
- Evaluation ... Evaluation Part
  - Focus Group ... Focus Group Results
  - Expert Review ... Expert Review Results
  - User Study ... User Study Results
    - Phase 1 ... Study Phase 1 Results
    - Phase 2 ... Study Phase 2 Results
- Presentation ... Presentation Slides
- References ... Quoted References
- Other ... Miscellaneous Documents

References

[1] R. Ackoff. From data to wisdom. *Journal of applied systems analysis*, 16(1):3–9, 1989.

[2] A. Adams and M. Sasse. Privacy in multimedia communications: Protecting users, not just data. In *People and computers XV: interactions without frontiers: joint proceedings of HCI 2001 and IHM 2001*, page 49. Springer Verlag, 2001.

[3] L. Bartram, J. Rodgers, and K. Muise. Chasing the negawatt: Visualization for sustainable living. *Computer Graphics and Applications, IEEE*, 30(3):8–14, 2010.

[4] G. Bell and P. Dourish. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. *Personal and Ubiquitous Computing*, 11(2):133–143, 2007.

[5] B. Berg. *Qualitative research methods for the social sciences*. Allyn and Bacon, 2001.

[6] B. Biörnstad and C. Pautasso. Let it flow: Building mashups with data processing pipelines. In *Service-Oriented Computing-ICSOC 2007 Workshops*, pages 15–28. Springer, 2009.

[7] R. Bosch, C. Stolte, D. Tang, J. Gerth, M. Rosenblum, and P. Hanrahan. Rivet: A flexible environment for computer systems visualization. *ACM SIGGRAPH Computer Graphics*, 34(1):68–73, 2000.

[8] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1121–1128, 2009.

[9] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. Srivastava. Participatory sensing. In *World Sensor Web Workshop*, pages 1–5. Citeseer, 2006.

[10] M. Buschmann and S. Motyka. Energieeffizienz als Schlüssel zur Klima- und Ressourcenschonung? – am Beispiel des Smart Metering. *Wirtschaftsrechtliche Blätter*, 25(1):11–17, 2011.

[11] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, H. Lu, X. Zheng, M. Musolesi, et al. The rise of people-centric sensing. *IEEE Internet Computing*, pages 12–21, 2008.

[12] S. Card, J. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[13] J. Chin, V. Callaghan, and G. Clarke. An end-user programming paradigm for pervasive computing applications. In *ACS/IEEE International Conference on Pervasive Services*, pages 325–328. IEEE, 2006.

[14] J. Corburn. *Street science: community knowledge and environmental health justice*. MIT Press Cambridge, 2005.

[15] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 211–224. ACM, 2008.

[16] D. Cuff, M. Hansen, and J. Kang. Urban sensing: out of the woods. *Communications of the ACM*, 51(3):24–33, 2008.

[17] E. Della Valle, S. Ceri, D. Barbieri, D. Braga, and A. Campi. A first step towards stream reasoning. *Future Internet Symposium, FIS 2008*, 1:72–81, 2009.

[18] P. Dourish. HCI and environmental sustainability: the politics of design and the design of politics. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, pages 1–10. ACM, 2010.

[19] M. Dutta-Bergman. Complementarity in consumption of news types across traditional and new media. *Journal of Broadcasting & Electronic Media*, 48(1):41–60, 2004.

[20] M. Elias and A. Bezerianos. Exploration views: understanding dashboard creation and customization for visualization novices. *Human-Computer Interaction–INTERACT 2011*, pages 274–291, 2011.

[21] European Commission. *EU Action Against Climate Change: Leading Global Action to 2020 and Beyond.* Office for Official Publications of the European Communities, 2009.

[22] S. Few. Data visualization: Past, present, and future. *Perceptual Edge*, 2007.

[23] D. Fisher, S. Drucker, R. Fernandez, and S. Ruble. Visualizations everywhere: a multiplatform infrastructure for linked visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1157, 2010.

[24] D. Foster, S. Lawson, M. Blythe, and P. Cairns. Wattsup?: motivating reductions in domestic energy consumption using social networks. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 178–187. ACM, 2010.

[25] R. Ganti, N. Pham, Y. Tsai, and T. Abdelzaher. Poolview: stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 281–294. ACM, 2008.

[26] L. Grammel, M. Tory, and M. Storey. How information visualization novices construct visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):943–952, 2010.

[27] R. Hague et al. Towards pervasive end-user programming. In *Adjunct Proceedings of UbiComp*, pages 169–170, 2003.

[28] H. He, S. Greenberg, and E. Huang. One size does not fit all: applying the transtheoretical model to energy feedback technology design. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 927–936. ACM, 2010.

[29] J. Heer and M. Agrawala. Software design patterns for information visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):853–860, 2006.

[30] J. Heer, F. van Ham, S. Carpendale, C. Weaver, and P. Isenberg. Creation and collaboration: Engaging new audiences for information visualization. *Information Visualization*, pages 92–133, 2008.

[31] T. T. Hewett, R. Baecker, S. Card, T. Carey, J. Gasen, M. Mantei, G. Perlman, G. Strong, and W. Verplank. *ACM SIGCHI curricula for human-computer interaction*. Association for Computing Machinery, New York, 1992.

[32] B. Jerman-Blazic. Four scenarios for future evolution of the internet. *Technology and Society Magazine, IEEE*, 30(4):39–46, 2011.

[33] C. Karat, J. Karat, and C. Brodie. Why hci research in privacy and security is critical now. *International Journal of Human-Computer Studies*, 63(1-2):1–4, 2005.

[34] T. Kim, H. Hong, and B. Magerko. Designing for persuasion: Toward ambient eco-visualization for awareness. *Persuasive Technology*, pages 106–116, 2010.

[35] T. Kindberg, M. Chalmers, and E. Paulos. Guest editors' introduction: Urban computing. *Pervasive Computing, IEEE*, 6(3):18–20, 2007.

[36] P. King and J. Tester. The landscape of persuasive technologies. *Communications of the ACM*, 42(5):31–38, 1999.

[37] J. Knaak. Vom Smart Meter über Smart Grid zum Smart Home. *Bulletin des SEV/VSE*, 102(9):13, 2011.

[38] R. Krueger. *Moderating focus groups*, volume 4. Sage Publications, Inc, 1997.

[39] V. Kulathumani, M. Sridharan, R. Ramnath, and A. Arora. Weave: an architecture for tailoring urban sensing applications across multiple sensor fabrics. In *Proceedings of the International Workshop on Mobile Devices and Urban Sensing (MODUS'08)*, 2008.

[40] D. Le-Phuoc and M. Hauswirth. Linked open data in sensor data mashups. *Proc. Semantic Sensor Networks*, page 1, 2009.

[41] M. Lee, Y. Uhm, Z. Hwang, Y. Kim, J. Jo, and S. Park. A ubiquitous computing network framework for assisting people in urban areas. In *32nd IEEE Conference on Local Computer Networks, 2007. LCN 2007.*, pages 215–216. IEEE, 2007.

[42] C. Maier. Der Weg zur naturverträglichen Energiewende. *e & i Elektrotechnik und Informationstechnik*, 127(9):237–240, 2010.

[43] J. Mankoff, S. Fussell, T. Dillahunt, R. Glaves, C. Grevet, M. Johnson, D. Matthews, H. Matthews, R. McGuire, R. Thompson, et al. Stepgreen.org: Increasing energy saving behaviors via social networks. In *Proc. AAAI International Conference on Weblogs and Social Media (ICWSM), 106*, volume 113, 2010.

[44] A. Marcus. Dashboards in your future. *Interactions*, 13(1):48–60, 2006.

[45] A. McClard and P. Somers. Unleashed: Web tablet integration into the home. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1–8. ACM, 2000.

[46] M. McKeon. Harnessing the information ecosystem with wiki-based visualization dashboards. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1081–1088, 2009.

[47] B. Metz, O. Davidson, P. Bosch, R. Dave, and L. Meyer. IPCC: Summary for policymakers. *Climate Change 2007: Mitigation. Contribution of Working Group III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*, 2007.

[48] A. Moere, M. Tomitsch, M. Hoinkis, E. Trefz, S. Johansen, and A. Jones. Comparative feedback in the street: exposing residential energy consumption on house façades. *Human-Computer Interaction–INTERACT 2011*, pages 470–488, 2011.

[49] D. Morgan. *Focus groups as qualitative research*, volume 16. Sage Publications, Inc, 1997.

[50] R. Murty, G. Mainland, I. Rose, A. Chowdhury, A. Gosain, J. Bers, and M. Welsh. Citysense: An urban-scale wireless sensor network and testbed. In *IEEE Conference on Technologies for Homeland Security*, pages 583–588. IEEE, 2008.

[51] S. Nichols. New interfaces at the touch of a fingertip. *Computer*, 40(8):12–15, 2007.

[52] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, pages 249–256. ACM, 1990.

[53] V. Nisi, D. Nicoletti, R. Nisi, and N. Nunes. Beyond eco-feedback: using art and emotional attachment to express energy consumption. In *Proceedings of the 8th ACM conference on Creativity and cognition*, pages 381–382. ACM, 2011.

[54] C. North and B. Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *Proceedings of the working conference on advanced visual interfaces*, pages 128–135. ACM, 2000.

[55] T. O'Reilly. Lessons from open-source software development. *Communications of the ACM*, 42(4):32–37, Apr. 1999.

[56] S. Oviatt. Human-centered design meets cognitive load theory: designing interfaces that help people think. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 871–880. ACM, 2006.

[57] A. Ozok, D. Benson, J. Chakraborty, and A. Norcio. A comparative study between tablet and laptop pcs: User satisfaction and preferences. *Intl. Journal of Human–Computer Interaction*, 24(3):329–352, 2008.

[58] E. Paulos and T. Jenkins. Urban probes: encountering our emerging urban atmospheres. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 341–350. ACM, 2005.

[59] W. Pettigrew. Selecting the most effective current sensing technology. *Current Sensing*, 8:27–31, 2007.

[60] Z. Pousman, J. Stasko, and M. Mateas. Casual information visualization: Depictions of data in everyday life. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1145–1152, 2007.

[61] J. Rowley. The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, 33(2):163, 2007.

[62] M. Seeger. Key-value stores: a practical overview. *Computer Science and Media, Ultra-Large-Sites SS09*, 2009.

[63] W. Shadish, T. Cook, and D. Campbell. Experimental and quasi-experimental designs for generalized causal inference. 2002.

[64] J. Shi, R. Zhang, Y. Liu, and Y. Zhang. Prisense: privacy-preserving data aggregation in people-centric urban sensing systems. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[65] B. Shneiderman. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 33–39. ACM, 1997.

[66] B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley, 4 edition, 2004.

[67] T. Skog, S. Ljungblad, and L. Holmquist. Between aesthetics and utility: designing ambient information visualizations. In *IEEE Symposium on Information Visualization, 2003. INFOVIS 2003.*, pages 233–240. Ieee, 2003.

[68] S. Solomon, D. Qin, M. Manning, Z. Chen, M. Marquis, K. Averyt, M. Tignor, and H. Miller. IPCC: Summary for policymakers. *Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*, 2007.

[69] A. Spagnolli, N. Corradi, L. Gamberini, E. Hoggan, G. Jacucci, C. Katzeff, L. Broms, and L. Jönsson. Eco-feedback on the go: Motivating energy awareness. *Computer*, 44(5):38–45, 2011.

[70] M. W. Steenson. Urban software: The long view. *Habitar*, 2010.

[71] Y. Strengers. Designing eco-feedback systems for everyday life. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2135–2144. ACM, 2011.

[72] L. Suchman. *Plans and situated actions: the problem of human-machine communication*. Cambridge University Press, 1987.

[73] E. Tufte. *The visual display of quantitative information*. Graphics Press Cheshire, CT, 1983.

[74] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

[75] K. Whitehouse, F. Zhao, and J. Liu. Semantic streams: A framework for composable semantic interpretation of sensor data. *Wireless Sensor Networks*, pages 5–20, 2006.

[76] J. Wong and J. Hong. Making mashups with marmite: towards end-user programming for the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444. ACM, 2007.

[77] S. Yates and E. Aronson. A social psychological perspective on energy conservation in residential buildings. *American Psychologist*, 38(4):435, 1983.

[78] F. Zambonelli. Pervasive urban crowdsourcing: Visions and challenges. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 578–583. IEEE, 2011.