TECHNISCHE UNIVERSITÄT DRESDEN

Master Thesis

Generic Frontend for Exploring Sensor and Information Services

Author:
M.Sc. Uliana Andriieshyna
Matrikel-Nr: 3828303

Supervisors:
Dr.-Ing. Josef Spillner
Prof. Dr. rer. nat. habil.
Dr. h. c. Alexander Schill

April 10, 2014

Declaration of Authorship

I, Uliana Andriiehyna, declare that this thesis, titled "Generic Frontend for Exploring Sensor and Information Services", and the work presented in it have been done on my own without assistance. All information directly or indirectly taken from external sources is acknowledged and referenced in the bibliography.

Dresden, XX.XX.2013

Contents

Chapter 1

Introduction

The increasing numbers of sensor devices has increased the number of sensor-specific protocols, platforms and software. As a result various approaches have been proposed to interconnect, maintain and monitor various type of sensors[1, 2, 3]. That specificly focused on a platform development, protocol definition and software architecture for the concreate user-oriented requirements and for a narrowly focused areas of usage instead of defining a common system approach. And mainly in proposed approaches was discovered such type of questions as security and privacy, easy of development and monitoring, social dimensions [4], that completely don't consider requirements and criteria of an end-user. Therefore the area of research of this master thesis is dedicated to define generic frontend for exploring sensor and information services. The folowing sections ground the motivation for the chosen research field, define the central research questions and goals of this master's thesis, and describe the overall structure of the work.

1.1 Motivation

In the recent years with the technological progress in the computer science, information systems, and in particular sensor data systems, have become an essential part in daily life of the modern society. People have started to use them more often not only for manufactory, business, education but also for private reasons. Currently, most of the research is concerned with the protocol and middleware levels, whereas the potential of a generic interactive access to sensor and information services needs to be explored. This involves their selection, mash-ups, and usage within a client-controlled interface. In this master thesis, a first web-based prototype (portal) for such services is to be created. Users should be able to explore not just services, but also the information provided by them, and eventually be led to advanced usage patterns such as the development of third-party applications to access the information data and real-time streams.

1.2 Research Questions and Goals

Creating composite third-party services and applications from reusable components is an important technique in software engineering and data management. Although a large body of research and development covers integration at the data and application levels, weak work has been done to facilitate it at generic level. This master thesis discusses the existing user interface frameworks and component technologies used in presentation integration, illustrates their strengths and weaknesses, and presents some opportunities for future work.

As mentioned in the previous section, there are already exist many solutions for creating sensor-aware applications. But these platforms focuses on a single area of usage and they are not commonly suitable to support the dynamic and adaptable composition and usage of different type of sensors in one portal.

- Concept for a generic information and sensor service portal
- Development of the portal and associated dependency tools
- Demonstration using a convincing scenario

Therefore, this thesis is aimed at the development of a concept that provides users a possibility to personalize their current environment indepently from any type and kind of devices.

1.3 Structure

The thesis is structured in the following way:

Chapter 2 defines the background of the master's thesis describing the basic used terminology and the foundation platforms. A reference scenario and the requirements to a concept that has to be developed are also introduced in this chapter.

Chapter 3 is devoted to the state of the art analysis. The related research works in the areas of the sensor-driven platforms, the component based groupware systems, the browser based and non-browser based systems are investigated and evaluated against the defined requirements.

Chapter 4 focuses on the concept of the generic Frontend for exploring sensor and Information sevices, considering possible approaches, strategies, frameworks and necessary criteries, defined in *Chapter 3*.

Chapter 5 provides the implemented functionalities of the concept and describes evaluaion of results.

Chapter 6 concludes the master's thesis underlining the achieved goals and providing prospects for the future work.

Chapter 2

Foundations and Requirements Analysis

2.1 Terminology

The fundamental terms used in this thesis are described below for better understanding of the presented research work.

2.1.1 Frontend and Criteria

In computer science, the front end is responsible for collecting input in various forms from the user and processing it to conform to a specification the backend can use. The frontend is an interface between the user and the backend[5] and the separation of software systems into front and back ends simplifies development and separates maintenance. Therefore need to be distinguished what are the main requirements to a generic frontend for exploring sensed data.

- Loosely-coupling
- Fine-grained structure
- Multy-user capability
- Cross-platforming
- Adaptivity

Web Portal

Such a system, that generates requested information dynamically, displays that information in a useful manner in Web, maintains control over the created/saved files, automatically updates the information, and supports a distributed environment[6]. A web portal is most often specially-designed Web page at a website which brings information together from diverse sources in a uniform way. Usually, each information source gets its dedicated area on the page for displaying information (a portlet); often, the user can configure which ones to display.Portals provide a way for enterprises and organizations to provide a consistent look and feel with access control and procedures for multiple applications and databases, which otherwise would have been different web entities at various URLs. The features available may be restricted by whether access is by an authorized and authenticated user (employee,member) or an anonymous site visitor[7].

Mashup

The mashup application is a composite Web 2.0 application that aggregates and integrates heterogeneous web resources offered in a form of available Web APIs and sources for creating a new service. Mashups differ from traditional component-based applications in providing more situational character of these applications[8]. In general there distinguish the following three types of mashups[9]. Customer mashups are aimed at the combination and reformation data from various public sources according to users' needs. Data mashups aggregate similar types of resources from diverse sources into a new single data representation. And business, or enterprise, mashups define composite applications that are focused on solving heterogeneous business problems by supporting collaborative activities[10]. Concerning architectural styles of mashup applications, server-side and clientside mashups are distinguished. In the server-side mashups a content aggregation is realized on a server. The server plays a role of a proxy between the mashup application and other web sites that involved in this application. The opposite client-side mashups aggregate content on a client, typically, at a client's web browser[11]. Earlier, composite web applications covered the integration at the low application layers, such as the traditional data and business logic layers. However, mashups with their intention to reuse pieces of user interfaces (UIs) from different web resources increase the necessity of the UI integration, and the composition at the presentation layer[12]. Several research approaches refer to the problem of the component- based development of web applications at the presentation layer[13].

2.1.2 Sensor Data Streams

2.2 Web-based Framework Analysis

- **Bootstrap**
  Bootstrap is the most popular and widely used framework, nowadays. It's a beautiful, intuitive and powerful web design kit for creating cross browser, consistent and good looking interfaces. It offers many of the popular UI components with a plain-yet-elegant style, a grid system and JavaScript plugins for common scenarios.

  It is built with LESS and consists of four main parts: Scaffolding – global styles, responsive 12-column grids and layouts. Bear in mind that Bootstrap doesn't include responsive features by default. If design needs to be responsive this functionality have to be done manually. Base CSS – this includes fundamental HTML elements like tables, forms, buttons, and images, styled and enhanced with extensible classes. Components

– collection of reusable components like dropdowns, button groups, navigation controls (tabs, pills, lists, breadcrumbs, pagination), thumbnails, progress bars, media objects, and more. JavaScript – jQuery plugins which bring the above components to life, plus transitions, modals, tool tips, popovers, scrollspy (for automatically updating nav targets based on scroll position), carousel, typeahead (a fast and fully-featured autocomplete library), affix navigation, and more.

- **Foundation**
  Foundation is a powerful, feature-rich, responsive front-end framework. With Foundation user can quickly prototype and build websites or apps that work on any kind of device, with tons of included layout constructs, elements and best practices. It's built with mobile first in mind, utilizes semantic features, and uses Zepto instead of jQuery in order to brings better user experience and faster performance.

  Foundation has a 12-column flexible, nestable grid powerful enough to create rapidly multi-device layouts. In terms of features it provides many. There are styles for typography, buttons, forms, and various navigation controls. Many useful CSS components are provided like panels, pricing tables, progress bars, tables, thumbnails, and flex video that can scale properly your video on any device. And, of course, JavaScript plugins including dropdowns, joyride (a simple and easy website tour), magellan (a sticky navigation that indicates where is the user on the page), orbit (a responsive image slider with touch support), reveal (for creating modal dialogs or pop-up windows), sections (a powerful replacement for traditional accordions and tabs), and tooltips.

- **GroundworkCSS**
  GroundworkCSS is a new, fresh addition to the front-end frameworks family. It's a fully responsive HTML5, CSS and JavaScript toolkit built with the power of Sass and Compass which gives the ability to rapidly prototype and build websites and apps that work on virtually any device.

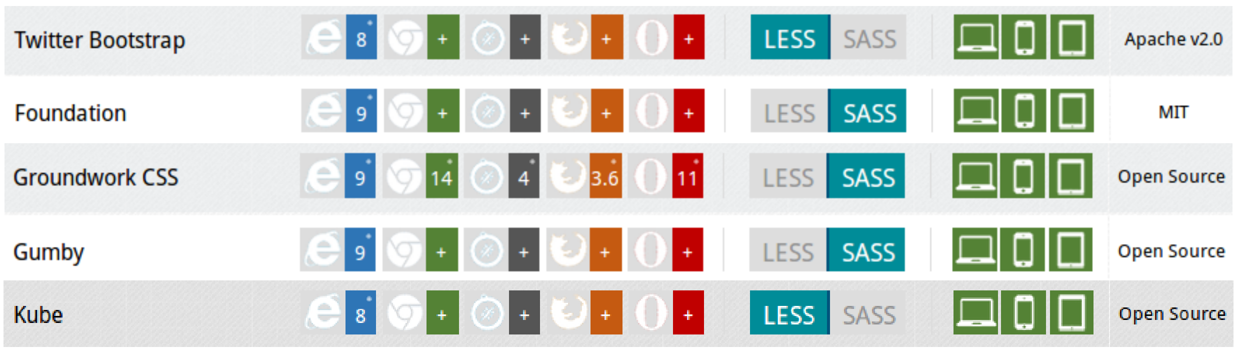
  It offers an extremely flexible, nestable, fraction-based, fluid grid system that makes creating any layout possible. GroundworkCSS has some really expressive features like tablets and mobile grids which maintain the grid column structure instead of collapsing the grid columns into individual rows when the viewport is below 768 or 480 pixels wide. Another cool feature is a jQuery ResponsiveText plugin which allows to have dynamically sized text that adapts to the width of the viewport: extremely useful for scalable headlines and building responsive tables. The framework includes a rich set of UI components like tabs, responsive data tables, buttons, forms, responsive navigation controls, tiles (a beautiful alternative to radio buttons and other boring standard form elements), tooltips, modals, Cycle2(a powerful, responsive content slider), and many more useful elements and helpers. It also offers a nice set of vector social icons and a full suite of pictographic icons included in FontAwesome. To see the framework in action user can use the resizer at the top center of the browser window. This way user can test the responsiveness of the components against different sizes and viewports while exploring the framework's features. GroundworkCSS is very well documented with many examples, and to get user started quickly the framework also provides several responsive templates. The only thing as a weakness is the missing of a way to customize download.

- **Gumby**
  Gumby is simple, flexible, and robust front-end framework built with Sass and Compass.

  Its fluid-fixed layout self-optimizes the content for desktop and mobile resolutions. It support multiple types of grids, including nested ones, with different column variations. Gumby has two PSD templates that get user started designing on 12 and 16 column grid systems. The framework offers feature-rich UI Kit which includes buttons, forms, mobile navigation, tabs, skip links, toggles and switches, drawers, responsive images, retina images, and more. Following the latest design trends the UI elements have Metro style flat design but can use Pretty style with gradient design too, or to mix up both styles. An awesome set of responsive, resolution independent Entypo icons, is completely integrated into the Gumby Framework. Gumby has also a very good customizer with color pickers which helps to build your custom download with ease.

- **Kube**
  Lastly, if user need a solid, yet simple base without needless complexity and extras, for your new project, Kube can be the right choice. Kube is a minimal, responsive and adaptive framework with no imposed styling which gives to user the freedom to create. It offers basic styles for grids, forms, typography, tables, buttons, navigation, and other stuff like links or images.

  The framework contains one compact CSS file for building responsive layouts with ease and two JS files for implementing tabs and buttons in your designs. If user is looking for maximum flexibility and customization, user can download developer version which includes LESS files, with variables, mixins and modules.



Twitter Bootstrap	8	+	+	+	+	LESS ✓	SASS	💻 📱 📱	Apache v2.0
Foundation	9	+	+	+	+	LESS	SASS ✓	💻 📱 📱	MIT
Groundwork CSS	9	14	4	3.6	11	LESS	SASS ✓	💻 📱 📱	Open Source
Gumby	9	+	+	+	+	LESS	SASS ✓	💻 📱 📱	Open Source
Kube	8	+	+	+	+	LESS ✓	SASS	💻 📱 📱	Open Source

Figure 2.1: Framework Comparison[1]

2.3 Prototype Requirements

Like most modern applications, each of these is structured into three layers: presentation, application (also called the business-logic layer), and data.

1. the granularity of the functions that the component applications provide is generally well suited for high-level integration (for example, we can tell an application to begin monitoring machine xyz without considering how this activity will affect data in the integrated application's database)

2. it's more stable because the component application is aware of the integration (it exposes the API) and will attempt to stabilize the interface across versions

2.4 Summary

Chapter 3

State of the Art

The following chapter covers an overview and analysis of the existent solutions in the related research areas of web-based third-party applications, which were designed specially for retrieving differet type of sensed data to the user application. At the beginning the prominent examples of dashboards platforms are studied and evaluated against the requirements described in the previous chapter with a purpose to clarify their personalized capabilities. Since the thesis is targeted at the creation of generic frontend,

3.1 Mashups Development Platforms

A growing popularity of mashup[1] applications, that is evidenced by a large amountof currently existent mashups, is accompanied by the rapidly increasing number of mashups development tools and platforms [14, 15, 16, 17]. Both commercial products and research prototypes have a broad range of features that simplify a mashups design process, and provide mashups storage and publication. One of the prominent consumer mashup tool is Yahoo Pipes[2], which allows to create new data mashups via a visual editor by mixing different feed types. Another example is DERI Pipes[3] tool, which is similar to Yahoo Pipes, but enhanced with the possibility to handle the Resource Description Framework (RDF) format. Different from the data pipes Intel Mash Maker[4] is implemented as a web browser's extension. While browsing a web page, the Mash Maker toolbar suggests possible additions, e. g. widgets, to the current page, which extend its capabilities and provide addition information from the other web sites.

3.2 Portal Development Platforms

Many of the Web applications seen today incorporate different patterns to merge functionality directed toward the end user. Many of them started out not being portals, but evolved

[1]http://www.programmableweb.com/applications
[2]http://pipes.yahoo.com/pipes/
[3]http://pipes.deri.org/
[4]http://software.intel.com/en-us/articles/intel-mash-maker-mashups-for-the-masses

into portals as functionality was added. This section focuses on the design approaches for portal solutions. It addresses the design stage of the solution development process, highlighting the key issues that are specific to portal and application designs[18, 19].

1. Object-Oriented design patterns, including Model-View-Control design
2. Portlet framework design

3.2.1 Model-View-Control Pattern

In the design shown in Figure 1 on page 8, Model represents the application object that implements the application data and business logic. The View is responsible for formatting the application results and dynamic page construction. The Controller is responsible for receiving the client request, invoking the appropriate business logic, and based on the results, selecting the appropriate view to be presented to the user. The Model represents enterprise data and the business rules that govern access to and updates to this data. Often the Model serves as a software approximation to a real-world process, so simple real-world modeling techniques apply when defining the Model. A View renders the contents of a Model. It accesses enterprise data through the Model and specifies how that data should be presented. It is the View's responsibility to maintain consistency in its presentation when the Model changes. This can be achieved by using a push Model, where the View registers itself with the Model for change notifications, or a pull Model, where the View is responsible for calling the Model when it needs to retrieve the most current data. A Controller translates interactions with the View into actions to be performed by the Model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a Web application, they appear as GET and POST HTTP requests. The actions performed by the Model include activating business processes or changing the state of the Model. Based on the user interactions and the outcome of the Model actions, the Controller responds by selecting an appropriate View.

3.2.2 Portlet Framework Design

3.3 Browser Based Collaborative Systems

3.4 Non-Browser Based Collaborative Systems

3.5 Summary

This chapter briefly introduced main approaches for building web-based dashboards by retriving sensed data. Main focus was given to its multy-user usability, adaptive UI design, dynamic content composition.

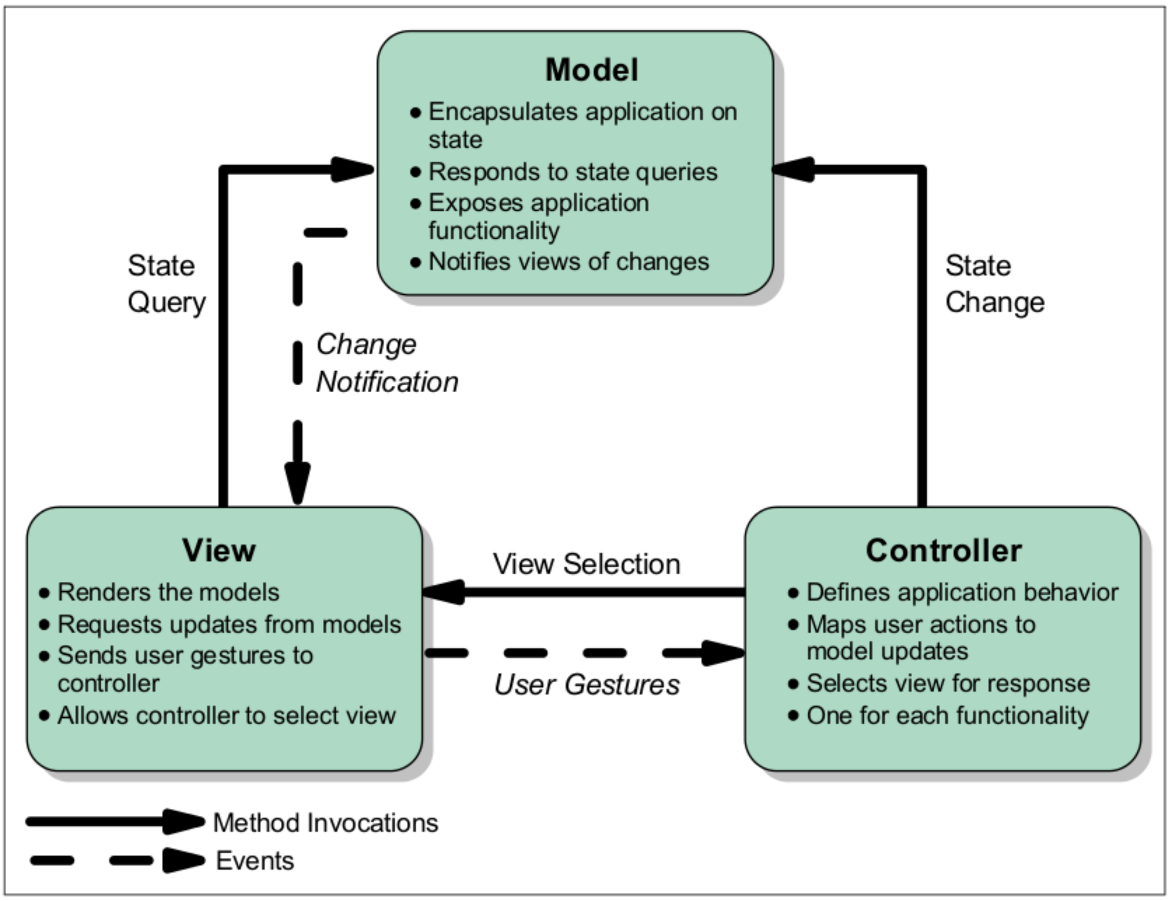

Model
• Encapsulates application on state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

Figure 1   The Mode-View-Controller design pattern

Figure 3.1: MVC Pattern

Chapter 4

Concept

The chapter describes a concept of a generic frontend for exploring sensor data, that in the same time controlled and provisioned by end users request. The concept is developed based on the analysis of the current state of the platform, the defined requirements to the third-party services and applications (section X.X) and knowledges gained from the studied related works (chapter 3).

- Concept for a generic information and sensor service portal
- Development of the portal and associated dependency tools

4.1 Requirements

4.1.1 Functional Requirements

4.1.2 Non-functional Requirements

4.2 Frontend Architecture

4.2.1 Sensor Registry

4.2.2 Proxy

4.2.3 Authentification Stub

4.2.4 Databases

4.3 Summary

In this master thesis, a first web-based prototype (portal) for such services is to be created. Along with it, a light-weight scenario service registry will be needed. Users should be able

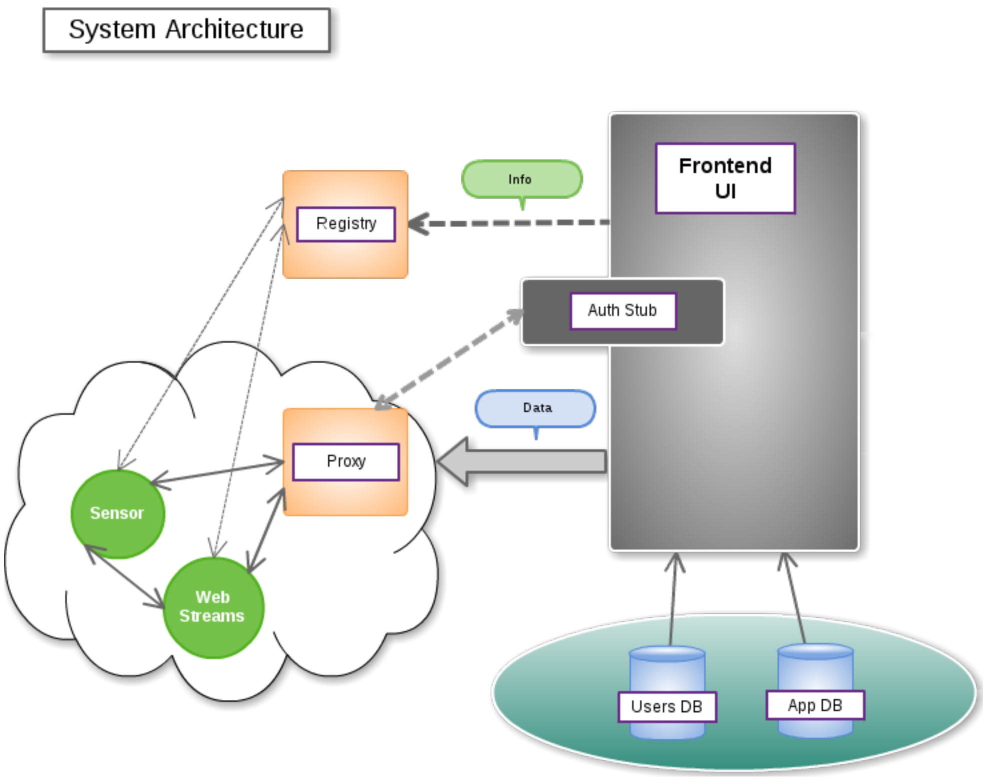


Figure 4.1: System Architecture

to explore not just services, but also the information provided by them, and eventually be led to advanced usage patterns such as the development of third-party applications to access the information data and real-time streams.

Chapter 5

Implementation and Evaluation

The chapter presents a prototype of the reference scenario considered in the section 2.3. The prototype implements the major aspects proposed in the concept (chapter 4).

5.1 Overview of Framework

5.2 Data Flow Model

5.3 Database Model

5.4 Use Cases

5.4.1 Frontend

5.4.2 Choosen Environment

5.4.3 Choosen Tools

5.4.4 Use Cases Realization

5.4.5 Summary

Chapter 6

Conclusion and Outlook

6.1 Conclusion

The chapter summarizes the presented thesis providing an overview of each chapter and describing the achievement of the thesis's goals defined in the section 1.2. At the end of the chapter suggestions for the future work are made.

6.1.1 Addressed research questions

Achieved Goals

- Some;
- Statements;
- Here;

6.1.2 Practical results

6.2 Future work

To conclude the thesis, suggestions for the future work are made. These suggestions are devoted to improve either the developed concept or the current implementation.
Visualization and interaction metaphor for the introduced access control
Enhanced user interface for application part
User interface to support the introduced dynamic composition

List of Figures

List of Tables

Bibliography

[1] S. Suakanto, S.H. Supangkat, Suhardi, and R. Saragih. Smart city dashboard for integrating various data of sensor networks. In *ICT for Smart Society (ICISS), 2013 International Conference on*, pages 1–5, 2013.

[2] S Bendel, T Springer, D Schuster, A Schill, R Ackermann, and M Ameling. A service infrastructure for the internet of things based on xmpp. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 385–388. IEEE, 2013.

[3] Xianfeng Song, Chaoliang Wang, Masakazu Kagawa, and Venkatesh Raghavan. Real-time monitoring portal for urban environment using sensor web technology. In *Geoinformatics, 2010 18th International Conference on*, pages 1–5. IEEE, 2010.

[4] Michael Eggert, Roger Häußling, Martin Henze, Lars Hermerschmidt, René Hummen, Daniel Kerpen, Antonio Navarro Pérez, Bernhard Rumpe, Dirk Thißen, and Klaus Wehrle. Sensorcloud: Towards the interdisciplinary development of a trustworthy platform for globally interconnected sensors and actuators. *arXiv preprint arXiv:1310.6542*, 2013.

[5] Wikipedia. Front and back ends — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Front_and_back_ends&oldid=572495173`, 2013. [Online; accessed 15-November-2013].

[6] Mohsen Rezayat. The enterprise-web portal for life-cycle support. *Computer-Aided Design*, 32(2):85–96, 2000.

[7] Wikipedia. Web portal — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Web_portal&oldid=578361576`, 2013. [Online; accessed 17-November-2013].

[8] Jin Yu, Boualem Benatallah, Fabio Casati, and Florian Daniel. Understanding mashup development. *Internet Computing, IEEE*, 12(5):44–52, 2008.

[9] Sunilkumar Peenikal. Mashups and the enterprise. *Strategic white paper*, 2009.

[10] Volker Hoyer, Katarina Stanoesvka-Slabeva, Till Janner, and Christoph Schroth. Enterprise mashups: Design principles towards the long tail of user needs. In *Services Computing, 2008. SCC'08. IEEE International Conference on*, volume 2, pages 601–602. IEEE, 2008.

[11] Ed Ort, Sean Brydon, and Mark Basler. Mashup styles, part 2: Client-side mashups, 2007.

[12] Florian Daniel, Maristella Matera, Jin Yu, Boualem Benatallah, Regis Saint-Paul, and Fabio Casati. Understanding ui integration: A survey of problems, technologies, and opportunities. *Internet Computing, IEEE*, 11(3):59–66, 2007.

[13] Stefan Pietschmann, Tobias Nestler, and Florian Daniel. Application composition at the presentation layer: alternatives and open issues. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, pages 461–468. ACM, 2010.

[14] Antero Taivalsaari. Mashware: The future of web applications. 2009.

[15] Agnes Koschmider, Victoria Torres, and Vicente Pelechano. Elucidating the mashup hype: Definition, challenges, methodical guide and tools for mashups. In *Proceedings of the 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web at WWW*, 2009.

[16] Florian Daniel, Agnes Koschmider, Tobias Nestler, Marcus Roy, and Abdallah Namoun. Toward process mashups: key ingredients and open research challenges. In *Proceedings of the 3rd and 4th International Workshop on Web APIs and Services Mashups*, page 9. ACM, 2010.

[17] Saeed Aghaee, Marcin Nowak, and Cesare Pautasso. Reusable decision space for mashup tool design. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 211–220. ACM, 2012.

[18] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. big'web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, pages 805–814. ACM, 2008.

[19] Daniel Su Kuen Seong. Usability guidelines for designing mobile learning portals. In *Proceedings of the 3rd international conference on Mobile technology, applications & systems*, page 25. ACM, 2006.