GRADUATION REPORT

The rise of HTML5: downfall of plugins?

ABSTRACT

A development driven research into the maturity and usefulness of HTML5 with respect to Sorama's current Silverlight powered products.

Koen Vrijdag

Sorama B.V.

FONTYS TITLE PAGE

Studentinformatie:

Naam student:	K. H. Vrijdag
Studentnummer:	2152260
Afstudeerrichting:	Software Engineering (Voltijd)
Afstudeerperiode:	2 september 2013 t/m 29 januari 2014

Bedrijfsinformatie:

Naam bedrijf/instelling:	Sorama B.V.
Afdeling:	Research & Development
Plaats:	Eindhoven
Bedrijfsbegeleider:	Dr. Merijn de Jonge – Chief Software Development

Gegevens docentbegeleid(st)er:

Naam*:*	Mieke van Vucht

Gegevens verslag:

Titel:	The rise of HTML5: downfall of plugins?
Datum uitgifte afstudeerverslag:	9 januari 2014

Getekend voor gezien door bedrijfsbegeleid(st)er:

Datum: 09-01-2014

[illegible]

De bedrijfsbegeleider,

Merijn de Jonge

DOCUMENT HISTORY

REVISION

This document has been published with the following distinguishable revisions:

Version	Status	Date	Changes
0.1	Concept	05-09-2013	Initial draft (mainly skeleton).
0.2	Concept	16-12-2013	Added a lot of content for first version.
0.3	Near final	06-01-2014	Improvements in structure, project definition, and depth of contents based on feedback by both school and company mentors. Also addition of previously missing content, improved figures and layout, and more.
0.4	Near final	08-01-2014	Small improvements and inclusion of Appendices.
1.0	Final	09-01-2014	Small improvements, final version.

DISTRIBUTION

This document has been distributed to:

Version	Date	Name	Function
0.2	16-12-2013	Mieke van Vucht + Merijn de Jonge	School and company mentors, feedback
0.3	06-01-2014	Mieke van Vucht + Merijn de Jonge	School and company mentors, possible additional feedback
0.4	08-01-2014	Sorama	Colleagues, possible feedback
1.0	09-01-2014	Sorama, Fontys	Final version that will be used for assessment.

PREFACE

For the graduation phase of my study ICT & Software Engineering at Fontys University of Applied Sciences, I accepted an internship project at Sorama B.V. in Eindhoven, The Netherlands. This internship period lasted from September 2013 until the end of January 2014, a period during which Sorama has provided me both with a place to work and also with the knowledge and support needed to execute and complete the assignment.

During the internship I was guided by Merijn de Jonge (Sorama B.V.) and Mieke van Vucht (Fontys University of Applied Sciences). I would like to thank both of them for their provided feedback and support. I would like to thank my other colleagues at Sorama as well, who were always available for feedback and ideas, and have had direct or indirect influence on the project.

Eindhoven, January 2014

Koen Vrijdag

TABLE OF CONTENTS

SUMMARY **5**

SAMENVATTING **6**

DEFINITIONS **7**

1 INTRODUCTION **8**

2 SORAMA B.V. **9**

2.1 Company and product 9

2.2 Sorama Web Portal 10

2.3 Performing a sound measurement 10

3 PROJECT BREAKDOWN **11**

3.1 Problem definition 11

3.2 Project goals and products 12

3.3 Global approach 12

3.4 Requirements and research questions 13

4 HTML5, WHAT IS IT ALL ABOUT? **14**

4.1 HTML4 and its limitations 14

4.2 The inadequacy of plugins 14

4.3 Removing the bandage: HTML5 15

4.4 What is new in HTML5 16

4.5 Current Implementation Status 17

5 HTML5 AND SINGLE-PAGE APPLICATIONS **19**

5.1 What is a SPA and why the interest? 19

5.2 SPA Components 19

5.3 MVVM Pattern 20

5.4 HTML5 SPA Frameworks 21

5.5 Which framework to choose? 22

6 HTML5 REMOTE CONTROL APPLICATION **23**

6.1 Description and features 23

6.2 Sorama Imaging System architecture 24

6.3 Remote Control architecture 25

6.4 User interface 25

6.5 Development issues and solutions 27

6.6 Resulting products 30

7 CONCLUSION AND RECOMMENDATIONS **32**

8 EVALUATION **34**

9 REFERENCES **35**

10 APPENDICES **36**

Appendix I: Project initiation document 36

Appendix II: Additional SIS and Services Background 36

Appendix III: Remote Control App UI Design 36

Appendix IV: Libraries, Frameworks and HTML5 features 36

SUMMARY

Sorama provides technology to measure and analyze sounds and vibrations in high detail. To allow configuration and control over these measurements and their results, Sorama has developed the Sorama Web Portal, which is powered by the web technology Silverlight. This dependency to Silverlight however introduces two problems that are distinguished in this project: (1) Silverlight has recently been discontinued, and (2) Silverlight is not very mobile capable. The first will become more of a problem in the future when the support for Silverlight will cease too, while the second problem can already be found in other issues, such as (negative) influence on the workflow in certain scenarios.

This project was built around the research into HTML5, a technology that is more multiplatform capable and unlike Silverlight does not require a browser plugin or suffer from discontinuation. The project focused on the analysis of and experiments with Single-Page Application (SPA) frameworks, which are essentially platforms that allow development of responsive, rich and maintainable HTML5 applications. There was also focus on specific libraries that add towards a SPA's architecture and functionality. The results demonstrate that HTML5 based SPAs offer a very capable platform, ultimately able to help dedicated browser plugins to become obsolete.

Using these technologies, an HTML5 Single-Page Application has been developed to both demonstrate some of HTML5's capabilities, and partially solve Silverlight's mobility issue as well by featuring remote control capabilities to the Web Portal's execution of measurements. The product can be used as a supplementary tool to the Web Portal's features, and actions between the two applications are synchronized in real time to benefit simultaneous usage in a typical scenario's workflow.

SAMENVATTING

Sorama levert technieken die het mogelijk maken om geluid en vibraties tot op hoog detail te meten en te analyseren. Om gebruikers hierin bij te staan, heeft men de Sorama Web Portal ontwikkeld, een applicatie die gebruik maakt van de web technologie Silverlight. Dit zorgt echter voor een tweetal problemen die aan de orde worden gesteld in dit project: (1) Silverlight wordt niet meer doorontwikkeld en heeft nog een beperkte levensduur, en (2) Silverlight is niet mobielvriendelijk. Het eerstgenoemde is vooral op de lange termijn een probleem, want voorlopig is er nog officiële support. Het tweede probleem uit zich echter nu ook al in andere zaken, zoals in een vertraagde workflow bij het uitvoeren van een meting in bepaalde scenario's.

In dit project is onderzoek gedaan naar HTML5, een technologie die meer multi-platform capabel is dan Silverlight, en in tegenstelling tot Silverlight aan het begin van zijn levensduur zit en volop groeit in features en support. Een belangrijke aspect tijdens het onderzoek was de verdieping in Single-Page Application (SPA) frameworks, die een platform bieden om responsieve, rijke en onderhoudbare HTML5 applicaties te ontwikkelen. Er is ook verdieping geweest in specifieke libraries die bijdragen aan de architectuur en functionaliteit van een SPA. De resultaten laten zien dat HTML5 SPA's een adequaat platform bieden, en er uiteindelijk bij helpen zorgen dat browser plug-ins in de toekomst niet meer zo nodig zullen zijn.

Gebruikmakende van deze technologieën is er een HTML5 Single-Page Application ontwikkeld die doel heeft om een aantal mogelijkheden van HTML5 te demonstreren, en om het mobiliteitsprobleem in bepaalde meetscenario's gedeeltelijk op te lossen. Dit doet het door als afstandsbediening te fungeren voor metingen die via de Web Portal zijn opgezet. Bij simultaan gebruik van de twee applicaties worden acties tussen de twee gesynchroniseerd voor een optimaal gebruik binnen de workflow van een meetscenario.

DEFINITIONS

The following table defines a series of terms used in this report that may or may not be unknown to the reader.

Term	Definition
Acquisition Service	The acquisition client hosts multiple (web) services. This term refers to either one or multiple of these services.
Acquisition client	Refers to the piece of Sorama software that is installed on measurement computers. It acquires measurement data from connected measurement hardware and provides a local web service interface to allow interactions from other applications.
Ajax	Asynchronous JavaScript and XML, refers to collection of techniques used to create asynchronous behavior
CSS(3)	Abbreviation of Cascading Style Sheets, a language used for dynamicly defining styles for web sites. The number 3 refers to the third and most recent revision.
Domain	When referred to simply “domain”, the reference relates to the server on which the Domain Services are hosted that provide a back-end for the Web Portal.
Durandal	A Single-Page Application framework.
Flash	A web technology by Adobe that is very broadly used to power video streams, games, advertisements and other things. Requires a browser plug-in.
Framework	Combines and build upon libraries to provide a (more) application scope basis that can be used for simplified application development.
HTML	A markup language used to define the content and layout of a web page.
HTML5	Newest iteration of HTML specification. When referred to, it usually covers both the new and improved HTML elements, CSS3 capabilities and JavaScript API’s.
JavaScript	Scripting language used to power many web sites. Code is interpreted and executed by the browser.
Library	Can be included into a project to (easily) provide additional features to the developed application. Traditionally focuses on a single mechanic or application aspect.
Measurement	A sound measurement, performed using Sorama hardware. May consist of multiple steps. Traditionally configured and controlled via the Sorama Web Portal.
RIA	Abbreviation of Rich Internet Application.
Service	Refers to a Web Service, which is the part of an application that exposes an interface to the network to which other applications may communicate in order to retrieve information or (remotely) execute commands.
Silverlight	A web technology by Microsoft that can power (parts of) web applications. Requires a plug-in, similar to Flash.
Single Page Application (SPA)	A web application that is rooted inside a single web page. This provides many benefits compared to normal web applications, such as reliability and responsiveness.
SOAP	Communication protocol using XML-based envelopes to wrap and pass objects.
WCF	Abbreviation of Windows Communication Foundation, a runtime used for creating service-oriented applications.
Web Portal	Refers to the Sorama Web Portal, which is a Silverlight powered web application that allows configuring and performing of sound measurements, among other features.

1 INTRODUCTION

The web is an ever evolving platform. While computers and internet connections continuously increase in speed and form factors, the ideas of both users and developers about what makes a good web site shifts also. From simple plain text-on-background websites that dominated the 90's, websites have grown out to become content-rich web applications that can offer extensive data manipulation and a lot of complex controls and graphics.

Browser plugins like Adobe Flash and Microsoft Silverlight have contributed greatly to the rise of such content-rich web sites, but new technologies still arise. The goal of the project described in this report is to research such a technology: HTML5.

Is HTML5 the technology that obsoletes plugins? Or does one technology's rise not mean the downfall of another? Read along and we will try to figure this out.

DOCUMENT STRUCTURE

Chapters *2* and *3* will elaborate on the project's background and goals, while chapters *4* and *5* will reflect the majority of the research and results. Chapter *6* will represent the development part of the project and chapters *7* and *8* will conclude the report and provide recommendations and a project evaluation.

2 SORAMA B.V.

Sorama is a technology driven company founded in 2009 by Dr. Ir. Rick Scholte as a result of his PhD. research at Eindhoven University of Technology (TU/e). Since then, the company has grown from a small company located in the basement of TU/e campus, towards a more established company with approximately twelve employees, located in the technology dense environment of Strijp-S in Eindhoven. This chapter will introduce some of Sorama's products and offer an abstracted insight into how Sorama's clients use these products.

2.1 Company and product

Sorama develops a technology to measure, analyze and visualize sounds and vibrations transmitted by objects and appliances. This technology allows Sorama's clients to gain more insight in their products and the possible noises and vibrations that (parts of) these products produce. As a result, it provides these clients with the possibility to more specifically be able to optimize (parts of) their products.

The Sorama *Sound Camera* (Figure 1) consist of small and affordable sound sensors. On their own, these sensors cannot capture sound accurately, but by using large quantities and combining their individually captured data into a single measurement, the accuracy can get very high. Sorama exploits this by placing the microphones on an 8x8 grid, counting up to 64 microphones per sound array. In a Sorama Sound Camera there are traditionally 4x4 of these arrays, and thus a total of 1024 of those small sensors. Using a number of data processing algorithms the data of all individual sensors is combined in such a way that it can be used to create images of the sound waves and their behavior during the measured period of time (Figure 2).

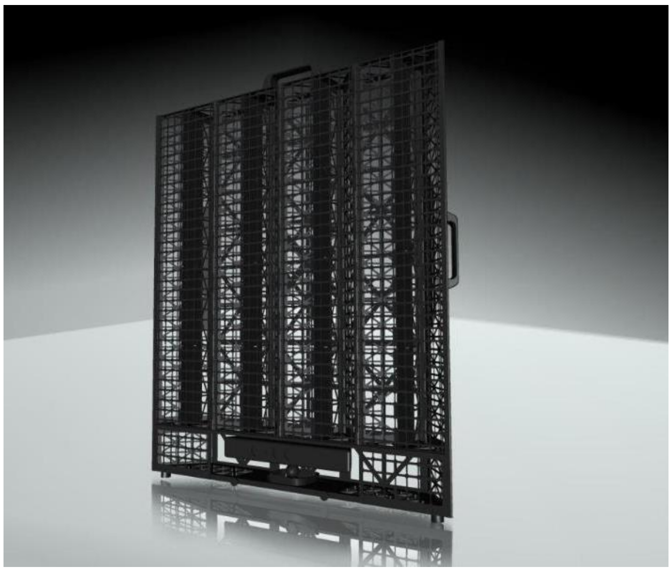


Figure 1. The Sorama Sound Camera.

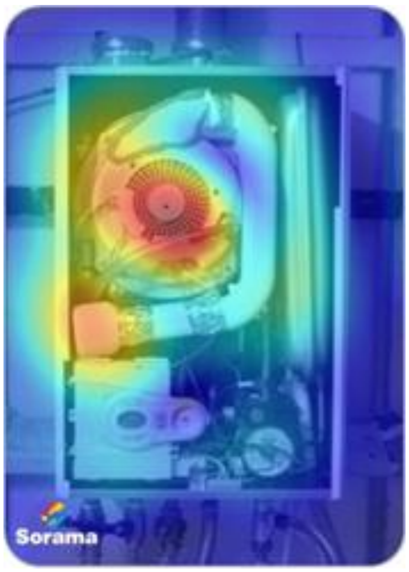


Figure 2. A sound image of a hard disk drive at a single time frame and frequency.

2.2 Sorama Web Portal

In order to allow users control over a Sorama Sound Camera and be able to execute measurements and analyze measurement results, Sorama has developed the Sorama *Web Portal* (Figure 3). This is an application that can be accessed from the browser and offers control over the measurement hardware, given that they are connected to a computer running the Sorama *Acquisition Client* software.

The Web Portal offers features to configure and start a measurement, and access and view results in high detail. Among these details on which one can focus are views of the sound frequency range (to easily spot anomalies), options to focus on a specific sound frequency, compare multiple results, use colors to differentiate pressure levels, and much more. The portal also handles uploading of (partial) results to the cloud so they can be accessed from devices other than the measurement computer as well. More detail on the Web Portal can be found in *Appendix II: Additional SIS and Services Background*.

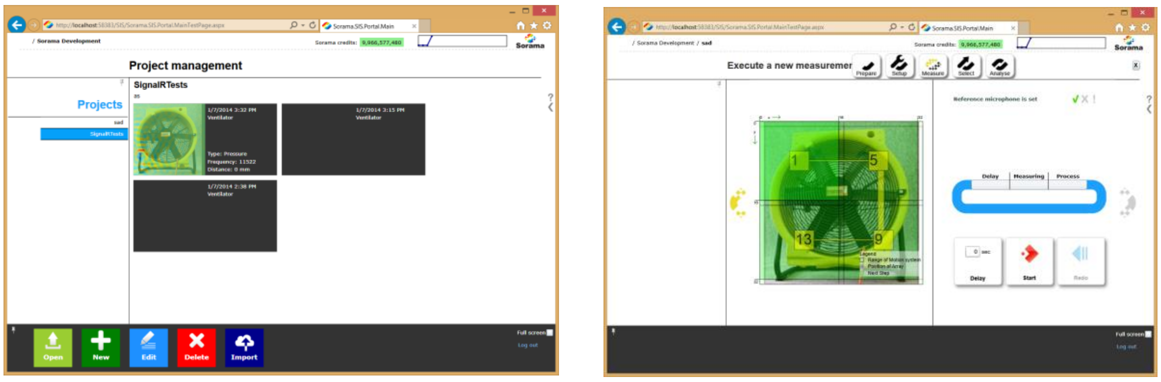


Figure 3. The Web Portal. Left shows the measurement overview, right shows the measurement execution screen.

2.3 Performing a sound measurement

An important aspect of the Web Portal's features is the execution of measurements. Performing a sound measurement requires a couple of steps. After pre-configuration of the measurement including its dimensions, for each step of a measurement the following three operations need to be iterated, during which the user will interact with the Web Portal.

1. **Position the sound camera** towards the position matching the dimensions that are presented by the Web Portal.
2. Instruct the Web Portal to **start** the measurement step.
3. **Accept** or **redo** the step.

This workflow can be visually represented with the flowchart as shown below in Figure 4.

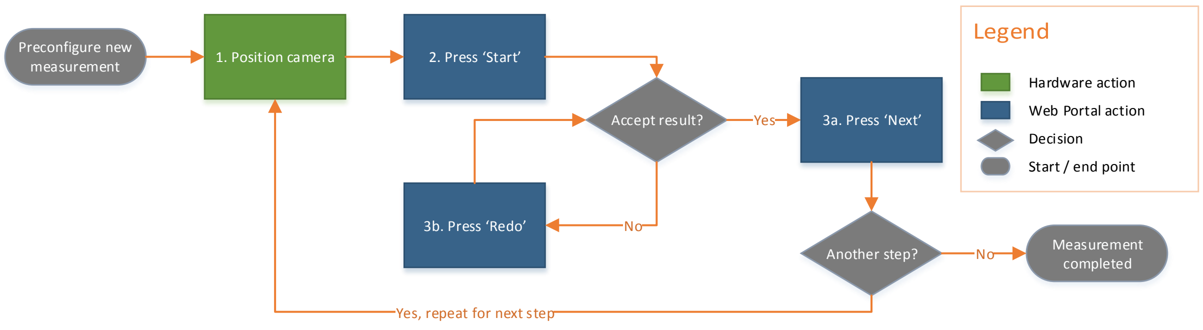


Figure 4. Flowchart representing the (abstracted) measurement process using the Web Portal.

3 PROJECT BREAKDOWN

This chapter will elaborate on the problems that initiated this project and what goals this project aimed to fulfill and the (main) products that have come with it. It will also introduce the major requirements and corresponding research questions and elaborate on the global approach used to execute the project. A more elaborate project definition can be found in *Appendix I: Project Initiation Document*.

3.1 Problem definition

SILVERLIGHT

The current Web Portal (which has been introduced in *Paragraph 2.2*) fully depends on *Microsoft Silverlight*, a powerful web technology that powers a series of unique and complex controls used by Sorama to provide a user interface with both a high usability and a lot of features. Although Microsoft had already announced that it had discontinued further development of the technology and will not be adding any additional features, this technology was found to provide the best tools for creating rich web applications that met Sorama’s requirements. The current version (Silverlight 5) will still receive support until the year 2021 though (Microsoft, n.d.), so for the time being, this technology will remain sufficient.

Figure 5. Silverlight logo

For future benefits however, Sorama wants to investigate the possibility of a potential migration to an alternative technology that provides a more long term solution. With *HTML5* being one of the modern technologies with growing support (by Microsoft amongst others) and continued development of its feature set (see *Chapter 4. HTML5, what is it all about?),* HTML5 is of main interest to Sorama.

MOBILITY

Due to increasing use of modern devices such as smartphones and tablets by both Sorama’s clients and other consumers, Sorama also desires to investigate possibilities into exploiting these products to improve current products. This includes investigation into (better) mobile support for the Web Portal, which is currently not very mobile friendly, mostly due to its Silverlight dependency.

It also includes improving upon the measurement workflow. This workflow sufficiently covers most measurement scenarios (Figure 4), but it is less ideal in certain situations in which the user cannot be near the hardware during the measurement (due to safety reasons or influencing results) and has to move out of the room before each step. The measurement workflow in such a case is longer than usual (Figure 6).

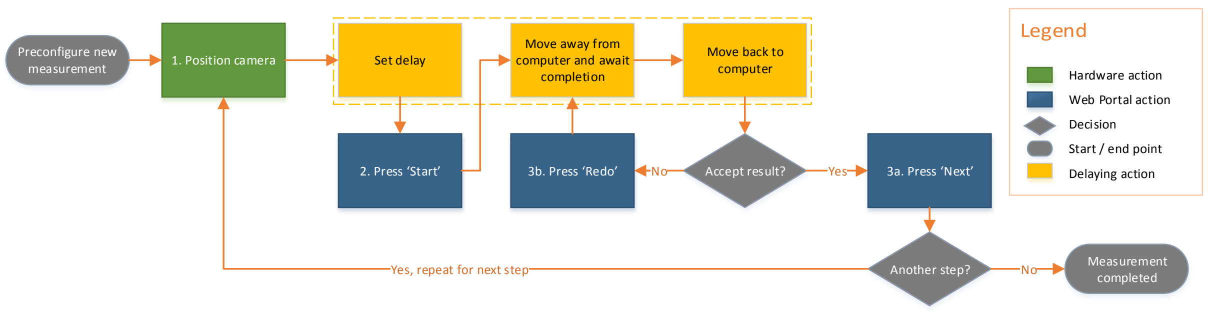


Figure 6. The measurement workflow from Figure 4 in a less ideal scenario. Note the additional actions.

3.2 Project goals and products

Primary goal of the project is to be able to provide insight and advise Sorama about HTML5 as a technology and its potential usefulness with respect to Sorama's activities, specifically with respect to the Silverlight Web Portal. This goal will be reached by matter of development driven research, as will be described later in this chapter.

Secondary goal of the project is to solve the mobility issue presented in *Paragraph 3.1*, by developing a supplementary HTML5 powered *Remote Control* application that can be used to remotely control measurement execution using smartphone or other device. This ought to improve measurement scenarios in which the user is not near the measurement computer at all times (Figure 6), and support the research of subjects related to the primary goal as well.

The main research question answered during this project is the following:

- **How can HTML5 and its feature set be exploited to benefit Sorama's Web Portal and other products?**

The remainder of this chapter will elaborate more on the used approach (*Paragraph 3.3*) and on the desires and requirements of the developed application and associated sub research questions (*Paragraph 3.4*). The main results to the research will be described in the chapters 4 and 5, while an elaborate description of the developed application can be found in *Chapter 6 HTML5 Remote Control Application*.

3.3 Global approach

As briefly stated in the previous paragraph, the research part of this project is development driven. Specifically by the development of the HTML5 powered Remote Control application. The requirements for this application have been defined during the orientation period of the project, and have been chosen in such a way that they cover a broad range of development related topics that are essential in any scalable HTML5 web application. Covering this broad range of topics during development assures that an equally broad range of topics should be researched, ultimately leading to completion of both the secondary goal (development of the application), and the primary goal (advise about HTML5).

To benefit this method of development driven research, the methodology of *Agile Scrum* is used to direct both development and research related activities. Sorama traditionally uses Scrum sprints of two weeks in order to plan and execute their activities and it was chosen that the student also uses this in order to allow a structured approach. Doing so, for example, research and development of the communication part of the application have been fit into one Scrum sprint. Other topics have been covered in sprints similarly.

3.4 Requirements and research questions

Since there has been chosen for a development driven research approach, it is fundamental to first not only introduce the research questions, but also the major application requirements that they belong to. The following five requirements are the ones that have been the most influential to the research.

1. *The application must be HTML5 powered*.
   a. What is covered by the term HTML5 and why has it been seeing increase of interest over the last couple of years?
   b. How 'finished' is HMTL5 and what is its current state of implementation?

These research questions will be answered in *Chapter 4. HTML5, what is it all about?*

2. *The application must be a Single-Page Application*.
   a. What is a SPA and why is it useful?
   b. What frameworks (and supplementary libraries) provide SPA implementations with HTML5?

These research questions will be answered in *Chapter 5. HTML5 and Single-Page Applications*.

3. *The application must be able to (either directly or indirectly) control the measurement hardware*.
   a. What is the architecture of the Sorama Imaging System?
   b. How can the Remote Control Application be fit into this architecture?
   c. How can HTML5 consume the services provided by this Sorama Imaging System architecture?

These subjects will be covered in *Chapter 6: HTML5 Remote Control Application* and also *Appendix II*: *Additional SIS and Service Background*.

4. *The application should be usable on smaller screen devices such as smartphones*.
   a. What user interface can both improve upon the Web Portal's measurement execution interface, and respect the Web Portal design elements?
   b. How can HTML5 and open source libraries be used to provide a responsive user interface and advanced graphics for the execution of measurements.

These subjects will be elaborated on in *Chapter 6* as well, with more details on design in *Appendix III: UI Design* and library details in *Appendix IV: Libraries, Frameworks and HTML5 features*.

5. *The application could feature synchronization between itself and the existing Web Portal*.
   a. What technology can be used to notify both the Silverlight Web Portal and the Remote Control?
   b. How can this technology be used and implemented to provide some sort of synchronization between both clients and the server?

This will be elaborated on in some extent in *Paragraphs 6.5* and 6.6, although (technical) implementation details have been omitted.

4 HTML5, WHAT IS IT ALL ABOUT?

The term HTML5 has been used multiple times already in this report, and as an IT professional or student, you are most likely already somewhat familiar with the term. It has been a rising topic in web development for the past few years (Google Trends, 2013), but what is it really about? Before answering this question, it is useful to briefly review earlier technologies.

4.1 HTML4 and its limitations

HyperText Markup Language (HTML) is a markup language with an open standard used to define information and graphics for web pages. Web browsers are developed to interpret this language and display the contents with respect to defined specifications. The most recent version of HTML is 4.01, and it defines many things on which web sites depend to display their contents, such as links, images, forms, and much more. It offers a lot of possibilities to create wide range of web sites and applications using this markup, supplemented by *Cascading Style Sheets (CSS)* and *JavaScript (JS)*, for respectively dynamic designs and more advanced behavior. However, since its definition in 1999 (W3C, 1999), the needs and expectations of both users and developers have changed and gradually started to exceed what HTML 4.01 offers.

Since pure HTML (with their CSS and JavaScript supplements) proved too limiting for these needs, developers had to embrace browser plugins in order to create their applications. These pieces of additional software hook into the browser to execute their logic and display their (embedded) contents. The best example of one such a plugin technology is Adobe Flash, a technology that gained most of its popularity due to its video and audio streaming possibilities, leveraged by many popular web sites such as YouTube and Vimeo. It is also commonly used by games and advertising.

Figure 7. Flash logo

4.2 The inadequacy of plugins

While plugins have been developed mostly to close the gap between HTML4 and web application requirements, they do not offer a perfect solution. For one, they can introduce additional security risks, which have proven multiple times to be exploitable (Adobe, n.d.). They require additional software to be installed and maintained as well. This leads to more memory overhead, but also to hardware dependencies. If a platform is not supported by the plugin developer, than you cannot run any applications that require it, even if you run a modern browser. Especially on operating systems other than Microsoft Windows and Macintosh OS X (including mobile device operating systems), the support of plugins is deficient.

Due to all these limitations, plugins cannot be considered to be an adequate solution to the limitations of HTML, but only a temporary solution; like a bandage on a wound it is only a matter of time until the problem has been solved effectively and the temporary solution becomes redundant.

4.3 Removing the bandage: HTML5

The organizations responsible for the specification of HTML, *World Wide Web Consortium (W3C)* and *Web Hypertext Application Technology Working Group (WHATWG)*, have been working on a new version of HTML since 2004. They do this based on suggestions by both browser developers and other companies. In 2008, they published the initial specification draft, and since that date, browser manufacturers have been gradually implementing features based on these early specifications. Since the specifications were not final, current implementations could in theory be changed in the future. The final recommendation of HTML5 specification is scheduled to be published in 2014.

Figure 8. HTML5 Logo

However, from both the initial specification and the Candidate Specification that was published in 2012 (W3C, 2012), and also from the current implementations in browsers, it is clear that HTML5 will introduce a lot of elements that currently require plugins, such as video streaming and advanced graphics.

Browser manufacturers seem already quite confident that HTML5 will suffice in obliterating (most, if not all) need for plugins. Google for example has recently announced that they will be phasing out plugin support in their browser starting in 2014, with (initially) the exception of the six most popular plugins, among which Adobe Flash and Microsoft Silverlight (Google, 2013). Mozilla is also already effectively blocking plugins in Firefox, requiring user's explicit permission to run (Mozilla, 2013). Even more interesting is that both Apple (iOS) and Microsoft (Windows 8) have already banned plugins from their browsers in their stock browsers. Another sign of their confidence in HTML5 is that most of these companies are also actively publishing HTML5 related articles and developing technologies that improve the ease of HTML5 development in order to stimulate HTML5 growth and embracement by developers.

Studying web trends it is quite notable that the interest in HTML5 is rapidly rising while interest in both Silverlight and ActionScript (Flash's programming language) decreases (Figure 9).

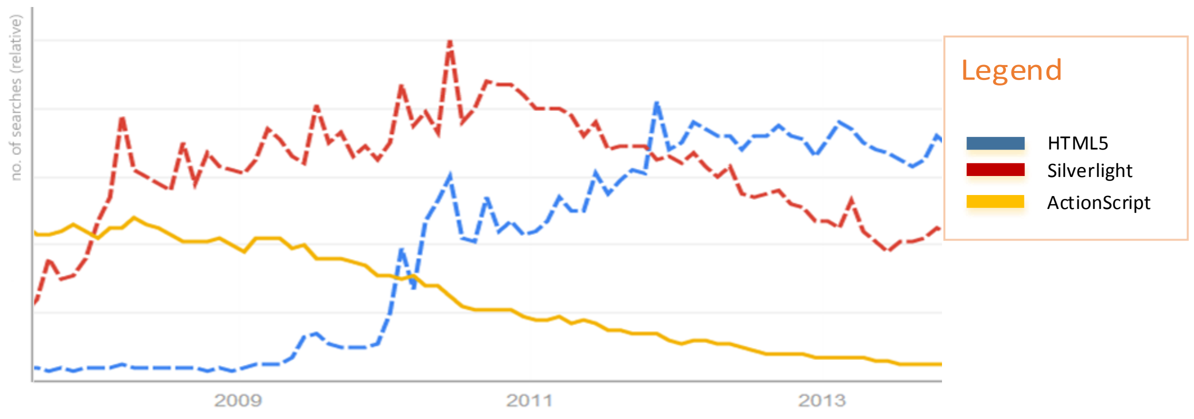


Figure 9. Interest in HTML5 over the years, compared to Silverlight and ActionScript. (Google Trends, 2013)

4.4 What is new in HTML5

When mentioning HTML5, one usually refers to the combined specifications (and implementations) of both HTML5, CSS3 and JavaScript. This thus includes the new HTML elements, but also added *Cascading Style Sheet* elements that come with CSS3 and help website layouts to be more dynamic, and also the new JavaScript API's (and libraries) that are used to exploit these new and other capabilities.

There is a large number of additions that come with HTML5, some of which being of more impact with respect to what web sites were previously capable of then others. Visually representing the most interesting features grouped by type may offer a good impression of what is covered by HTML5 (Figure 10).

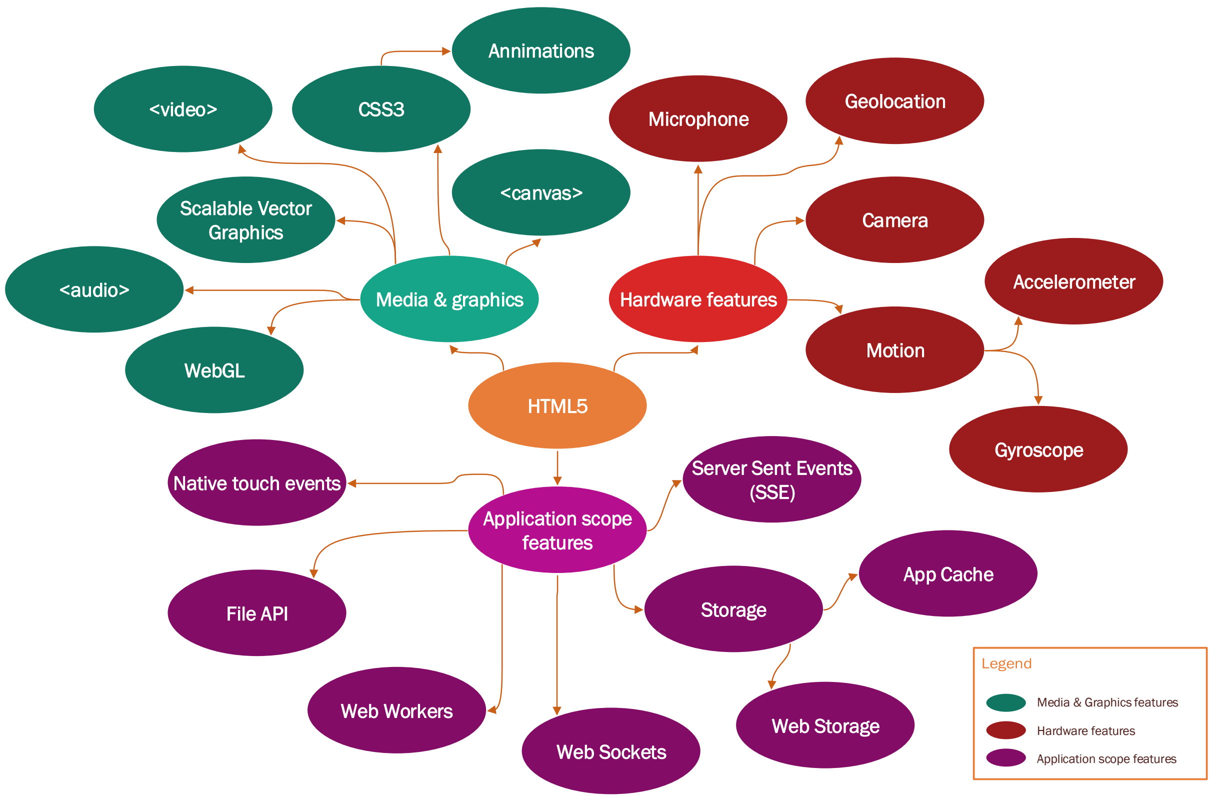


Figure 10. Illustration of the most interesting features that are part of HTML5, using colors to differentiate categories.

MEDIA & GRAPHICS

One of the major catalysts in the success and adaptation of plugins is their ability to stream media and present professional looking graphics. A simple UI animation can influence the way that users experience an application. With HTML5, the browser will natively be able to do all of this. Contributing features are *audio* and *video* elements that enable streaming. Advanced graphics can be realized via *canvas* and inline scalable vector graphics (SVG), very much like what Flash is used for in many current cases. *WebGL* allows even more advanced (3d-capable) graphics by implementing OpenGL. Traditional web site components, such as forms or containers of elements, can also be animated using some of the many additions that come with *CSS3*. A lot of *semantic elements* have been added also.

HARDWARE FEATURES

Another type of feature that an HTML powered web application was incapable of doing without plugins, was accessing (a range of) hardware related features. In order to access the *microphone* and *camera* (whether video stream or simply a photo), a plugin was required. Accessing *motion sensors* and *geographical location* sensors, which are available in an increasing number of devices, was also not possible. HTML5 intends to solve this also. In early drafts of HTML5 there were also suggestions about Bluetooth capabilities (W3C, 2008), but these are no longer included. There is however a possibility that these will be included into HTML 5.1 or HTML 5.2, which are scheduled to gain final recommendation in respectively 2016 and a later year (W3C, 2012).

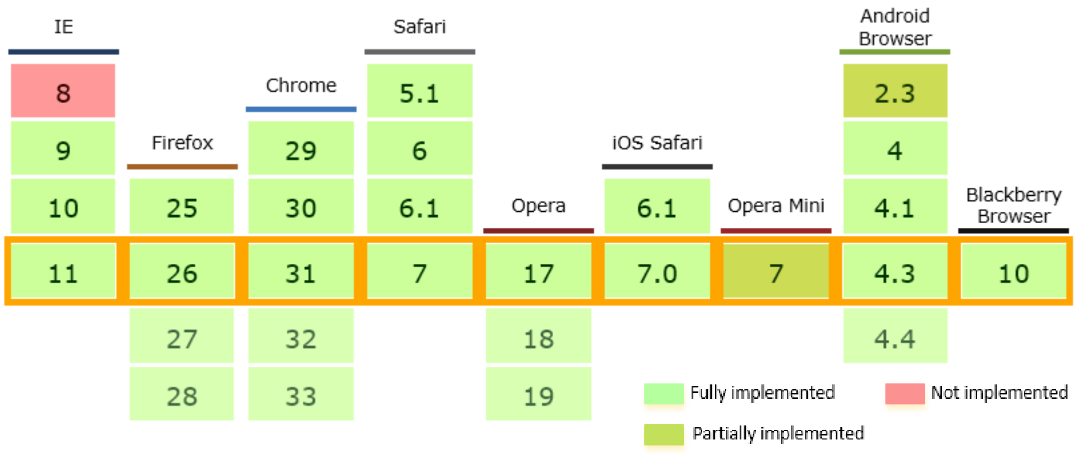
APPLICATION SCOPE FEATURES

The new HTML specification also defines some more advanced, application scope features. Among these are *storage* related features that will allow better cache control and a storage that is supposed to be much securer and faster than the traditional cookie storage, with respectively *App Cache* and *Web Storage*. It will also bring *native touch events* to allow more advanced touch gestures such as ‘Pinch to zoom’, *server sent events (SSE)* to allow more dynamic UI updates initiated from the server, *Web Sockets* to allow advanced communications, a *File API* that allows web applications to access the file system, and *Web W*orkers which introduce a form of multithreading to the traditionally single threaded JavaScript.

More in-depth descriptions of these and other individual features can be found in *Appendix IV: Libraries, Frameworks and HTML5 features*.

4.5 Current Implementation Status

Since the initial specification draft, browsers have gradually been updated with new HTML5 related features. By now, many browsers have implemented the most important additions. For example canvas graphics support (which allows advanced graphics similar to what Flash offers), is implemented in all current browsers, with the exception of Opera Mini, in which it is only implemented partially (Figure 11).



IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
🟥 8			🟩 5.1				▨ 2.3	
🟩 9		🟩 29	🟩 6				🟩 4	
🟩 10	🟩 25	🟩 30	🟩 6.1		🟩 6.1		🟩 4.1	
🟩 11	🟩 26	🟩 31	🟩 7	🟩 17	🟩 7.0	▨ 7	🟩 4.3	🟩 10
	🟩 27	🟩 32		🟩 18			🟩 4.4	
	🟩 28	🟩 33		🟩 19				

🟩 Fully implemented 🟥 Not implemented ▨ Partially implemented

Figure 11. Implementation status of basic canvas support per browser version. (CanIUse.com, 2013)

The more advanced features (having a lower priority) on the other hand, currently have a lower implementation saturation. The ability to access camera and microphone input for example, is implemented in only two major desktop browsers, with the addition of Blackberry Browser (Figure 12).

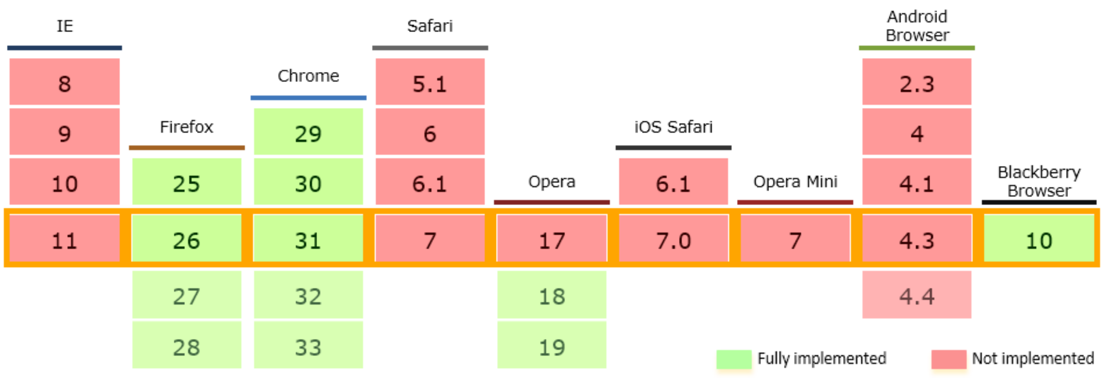


Figure 12. Implementation status of Camera/Microphone per browser version. (CanIUse.com, 2013)

Overall, Chrome and Opera seem to have the broadest implementation of HTML5 features. Firefox also has a high implementation ratio, while Internet Explorer and Safari are slightly behind in implementation progress (Figure 13). With the exception of Opera Mini and stock Android Browser, mobile browsers are pretty much on par with their desktop counterpart. A problem with most of the mobile browsers though is that their version depends on the version of the operating system it is running on. Since these cannot be updated in all cases, many users will be stuck with an ‘outdated’ browser. Android’s stock browser is a good example of this, where quite a lot of users are still on version 2.3, which as a very low number of HTML5 features implemented.

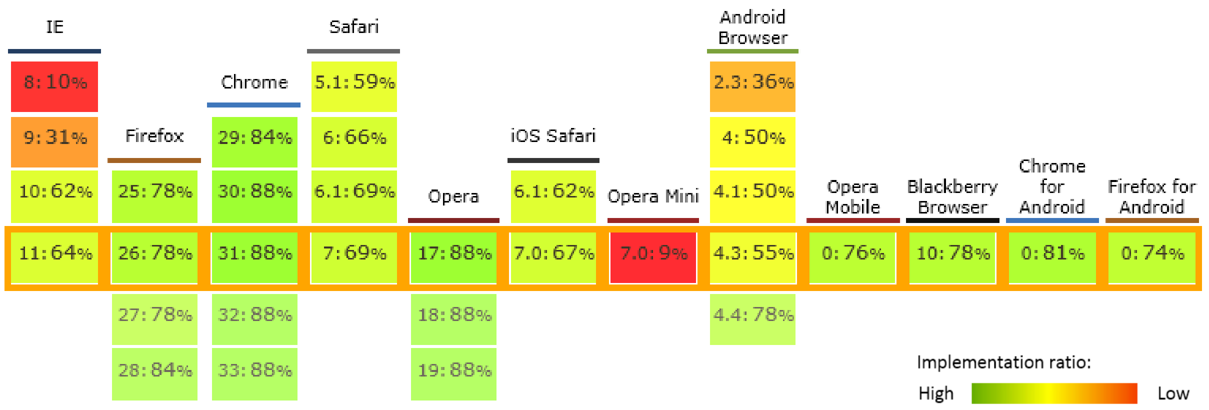


Figure 13. Comparison of total amount of implemented HTML5 features between browser versions. (CanIUse.com, 2013)

While this shows that implementation is nowhere near completion yet, one should not take these numbers too seriously, as most (if not all) HTML5 applications do not require *all* features. When using the more exotic advanced features, the developer should keep in mind that the feature may not be available on all platforms. If necessary, a range of features can even be supplemented with fallback code to support older browsers. A library that helps to do this is *Modernizr*, by automatically detecting which features are supported by the browser and adjusting application behavior accordingly.

More information on the implementation status of specific features can also be found in *Appendix IV: Libraries, Frameworks and HTML5 features*.

5 HTML5 AND SINGLE-PAGE APPLICATIONS

User experience aspects of an application such as reliability, responsiveness, mobility and availability are of obvious importance when designing and developing an application. One way to realize these aspects in a web application is by developing it as a *Single-Page Application* (SPA).

5.1 What is a SPA and why the interest?

Legacy web sites use distinguishable pages to navigate between different parts of the application. It is a simple way to separate content, and the browser already handles the navigation automatically. However, every single time that the user clicks a link towards a next page, or when the browser redirects itself to do so, the user is thrown out of the application experience by a temporarily entering a loading state that in many cases means that a blank page is shown.

A Single-Page Application does not have this problem, since it is what its name suggests; a web application that roots completely within a single page. This means that when the user accesses the web page, the (almost) entire application is loaded. After that moment, it does not need to load again (unless it is refreshed), although specific additional resources that are not needed for the main parts of the application can be loaded asynchronously at a later time in order to reduce the initial loading time.

This benefits the aspects mentioned in this chapter's introduction: it leads to a more *reliable*, *responsive*, *mobile capable* and *continuously available* application. For example, when trying to access a different view of the application (not to be confused with a page) while there is no internet connection, a normal web site would state that the connection has timed out and it could not resolve the address. A SPA has all the resources it needs to check this beforehand inside the application, and will not perform any navigation until it is sure that it is possible. The developer can present the user with a more user-friendly message informing them of the connection loss, and the user (and application) do not need to lose any of the information contained in the application.

To sum up, characteristics of a SPA include:

- It fits on a single web page.
- Fully (or mostly) loaded during initial page load.
- Progressively downloads additional resources when required.
- Important state of application can be persisted on client.
- Maintenance of navigation, history and deep linking.
- Contributes to a reliable and responsive application that is mobile capable and offers a continuously available experience.

5.2 SPA Components

Even though the true definition of a Single-Page Application is limited to its Single-Page aspect, in practice there is really much more to it. SPAs are used to provide rich web applications, and should therefore ideally contain components dedicated to enabling this richness (Figure 14). There are multiple SPA frameworks openly available (two of which will be elaborated on in *Paragraph 5.4*), and most (if not all) of them provide many of these richness enabling components out-of-the-box.

Such components include those that allow *Dependency Injection* (a method of Inversion of Control), *Model-View-ViewModel* or *Model-View-Controller* patterns (MV* in short, a method to separate concerns between UI, behavior, and data), *Data Binding* (a way to map UI elements to data), Views (the 'pages' of a SPA), *Routing* (the handling of navigation between views), *Templating* (reusing the same UI definitions), *Promises* (a pattern that improves asynchronous behavior), *Object State Management* (tracking and submitting changes), and more.

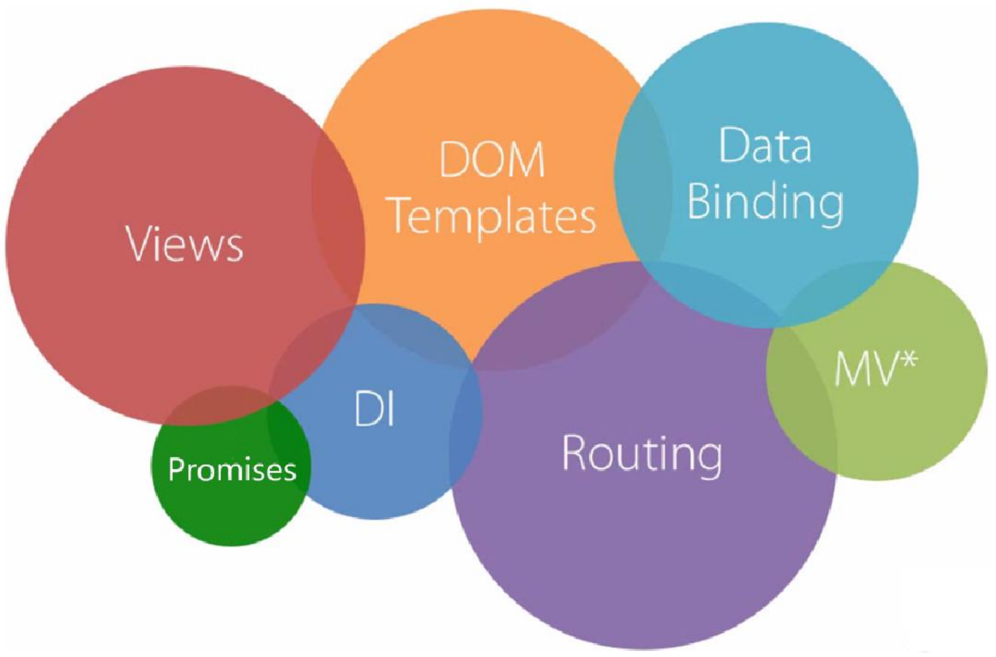


Figure 14. Illustration of how different components can form a SPA. **(Pluralsight, 2013)**

Most of these are enabled by JavaScript libraries that are (or can be) included in a SPA framework, others are enabled by the SPA framework itself. *Paragraph 5.4* will mention a few of these libraries, more detail on their usage and the patterns which they implement can be found in *Appendix IV: Libraries, Frameworks and HTML5 features.*

5.3 MVVM Pattern

In order to build a SPA with a structured architecture, SPAs commonly implement the Model-View-ViewModel pattern (Figure 15). This is a separation of concerns pattern that essentially makes sure that code can be split up in maintainable pieces. The *View* is the part that defines the user interface and passes through interaction, built up from HTML and CSS. The *Model* is the data that can be a plain old class object (POCO), directly retrieved from a database or web service. The *ViewModel* defines the to-be mapped data, and handles events initiated from the View. Implementation of such a pattern is important for a SPA since it is at the core of the way it handles dynamic managing of the application's contents.

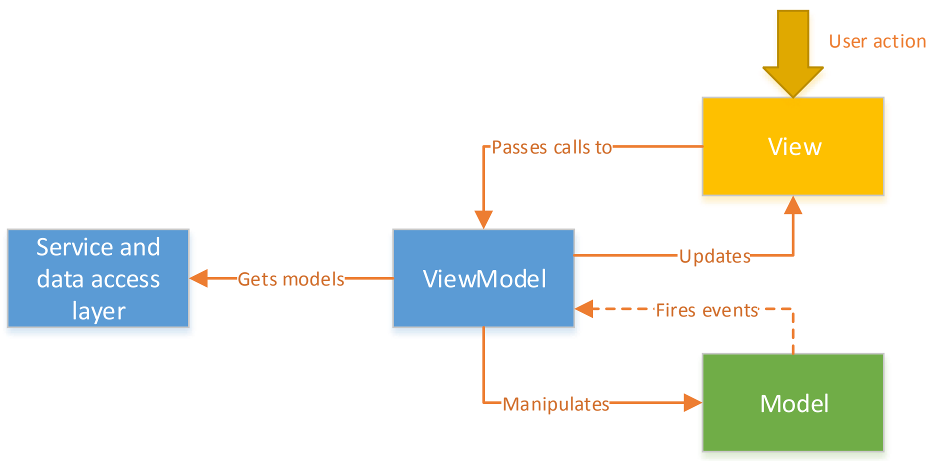


Figure 15. Schematic representation of MVVM (+Services)

A similar pattern is the Model-View-Controller (MVC) pattern, which instead of a ViewModel, uses a Controller. To handle the behavior. Major difference is that in MVVM, the View's state is continuously connected to the ViewModel, while in MVC the Controller initially chooses the View and after that has slightly more passive role.

5.4 HTML5 SPA Frameworks

While it is certainly possible to create a Single-Page Application from scratch using only the core JavaScript APIs (with HTML and CSS supplements), this is a cumbersome process and most likely a waste of resources, since there are a lot open source JavaScript libraries that already implement a lot of mechanics essential to any web application. Most of these have been optimized over the years.

For the purpose of creating Single-Page Applications, there are some JavaScript based frameworks that focus on the sole purpose on providing the tools for building SPAs with (some) ease. The difference between a *framework* and a *library*, is that a library traditionally focusses on a single aspect (such as providing MVVM pattern, or managing navigation), while frameworks are essentially collections of libraries, combining their features in such a way that they can offer a full scale application. The two major frameworks in the HTML5 SPA business are Durandal and Angular. The first being slightly easier to get started with, while the second (recently) gaining the most community interest.

An important aspect that both of them provide as well, is an abstraction of some of JavaScript's flaws. Legacy JavaScript development can be characterized by their exploitation of global variables, raw manipulation of UI components, and manual loading and referencing of scripts. These aspects are much less of an issue thanks to the more structured architecture and abstraction layer introduced by the SPA frameworks. This can be improved upon even more by including a to-JavaScript compiling language like *TypeScript*, which adds type safety and as a result also improved error checking, something of which JavaScript is also characterized as to be lacking.

DURANDAL

Durandal is a SPA framework that combines a lot of popular and commonly used libraries in order to provide relatively easy HTML5 Single-Page Applications development. In contrary to Angular, it almost entirely bases itself on top of pre-existing libraries. The advantage of this is that there is a large community for each of Durandal's components, and also a large number of supplementary libraries that have been designed to interact with these and extend their functionality. Durandal's core libraries are *Knockout* for data binding and MVVM composition, *jQuery* for core functions like user interface manipulation and communication towards services, and *RequireJS* to handle dependency injection using the Asynchronous Module Definition pattern. More on these libraries can be found in *Appendix IV*.

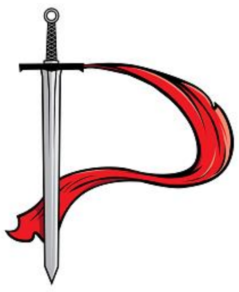


Figure 16. Durandal logo

ANGULAR

Angular is SPA framework by Google that is quite similar to Durandal in many aspects. However instead of building upon pre-existing libraries that have proven themselves in the past, Angular is based mostly on own components. The advantage of this is that the components of Angular have been designed to complement each other, while the components of Durandal have been chosen and configured to do this. The downside is that for support on the individual components, one is mostly dependent on Angular's own documentation. While decent, this documentation seemingly does not provide high detail on all aspects, and one may more quickly find themselves to depend on community sites for specific problems.

Figure 17. Angular logo.

5.5 Which framework to choose?

While this question is very likely to come up when choosing a framework, there is no definite answer. The fact is that both of them provide a good basis and very similar main features, while also each having their own characteristics. The way that data binding has been implemented in Angular is more developer friendly than in Durandal (as it does not require observables to be defined). On the other hand, Durandal's implementation of Model-View mapping and dependency injection is more clear and straight forward than that of Angular.

More detail and usage examples on the individual libraries, their features and design patterns that they may implement can be found in *Appendix IV: Libraries, Frameworks and HTML5 features.*

Both frameworks have been used in development during this project, although Durandal has had the most focus since it was the easiest to platform to get familiar with and been chosen to use for development of the Remote Control application (briefly introduced in *Chapter 3*). A (small) prototype using Angular has been created as well though, illustrating both Angular's code architecture and an alternative and more complex UI to the Remote Control. This prototype will be shown (in brief) in *Paragraph 6.6 Resulting Product*. The Remote Control application that was introduced in *Chapter 3*, will be elaborated on in-depth in the complete next chapter.

6 HTML5 REMOTE CONTROL APPLICATION

As described *Chapter 3*'s project description, the research of this project has been development driven. While the previous chapters have reflected not much of the development aspect, their contents have been fundamental for the development of the *Remote Control* application that was chosen to solve the measurement scenario problem presented in *Paragraph 3.1*. This chapter will elaborate more on this product, the architecture behind it, and some development issues encountered.

6.1 Description and features

The Remote Control's focus and functionality are to provide remote control access to the Web Portal's measurement execution capabilities. In other words; it provides users the ability to remotely start and restart measurements. In future stages the application features could possibly be expanded to offer second screen experiences to other components of the Web Portal as well, or be expanded upon to gradually shift to HTML5 if that is what Sorama desires. This is however not part of the project.

For completion purposes, the application's core aspects are:

- HTML5 based Single-Page Application
- Communicates with existing Sorama services (more on this in the following paragraphs)
- Retrieves and displays (to-be) executed measurements (with data from Domain Services).
- Allows to start and restart measurements and their individual measurement steps on a measurement PC remotely.
- Displays measurement progression.
- Allows to finalize measurement after execution.
- Interface that is mobile-friendly and works both on smaller and larger screens.

Implementing these capabilities, the application will solve the mobility issue presented in *Paragraph 3.1*. With it, the workflow in a measurement scenario where the user is not near the measurement computer at all times (as shown in Figure 6), can be effectively be reduced to one that is more like the scenario in which they are (as in Figure 4). The resulting flowchart (Figure 18) is nearly identical to the non-delayed scenario, apart from requiring usage of the Remote Control instead of the Web Portal. This can be done from any location with local network connection. Pre-configuration still needs to be done using the Web Portal.

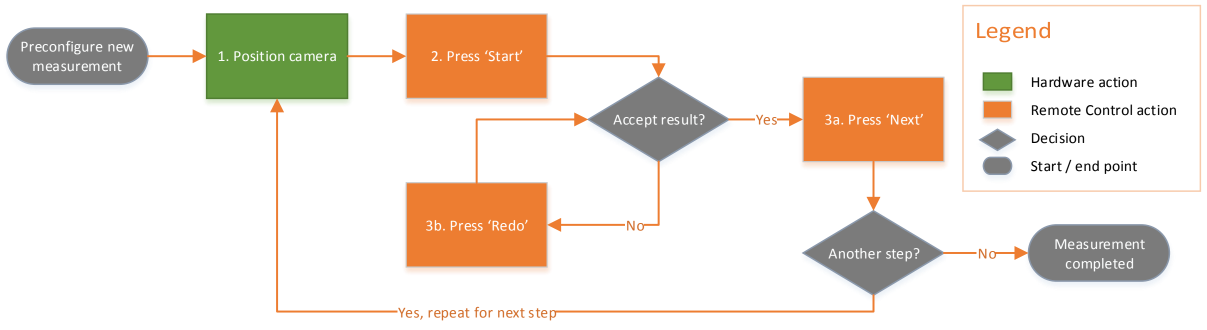


Figure 18. Mobile measurement workflow using remote control. Note that is nearly identical to the one from Figure 4.

6.2 Sorama Imaging System architecture

Since the Remote Control will have to be able to communicate with existing parts of Sorama's Imaging System, it is important to analyze its architecture, before being able to determine an architecture to include the Remote Control in this. The architecture as described in this paragraph is abstracted into the components of interest to this project.

- Domain Services (on server):
  - Authorization Service – web service used to authorize users.
  - Measurement and Project services: web services used to retrieve and persist project and measurement data.
- Acquisition Client (on measurement computer)
  - Sound Acquisition Service – locally hosted web service used to initiate and perform measurement steps using Sorama hardware.
- Silverlight Web Portal (on measurement or other computer) – The Silverlight client software that has been described in *Chapter 2*. Can be used to control measurements if run on the measurement computer, only usable to show results if run on another computer.

The following image (Figure 19) represents how the components communicate on a measurement computer. On a device where there is no Acquisition client installed, its respective component is absent.

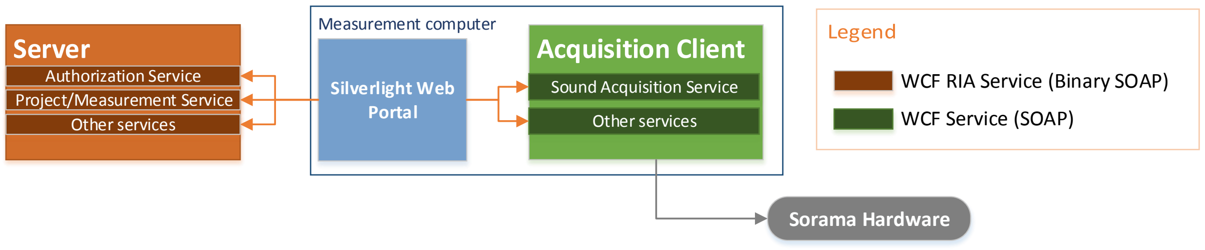


Figure 19. Architecture of SIS components that power the Web Portal.

The Domain Services are all WCF RIA web services binary encoded version of the *SOAP* communication protocol. The web services on the Acquisition Client are plain WCF services (which differ in implementation from the WCF RIA services), and use regular, text based SOAP messages. An analysis of the communication in a measurement scenario can be found in *Appendix II: Sorama Imaging System background*.

6.3 Remote Control architecture

The simplest architecture to use for the Remote Control application (Figure 20) is one that closely resembles the Silverlight Web Portal client described in the previous paragraph. The client will be deployed by the server and will directly consume the same services that the Web Portal also depends on. The major difference architecturally is that the Remote Control does not need to reside on the same device as the Acquisition Client to which it communicates to control hardware (it could though). Also, due to its relatively lower feature count, it depends on less services than the Web Portal.

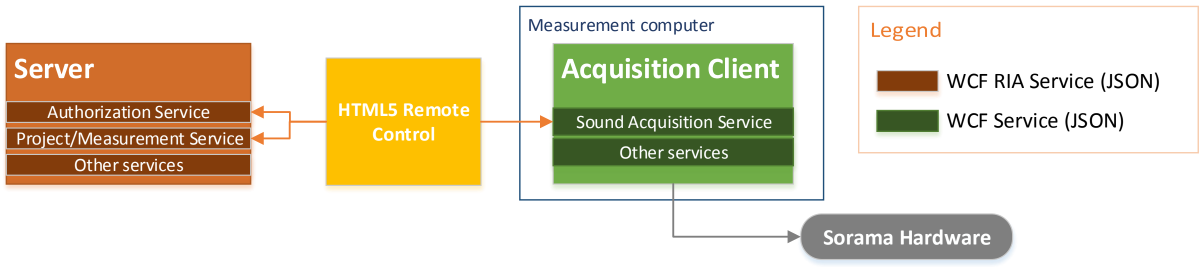


Figure 20. Architecture of HTML5 Remote Control communicating to services

While legacy web sites populate the application's views with data received by host that has deployed it (the back-end), the Remote Control application does not have a dedicated back-end. Its host is only communicated to by the browser to initially deploy the application's core and when necessary some additional resources.

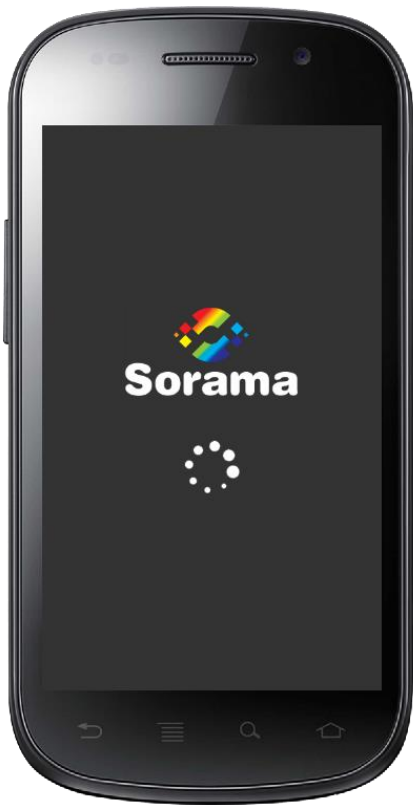
In contrary this website behavior common to most web sites, the Remote Control operates entirely client side, and populates its Views with data requested and retrieved from external services (namely the domain services that reside on the server, and the acquisition services on measurement computer). An inevitable issue here is that this communication is considered to be cross-origin, since it communicates to software residing on other hosts than its own. Cross-origin communication is blocked by browsers and web services by default to prevent against certain malicious attacks, so in order to allow this behavior for an applications, some alterations to the services and the way they are used are required. With HTML5, clients have gained a method to implement this relatively easily, referred to as Cross-Origin Resource Sharing (CORS). This subject required some dedicated research during this project. More on CORS with respect to Sorama services can be found in *Appendix II*.

6.4 User interface

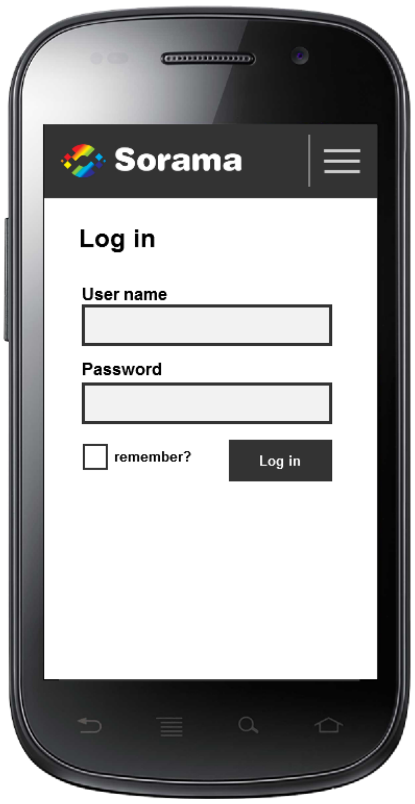
To come up with a design for the Remote Control application (Figure 21), a number of four design goals were used as guidance:

1. Design that **resembles the Web Portal**. Use the same color pattern and similar navigation and actions. This should benefit familiarization and usability in combination with Web Portal.
2. **Flat and minimalistic design** like Windows 8's Modern UI, just as the portal is based on. No three-dimensional elements or buttons, in contrast to many web sites that (currently) use this. No unnecessary use of space-filling components and no unnecessary actions and/or navigation either.
3. **Mobile First** design, meaning that the design is from a mobile point of view (all elements should fit on a mobile screen and work with touch gestures, and no mouse dependent features either).

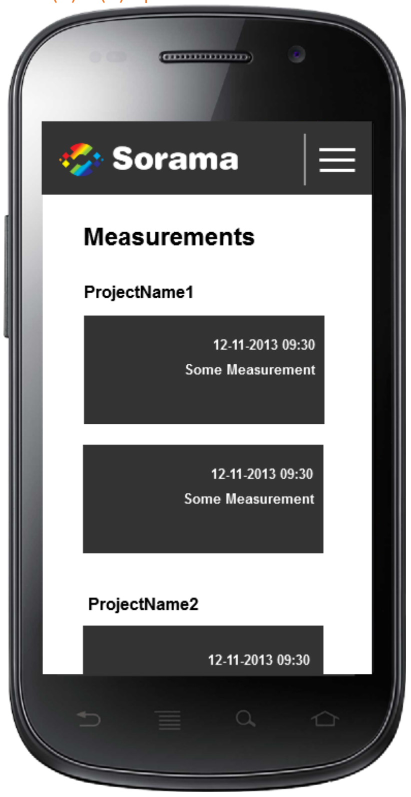
Figure 21. Remote Control UI Design



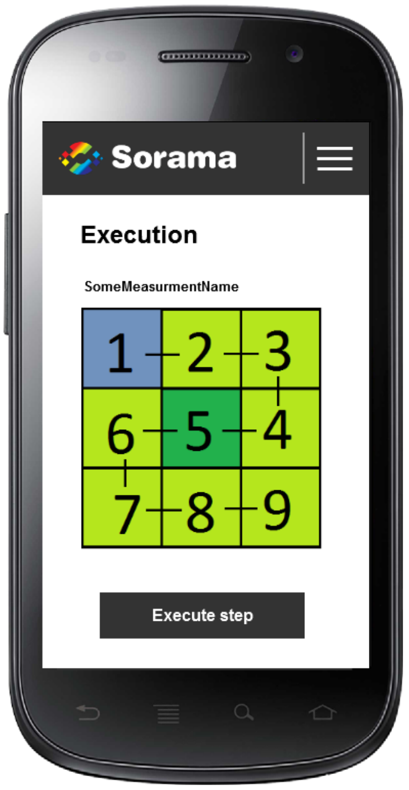
(a) (a) Splash Screen



(b) Login Screen



(b) (c) Measurement Overview



(c) (d) Execution view

As clear from the designs in Figure 21, the Remote Control features a total of four main views, of which the fourth is the most graphically challenging and interactive (although the simplicity in the design may suggest otherwise). The user can choose a to-be-executed measurement step, a feature which the Silverlight Web Portal does not feature (as of yet). It also improves on the complexity of the execution view compared with its Silverlight counterpart on other fronts. For one, the number of components has decreased from seven to two, while it should allow more usability and freedom by letting users deviate from the default measurement step order. When executing higher-resolution measurements, the view will zoom-in the viewport for each step block, allowing more focus on the individual steps.

A more substantiated reasoning on the design can be found in *Appendix IV: Remote Control UI Design*.

6.5 Development issues and solutions

While most of the development process have run without too much hiccups, a few development subjects required some dedicated attention. Cross-Origin Resource Sharing, as was already described in *Paragraph 6.3*, is one such a subject. In this paragraph, subjects related to UI graphics, synchronization with the Web Portal, and C# dependencies will be elaborated upon as well.

UI FOR MEASUREMENT EXECUTION

The Web Portal's measurement execution control screen only allows a default step order in which measurement steps can be executed. Since there are scenarios in which the user may prefer an alternate route, it was desired that the Remote Control would allow this behavior. As a result, it has been implemented such that clicking (or tapping) a step sets that step as the to-be measured step. For higher resolution measurements, the steps have sub-steps that differ only millimeters in location. These are therefore tougher to present in the UI, an issue the Web Portal suffers from (Figure 22) as well. In the Remote Control this is solved partially by zooming in on blocks of measurement steps, providing more detail on multi-step measurements with a high resolution than the Web Portal does (Figure 23).

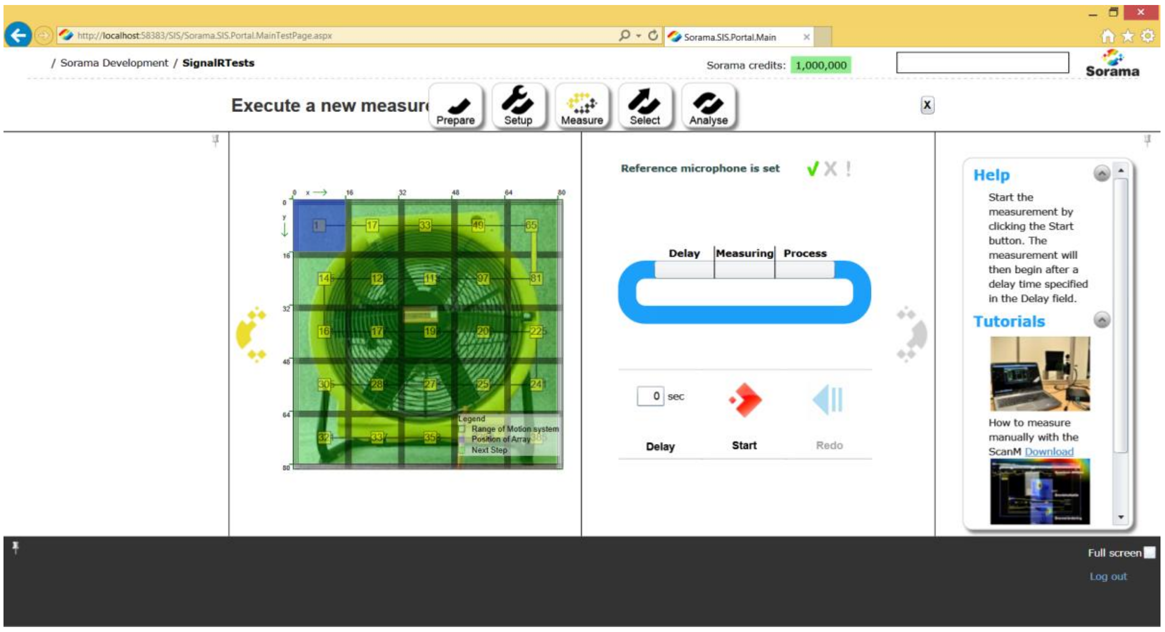

http://localhost:58383/SIS/Sorama.SIS.Portal.MainTestPage.aspx
Sorama.SIS.Portal.Main
/ Sorama Development / SignalRTests
Sorama credits: 1,000,000
Execute a new measure
Prepare
Setup
Measure
Select
Analyse
Reference microphone is set
Delay
Measuring
Process
0 sec
Delay
Start
Redo
Help
Start the measurement by clicking the Start button. The measurement will then begin after a delay time specified in the Delay field.
Tutorials
How to measure manually with the ScanM Download
Legend
Range of Motion system
Position of Array
Next Step
Full screen
Log out

Figure 22. High resolution measurement consisting of a total of 400 measurement steps. The thin black lines represent sub steps, which leads to a very crowded representation.

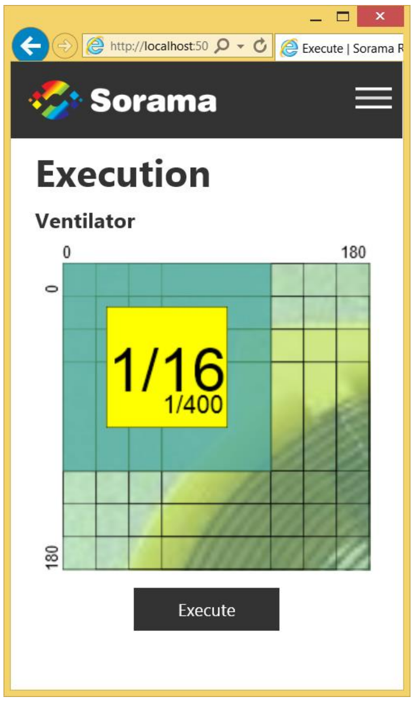


Figure 23. The same measurement in the Remote Control, zoomed in on one step block.

For the highest resolution measurements this is still not sufficient to be able to present enough detail. Execution for these resolutions is still supported, but the visual representation can be improved with more dynamic behavior. The current behavior which zooms in on a block of measurement steps, will in the highest resolution measurements (with 100 or 200 steps per block) crowd the view in a matter similar to Web Portal's issue with less high resolution measurements. Since these high resolution measurements are usually performed with an automated movement system to position the camera anyway, the necessity for a clear representation is less significant though. This is why it has been chosen to focus on more useful aspects instead, such as synchronization with the Web Portal.

SYNCHRONIZATION WITH SILVERLIGHT WEB PORTAL

The initial version of the Remote Control that was developed did not offer any real time synchronization between the Silverlight Web Portal and itself. Of course they used the same data set, but alterations made on one platform were not reflected automatically on the other and required a manual refresh. This is not an ideal scenario, one reason being that the Silverlight Web Portal does not have the required state management to be able to resume a measurement after partially performing it using the Remote Control. When using the Remote Control to perform measurements, one had to complete the entire (not partial) measurement, and then restart the Web Portal in order to view the result.

A solution to this problem was found with the technology *SignalR*, which offers a dynamic and relatively easy method to implement server-sent events. Since the Acquisition Client software uses self-hosted web services to expose itself to both the Silverlight Web Portal and the HTML5 Remote Control, it offered the perfect place to enable real-time synchronization. Using SignalR, the Remote Control's usability combined with the Web Portal was improved in a number of ways:

1. Configuring a new measurement on the Web Portal automatically opens it on the Remote Control.
2. Executing a measurement step is reflected in the Web Portal's execution view.
3. The Web Portal's execution view does not offer a way to deviate from the default step execution order. Using the Remote Control does allow this in some extent.
4. After remotely finalizing the measurement, the Web Portal automatically navigates to the results view.

SignalR's capabilities can be used to improve both the Remote Control and the Web Portal's features even further. One could think of features like real-time feedback during a measurement (which is estimated currently), notifications when attaching or disconnecting hardware, or more synchronization between the different clients.

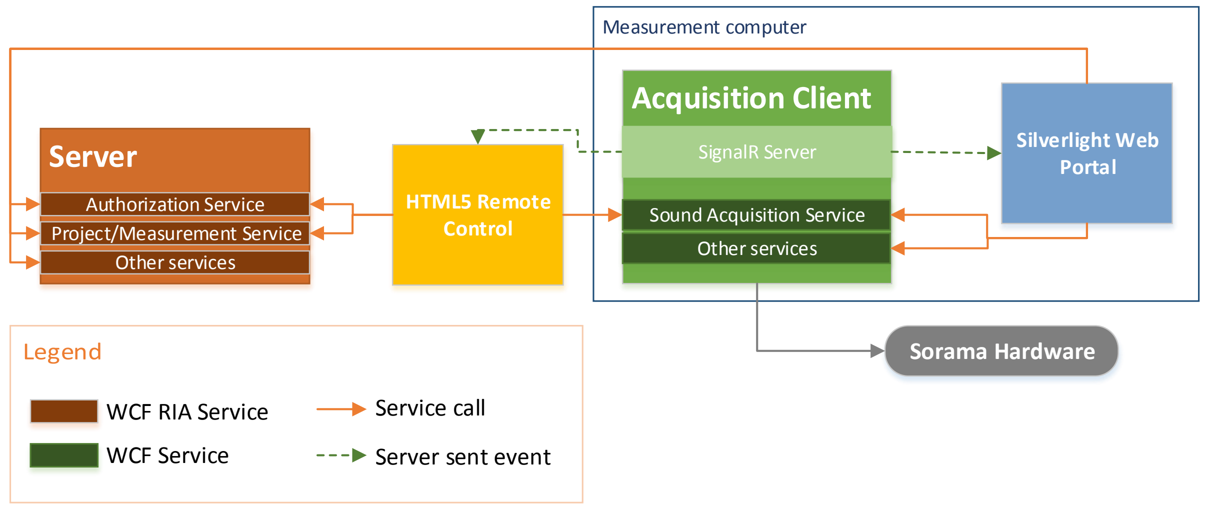


Figure 24. Architecture showing both clients and synchronization via SignalR.

C# PROJECT DEPENDENCIES

The majority of Sorama's software is written in C#. This not only includes their algorithms and code to manage measurements, but also the services and models on which the Remote Control depends. A problem the comes with this, is that when changes are made to any of these services or object models, it might impact the functioning of the Remote Control, in which case the JavaScript code must be maintained as well.

To reduce these dependencies that the JavaScript based Remote Control has towards C# code, an additional C# based component resided on the measurement computer can be introduced as a dedicated service for it to communicate with (Figure 25). This reduces the number of services that the Remote Control depends on, by delegating behavior to this dedicated component. This component is more easily maintainable and benefits from code reusability as well, and can be provided with a web service type that is more easily compatible with JavaScript service consumption than those communicated with in the current setup. For a *WebAPI* web service for example, there is a popular JavaScript library (called *Breeze*, see *Appendix IV*) that manages communication and entity management, something which cannot be done as easily with the two types of web services used by the Domain and Acquisition services respectively.

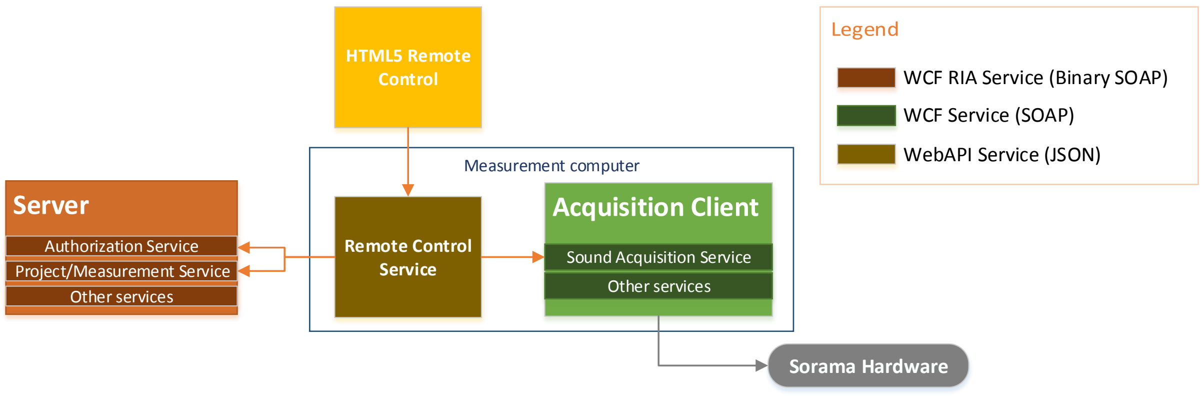


Figure 25. Remote Control architecture with added dedicated component and reduced dependencies.

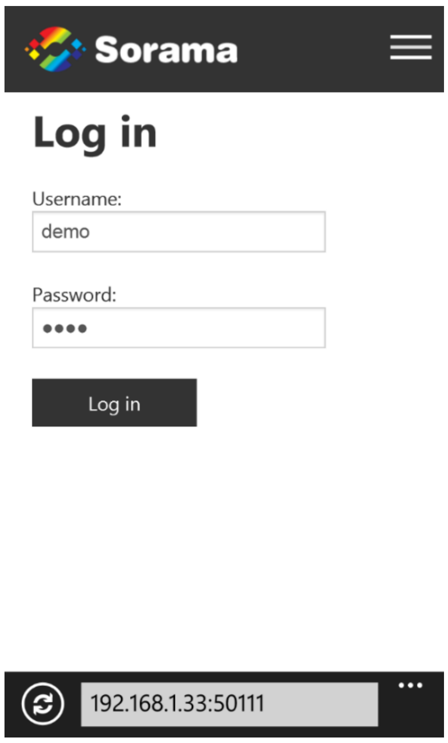
At the time of writing, the Remote Control does not yet use this architecture. However, an Angular SPA Framework prototype (on which will not be elaborated in this report) is using this architecture, which should ultimately be used by the Remote Control too.

6.6 Resulting products

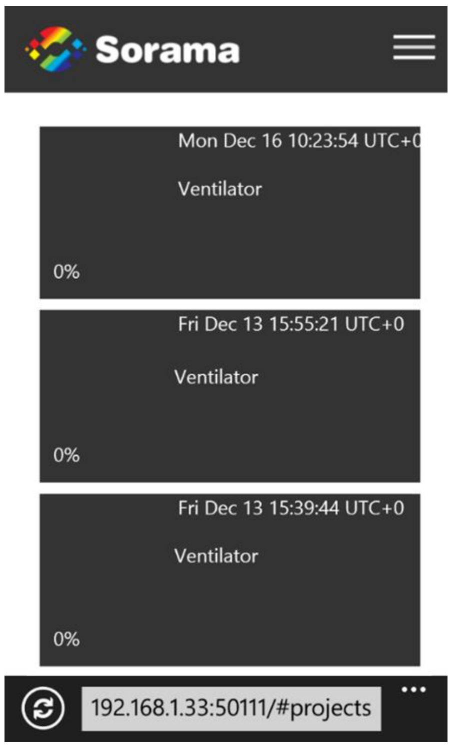
REMOTE CONTROL

The resulting Remote Control (Figure 26) very much resembles the design presented earlier and was created using the Durandal framework. The requirements presented in *Paragraph 3.4* have been implemented, including the lesser priority requirement regarding synchronization with the Silverlight Web Portal, improving the application's usability greatly. At the moment of writing, the product is being optimized for usage, in order for Sorama to be able to use it internally.

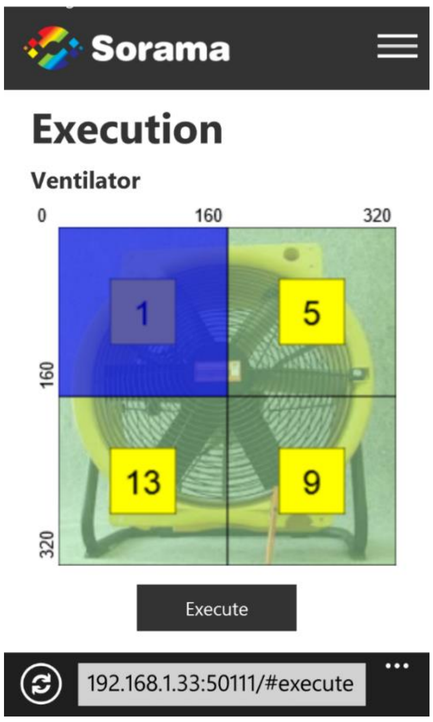
Figure 26 Screenshots of the Remote Control made on a smartphone.



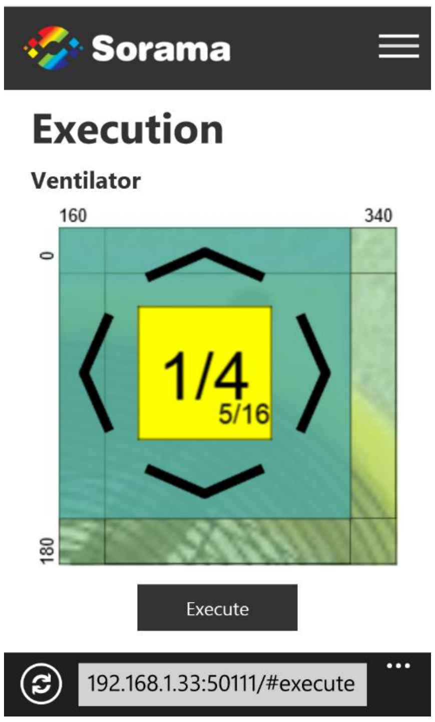
(a) Log in view.


Sorama
Mon Dec 16 10:23:54 UTC+0
Ventilator
0%
Fri Dec 13 15:55:21 UTC+0
Ventilator
0%
Fri Dec 13 15:39:44 UTC+0
Ventilator
0%
192.168.1.33:50111/#projects

(b) Measurement overview.



(c) Full execution view.



(d) Execution view, zoomed in on high-res measurement.

ANGULAR PROTOTYPE

The Angular framework prototype, as stated briefly in the previous chapter, illustrates some interesting features with respect to UI responsiveness. It has been configured to show a more complex UI that adapts itself more visibly than the Remote Control (Figure 27), although the Remote Control contains some responsive UI elements too. Some of these elements may be used when the Remote Control's UI and features are expanded. Its current feature set has no direct need for it, since it only features measurement execution which can be navigated towards via the main view.

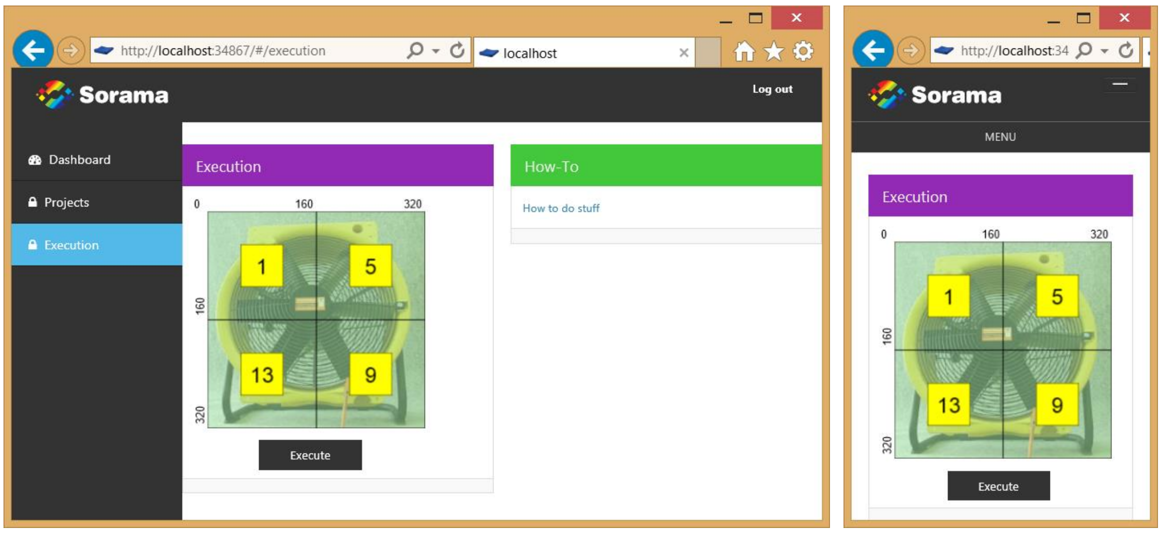


Figure 27. Screenshots of Angular prototype, which contains a more complex UI than the Remote Control. It illustrates a UI implementation that differs more notably between resolutions.

7 CONCLUSION AND RECOMMENDATIONS

DOCUMENT CONCLUSION

One thing that has become clear during this project is that HTML5 is definitely going to be an adequate platform capable of annihilating the majority of browser plugin requirements. The issue with it at this time however, is that HTML5's performance and capabilities are still very much dependent on the platform and browser that the user is running the HTML5 powered application in. Especially on platforms where browsers cannot be updated easily (such as Android), this can be a problem when using certain newer HTML5 features. Rolling out an HTML5 powered application at this time would as a result require Sorama to set hardware requirements for their clients, although this is similar to what they do with Silverlight.

The final version of HTML5's specification is scheduled to be released during the course of 2014, and although it is hard to express an estimation on how long it will take browsers to get on par with their implementations, looking at the rate at which their developers add more features, it will be late 2015 in the best case. It is however likely that it will take longer than that, although most basic features have already been implemented in a broad range of browsers at this moment.

The HTML5 powered Remote Control application developed during this project has shown that HTML5 powered Single-Page Applications can be quite capable of providing rich and responsive multiplatform applications. Two modern major frameworks that can enable this richness, Angular and Durandal, have both been found to provide a structured application architecture, abstracting many of JavaScript's downsides and making JavaScript development more scalable and maintainable than JavaScript has traditionally been known for to be lacking. This can be improved upon by using a to-JavaScript compiling language such as the type safe language TypeScript.

Web development is changing. Many developers are already making the shift away from old technologies and while there are alternatives, many seem to be embracing JavaScript based development. The mentioned frameworks are not only a platform, but a result of this shift as well. Both these and other technologies, among which the tooling such as development environments and UI designers, have been evolving simultaneously with web browsers' adaptation of HTML5. Continuing this trend, it can be expected that HTML5 development will evolve even more over the next few years.

Since Silverlight will have official support until 2021, there is no direct need for Sorama to rush into a migration of their Web Portal. The scheduled decease however means that the question whether to stay with Silverlight or not is insignificant. The true question is when to officially start the move, which is really something Sorama has to decide based on their resources and business plan. On the one hand the move to another platform will require a long-term project, on the other hand the safest bet (resource wise) is to await further maturity of both HTML5 and its tooling and enriching technologies. As in-between solution one could imagine a construction in which an HTML5 environment can be developed and used alongside the Silverlight Portal, until they are on par with features and the need for the old platform has faded.

RECOMMENDATIONS

If the Remote Control is going to be used actively, it may be useful to extend its feature set with additional capabilities:

- Improve the execution's interface to scale for highest resolution measurements as well.
- Include a feature to use the device's camera to set an object image. Currently an image need to be uploaded with the Web Portal manually.
- Include control over motion system to automatically position the sound camera between measurement steps.
- To improve automation even more, one could include an option to automatically execute steps as well (when the motion system has positioned the camera). This will increase usability with the larger measurements even more, by no longer requiring the user to manually start execution in-between tasks. Once the final step has been executed, the application will notify the user. This feature may be interesting for the Web Portal as well.
- Include a feature to alter measurement parameters. If the Web Portal has been closed after configuration of the measurement, this measurement can no longer be accessed from the Web Portal (unless it has been finalized using the Remote Control).
- In addition to the previous bullet point, possibly also add the ability to create measurements via the Remote Control to remove the need for simultaneous use of both applications.

Recommendations with respect to the Web Portal:

- In addition to the previous two recommendations, the Web Portal could be improved with measurement state restoring behavior, allowing one to continue a measurement at any point after the Web Portal has been closed.

The technology SignalR that was researched and used for synchronization between the Remote Control and Web Portal may prove useful for other features as well, since it enables communication from server to client:

- Communicate real-time measurement status updates. The duration of a measurement is currently estimated, which is not fully adequate since duration does not only depend on the camera but also on the measurement computer.
- Notify Web Portal of hardware events, such as connection or disconnection of hardware.

8 EVALUATION

In the end, there are many things left that I would have liked to focus on as well. HTML5 SPA development is a very broad subject and there are countless libraries out there that might have been of interest, but could not be included within the timeframe that I had for this project. The Remote Control application that was developed could have been expanded on with many additional features, but at some point one has to draw the line between aspects that are feasible in the given time frame and those that are not.

While stated in the conclusion that it is not so much a question about which technology is the more powerful but more of when to start the shift, I would have liked to been able to do some lower level performance tests between Silverlight and HTML5 performance as well. This would have required focusing more on Silverlight, and as a result less on the HTML5 and Remote Control part of the project. Comparisons can be found however, and interestingly many seem to indicate that HTML5 is actually very capable of outperforming Silverlight in terms of both processing needs and memory overhead. Netflix' HTML5 version of their video streaming service for example uses about 30% less resources than its Silverlight counterpart, with an equally smooth performance, if not better.

Despite of the elements that I was not able to include in the project, I am content with the result. I have learned a whole lot about HTML5, JavaScript and SPA development, all of which topics on which I had no experience yet, but was now able to use to create a product that may actually be useful. I also learned about the technologies Silverlight, SignalR and WCF and got a subtle taste of some of the many topics in sound imaging.

9 REFERENCES

Adobe. (n.d.). *Security Bulletins and Advisories*. Retrieved from Adobe.com: http://helpx.adobe.com/security.html

CanIUse.com. (2013, december). *Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers.* Retrieved from CanIUse.com: http://caniuse.com/

Foley, M. J. (2010, 10 29). *Microsoft: 'Our strategy with Silverlight has shifted'*. Retrieved from ZDNet: www.zdnet.com/blog/microsoft/microsoft-our-strategy-with-silverlight-has-shifted/7834

Google. (2013, September 23). *Saying Goodbye to Our Old Friend NPAPI*. Retrieved from The Chromium Blog: http://blog.chromium.org/2013/09/saying-goodbye-to-our-old-friend-npapi.html

Google Trends. (2013, december). *Google Trends*. Retrieved from Google Trends: http://www.google.com/trends/

Joel. (2011, May 6). *MVVM vs MVP vs MVC: The differences explained*. Retrieved from InPointForm: http://joel.inpointform.net/software-development/mvvm-vs-mvp-vs-mvc-the-differences-explained/

Microsoft. (n.d.). *Silverlight Support Lifecycle*. Retrieved from Microsoft Support: http://support.microsoft.com/lifecycle/default.aspx?LN=en-us&x=8&y=11&c2=12905

Mozilla. (2013, September 24). *Plugin Activation in Firefox*. Retrieved from Firefox Blog: https://blog.mozilla.org/futurereleases/2013/09/24/plugin-activation-in-firefox/

Osmani, A. (2012). *Learning JavaScript Design Patterns.* Retrieved from AddyOsmani.com: http://addyosmani.com/resources/essentialjsdesignpatterns/book/

Papa, J. (2012, July 16). *Building Single Page Apps with Knockout, jQuery, and Web API*. Retrieved from JohnPapa.net: http://www.johnpapa.net/building-single-page-apps-with-knockout-jquery-and-web-api-ndash-the-story-begins/

Pluralsight. (2013). Retrieved from Pluralsight: http://pluralsight.com/

W3C. (1999, December 24). *HTML 4.01 Specification*. Retrieved from W3.org: http://www.w3.org/TR/1999/REC-html401-19991224/

W3C. (2008, January 22). *A vocabulary and associated APIs for HTML and XHTML*. Retrieved from W3.org: http://www.w3.org/TR/2008/WD-html5-20080122

W3C. (2012, December 17). *HTML5 W3C Candidate Recommendation*. Retrieved from W3.org: http://www.w3.org/TR/2012/CR-html5-20121217/

W3C. (2012, September). *Plan 2014*. Retrieved from w3.org: http://dev.w3.org/html5/decision-policy/html5-2014-plan.html

W3Schools. (n.d.). Retrieved from W3Schools: http://www.w3schools.com/html/html5_intro.asp

10 APPENDICES

Appendix I: Project initiation document

Appendix II: Additional SIS and Services Background

Appendix III: Remote Control App UI Design

Appendix IV: Libraries, Frameworks and HTML5 features

APPENDIX I

Project Initiation Document

DOCUMENT HISTORY

REVISION

Version	Status	Date	Changes
0.1	Concept	05-09-2013	Initial draft
0.2	Revised concept	06-09-2013	Changes in approach, products and planning. Additional changes based on Merijn's feedback.
0.3	Revised concept	09-09-2013	In response to Mieke's feedback, the definition of the project (mainly chapter *3.4* but also some of *3.1* and *3.2*) has been updated to allow more project assurance. The planning (appendix A) has also slightly been updated to match these changes and allow some more flexibility.
0.4	Revised concept	11-09-2013	In addition to a meeting with Merijn (with regards to Mieke's feedback), made some more changes in the project definition. Now includes an additional main product (Prototype Application).
0.5	Pre Final	19-09-2013	Slight changes in project definition. Also a few improvements in grammar and spelling.
0.6/1.0	Final	22-09-2013	Described additional sources in *3.6 Dependencies*.

Distribution

This document has been distributed to:

Version	Date	Name	Function
0.1	05-09-2013	Merijn de Jonge	Supervisor
0.2	06-09-2013	Merijn de Jonge	Supervisor
0.2	06-09-2013	Mieke van Vucht	Project assurance
0.3	09-09-2013	Merijn de Jonge	Supervisor
0.4	12-09-2013	Merijn de Jonge	Supervisor
0.4	12-09-2013	Mieke van Vucht	Project assurance
0.5	19-09-2013	Merijn de Jonge	Supervisor
0.5	19-09-2013	Mieke van Vucht	Project assurance
0.6/1.0	22-09-2013	Mieke van Vucht	Project assurance
1.0	26-09-2013	Albert Lak	Project assurance / secondary assessor
1.0	08-01-2014		As appendix to Graduation Report.

MANAGEMENT SUMMARY

Goal of this document

The goal of this document is to define the project, serve as a basis for managing the project, and to allow assessment at the end of the project.

Two main reasons to use this document are:

- To be sure that the project has a healthy basis and the student will commit to the project.
- To serve as a basis to allow student and project supplier to assess and ensure the project's progress and changes.

Cause

The cause for this project is the interest of the contractor into alternative technologies that may support their desired functionality in a future web application. Their current software uses a technology that on the long term should be replaced by one that benefits from continued support from their manufacturer. A more extensive description of the cause and background can be found in chapter *2 Background*.

Global approach

In order to effectively complete the project, the research, which forms the main part of the project, will be split up in multiple research subjects. Doing this, the student can research the individual subjects while avoiding to jeopardize the end result when a certain subject does not lead to the desired results.

The main product of the project is the research document and its associated advice regarding the researched subject(s). In addition, an HTML5 application will designed and developed in order to demonstrate some of the researched subjects. Other products include this project initiation document and the thesis. A more comprehensive discussion of the individual products can be found in chapter *3.4 Products and results*

Project timeframe

The project has a total of 18 weeks available, of which at least 85 days will be spend working on the project. The additional days within this timeframe can be used when parts of the projects have suffered delays or when certain other aspects have not been yet been completed after the 85th day.

TABLE OF CONTENTS

1 INTRODUCTION 4

1.1 Goal of this document 4
1.2 Layout of this document 4

2 BACKGROUND 5

3 PROJECT DEFINITION 6

3.1 Project goals 6
3.2 Chosen approach 7
3.3 Project scope 8
3.4 Products and results 8
3.5 Exclusions 10
3.6 Dependencies 11
3.7 Conditions 11
3.8 Assumptions 11

4 PROJECT ORGANIZATION 12

APPENDIX A: GLOBAL PROJECT PLANNING 13

1 INTRODUCTION

1.1 Goal of this document

This document has been set up in order to capture all relevant information and basic principles regarding the project in such a way that it both defines the goal of the project, will serve as a basis to manage the project and allow evaluation of the success of the project.

This Project Initiation Document (or PID) covers the following fundamental aspects of the project:

- What objectives are desired to be achieved with the project?
- What is the significance of achieving these objectives?
- Which parties are involved with the project management and what are their roles and responsibilities?
- How and when will the aspects discussed in this PID be realized?

1.2 Layout of this document

In order to state which parts of this document might be altered and/or updated after the final version, we can define two parts of this document, namely a static and a dynamic part. The static part is not likely to change dramatically, the dynamic part is very likely subject to change during the course of this project. In short, mainly the project plan will change during the project.

The "static" part consists of the following chapters and additional appendix:

- Background (Chapter 2)
- Project definition (Chapter 3)
- Project organization (Chapter 4)

The "dynamic" part consists of the following appendix:

- Global project planning (Appendix A)

2 BACKGROUND

Sorama develops a technology to measure, analyze and visualize sounds and vibrations transmitted by objects and appliances. This technology allows their clients to have more insight in their products and the possible noises that they produce. As a result, this provides these clients with the possibility to more specifically be able to optimize (parts of) their products.

In order to allow clients to make and access the measurements that provide this insight, they have developed a web portal which clients can use (in combination with the measurement hardware they have been provided with). This web portal provides options to set up and start a measurement, and access and view the results in high detail. Among these details on which one can focus are views of the frequency range in order to spot anomalies, options to focus on a specific frequency, compare multiple views or measurements, colors that differentiate pressure levels, and much more. The portal also handles some of the data processing.

The current web portal fully depends on Microsoft Silverlight, a powerful technology that powers a series of unique and complex controls used by Sorama to provide a user interface with both a high usability and a lot of features. However, Microsoft has recently announced that they have stopped further development of the technology.

For future benefits, Sorama wants to investigate possibilities of a potential migration of their portal to an alternative technology. With HTML5 being one of the modern technologies with growing support and continued development of its feature set, HTML5 is of main interest to Sorama.

In addition, due to modern devices such as smartphones and tablets becoming increasingly more used by both Sorama's clients and other consumers, Sorama also desires to investigate possibilities to allow (better) support for these products. This encapsulates both mobile support of the web portal, which currently is not very mobile friendly (mostly) due to its Silverlight dependency, and possible other usages of mobile platforms for their sound imaging technologies.

3 PROJECT DEFINITION

3.1 Project goals

The primary goal of the project is to be able to advise Sorama about HTML5 as a technology and its potential usefulness with respect to Sorama's activities. More specifically, Sorama desires to be advised about a potential migration of their current web portal to one that is powered by HTML5. The main research question is the following:

How can HTML5 and its feature set be exploited to benefit Sorama's web portal and other products?

This goal can be split up in the following subjects. Both of these are of interest to Sorama and within the scope of this project.

1. HTML5 as potential technology to power the Sorama web portal and future web projects.
2. HTML5 and its usefulness with respect to mobile devices and potential new products.

Both subjects will be split up further into smaller subjects that will be researched individually. These subjects will be described in more depth in *3.4 Products and results*.

In addition to and in support of the research goals, an application will be developed. Its goal is to test and demonstrate the researched technologies, and possibly be of use to Sorama as either a separate application or in support of future projects.

The project will be deemed successful if Sorama has received a clear advice with good justifications regarding the abovementioned subjects. The magnitude of these subjects depends partially on available resources, which is also described in more depth in chapter *3.4 Products and results*. In the case that the researched technologies offer sufficient support to (partially) conform Sorama's requirements, the production of the *Prototype Application* will be deemed as an additional success, as this may be of even more benefit to Sorama in the future.

This project is also the graduation project of Koen Vrijdag, and therefore regulations of Fontys University of Applied Sciences must also be complied with to ensure this project will lead to a successful graduation.

3.2 Chosen approach

The methodology of the Ten Step Plan will be used in order to manage the complete project. This *Product Initiation Document* is the product of the first six steps, which is the orientation phase of the project with in addition a part of the development phase "work planning and organization".

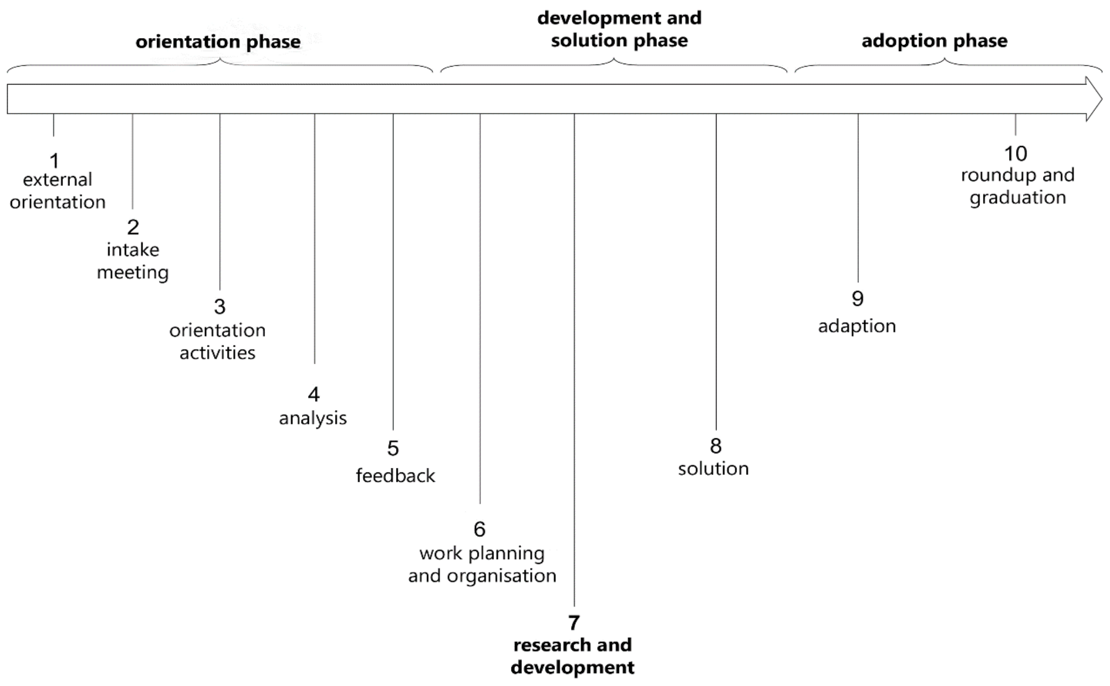


Illustration 1. Ten Step Plan

Steps eight to ten will be covered by writing the *Research* document and *Thesis*. It may also include a possible adaptation of the developed application. Step seven is the major step and this contains the research and development part of this project, a part of which we will now describe in more detail how it will be approached.

In order to provide a structured method of research, the research goals will be split up in smaller subjects. These subjects will be researched individually, and some will only be executed if resources allow it (as described in chapter *3.4 Products and results*). Doing this, the student can focus on different goals which allows for a more organized research, and avoid jeopardizing the end result when a certain subject does not lead to the desired results. In addition to the research, a *Prototype Application* will be developed. This will serve both as a testing platform to support the research, as a method to demonstrate some of the researched aspects and their potential use to Sorama. It may also be of use to Sorama depending on its feature set, which will be determined after a brainstorm session with colleagues. In chapter *3.6 Dependencies*, we predefine sources that may be used to support the research and development.

Because Sorama uses the Scrum methodology with sprints of two weeks for their development process, it is desired that the student also tries to fit his work into this methodology. It is therefore targeted to try and define features for the *Prototype Application* that will be developed during these sprints, and their exact contents will be determined at the start of each sprint. This should allow for additional focus during both the prototype development, as to its corresponding research subjects.

3.3 Project scope

The following parties are involved with this project:

- *Sorama*: The products resulting from this project are created on behalf of Sorama. They are the main party that is involved with this project, seeing that a successful realization will result into an advice regarding Sorama and their future endeavors.
- *Clients*: Sorama's clients are (indirectly) also involved with the project. A successful project may impact (future) products that they may have access to.
- *Software-team*: Using daily Scrum methodology, progress of the project and its activities will be communicated with the members of Sorama's software team. They can also be approached for technical assistance when necessary.
- *Company supervisor*: For specification of the project and support and guidance during the progression of the project, a supervisor within the company has been assigned to help the student define and progress through the project.
- *School supervisor*: For evaluation and project assurance, a school supervisor will be involved with the project.
- *The intern / student*: This is project performer, success of the project is of importance for his (school) career.

3.4 Products and results

This project will be rather research focused, and as a result, aside from this *Project Initiation Document* and the *Thesis*, the main product(s) will thus include *(Research) Documentation*, and, as stated before in chapter *3.1 Project goals*, also a *Prototype Application*. This application will serve as a method to test and demonstrate the researched technologies. Its contents will be determined during a brainstorm session (at a later moment) with other Sorama members, but it ought to fit the research context and be flexible in its definition, meaning that its functionality may either be decreased or expanded, depending on the research results and available resources. Development of this application should not hinder the other main products and its contents will be adjusted accordingly when deemed necessary.

As stated in chapter *3.1 Project goals as well,* the two main goals and research subjects will be divided into smaller subjects in order to separate work and provide shorter term goals to work towards. Each of these subjects will have its own documentation, but these will most likely be bundled in one document.

For both of these main goals and subjects, we will now breakdown the different portions in which the two subjects will be divided.

HTML5 as potential technology to power the Sorama web portal and future web projects

These research subjects will be executed by matter of investigation of available information, and by using the technologies in testing/development environments which may or may not be delivered as small stand-alone prototypes, or included in the described *Prototype Application* main product, depending on the research results.

Because these are the initial definitions of the research subjects, one should keep in mind that the exact contents of the research might (slightly) change during the course of this project, depending on the in-between results. We do however, describe the main aspects of them and their initially determined contents.

- *Research into the realizability of current Silverlight controls with HTML5.*
  In order for Sorama make decisions about the true usefulness of HTML5 with regards to their current software, this research part will focus on the front-end (client side) of web applications using HTML5 and the realizability of current Silverlight web portal controls with other technologies (of which HTML5 in particular).

  The research may include, but will not be limited to: similarities, differences, flaws and benefits in a comparison between Silverlight components and their HTML5 counterparts and/or alternatives, missing functionality in the current HTML5 standard and their implementations, (potentially) interesting libraries that expand on the basic HTML5 controls and/or functionality with respect to currently implemented/planned functionality, mobile features such as resizing and alternative input, and the impact of a (potential) migration with respect to current functionality and components. If available research time proves to be deficient, prioritizing of the subjects may lead to some being researched in less depth, while in the opposite case, research may be extended to include additional subjects, such as comparisons in performance.

  This part will also greatly contribute to the development of the HTML5 based *Prototype Application*, and may potentially lead to few (simple) prototype(s) of current controls, highlighting HTML5's flaws and benefits. These are not part of the main products, but may be referred to in the *Research Documentation*.

- *Research into implementation aspects of (Single-Page) Applications with HTML5.*
  In this research part, there will be focused more closely on general implementation aspects when developing web applications using HTML5 (in contrast to the previous subject, where the emphasis is on the individual controls and features). There will (also) be more focus on the back-end (server side), than in the previous part. In particular, there will be focused on Single-Page Application (SPAs) in combination with HTML5.

  The research may include, but will not be limited to: available, necessary and popularly used technologies, front-backend communication, implementations and libraries in web application development, impact of (possible) platform migration with respect to the current project and its design and potential code reusability, backend structure and database(s). Similarly to the previous research subject, deficient resources may lead to prioritizing of the subjects and some being researched in less depth, while in the opposite case, research may be extended to include additional subjects.

This research part will also contribute to the HTML5 based *Prototype Application*, and may possibly lead to (simple) test applications in order to test and/or demonstrate certain technologies and/or implementations. This depends on the research results and the (estimated) usefulness of these protoypes, and while they are not part of the main products, they may be referred to in the *Research Documentation*.

HTML5 and its usefulness with respect to mobile devices and potential new products

This goal can split up into two parts, which will not be investigated individually (unlike the previous subjects). Instead during the subjects that were mentioned above, there will be some specific focus on the following aspects, as they form are important aspects both to Sorama, and to HTML5 as a modern technology.

- *Research into aspects of developing mobile compatible web applications.*
  During the subject "*Research into the realizability of current Silverlight controls with HTML5"*, there will be focus into how design and development of controls and functionality should occur when (also) offering mobile support of a web application, such as rescaling and touch input. Similarly, during "*Research into implementation aspects of (Single-Page) Applications with HTML5"*, there will be additional focus into whether there are any general aspects in mobile support that should be noted, such as application storage features.

- *Research into usefulness of HTML5 with respect to possible applications of mobile devices for potential new services and/or products*
  There will also be some focus on what additional features HTML5 offers in order for Sorama to develop new types of products and services. The base contents of this sub research will be determined during the brainstorm session as described in order to define the *Prototype Application*. One could think of HTML5s multiplatform nature, extensive support for touch input and possibly other IO features such as microphone or motion support. Consequently, this subject will be manifested for a large part in the *Prototype Application*, demonstrating (some) of these interesting useful aspects and possibly demonstrating an example product with the *Prototype Application* itself.

3.5 Exclusions

Not part of the end result are the logbook and time recordings that are created during the project in the context of internship documentation. Also, a website will be used to manage progress and archive information and decisions. While used for (mostly internal) assistance and for archiving purposes, this website is not regarded to as being part of the end result of the project.

3.6 Dependencies

The research and prototype development will mainly be based on information available on the web. The major dependency in this project therefore is both the quantity and quality of available information. In order to offer some degree of fallback to this, the research may be expended to other sources than regular web information. We predefine the following alternative sources:

- Presentations and/or courses about HTML5 and related technologies.
- Books and/or publications about HTML5 and related technologies.
- Experts in web technologies, such as professionals, students and teachers.
- Colleagues that have knowledge of programming, architecture and/or current system and its work flow.
- Internal documentation about currently used technologies, application architecture, and other interesting subjects.
- Existing web sites and (open source) projects that already exploit (some) interesting technologies and offer (partially) visible source code.
- Forums that can be used to discuss problems and solutions regarding the researched material and the *Prototype Application*.

Another dependency that should be noted is that HTML5 has not yet been fully specified as of this moment and that implementations of the different implementations of browser manufacturers are not complete and they differ between different browser brands and versions. This will be an element of the research, and may influence he contents or the (multiplatform) capabilities of the prototype(s).

Other web technologies that (most likely) will be involved are also considered as dependencies. Primarily JavaScript, the many JavaScript libraries commonly used in web development, Cascaded Style Sheet (CSS), but also web frameworks such as ASP.NET and even IDE's such as Visual Studio may be considered as dependency.

3.7 Conditions

The main condition for this project is that it complies with the requirements as stated by Fontys University of Applied Sciences regarding graduation internships. Among other things, this means that the internship providing company offers a support of at least HBO level, and that the company is not a family company. Also, the school supervisor will visit the company at least two times to check the process and the student will write a thesis and defend his project during a graduation presentation.

3.8 Assumptions

For this project it is assumed that company (Sorama) provides the student with a place to work and access to the resources needed to be able to make progress in his project. It is also assumed that both the student himself, and as well the school and company supervisors will make the effort needed to ensure the project's healthy progression.

4 PROJECT ORGANIZATION

For the different parties that have an active role in this project we will now describe their role and activities.

School supervisor – Mieke van Vucht

The school supervisor's role is mainly to offer global support regarding the internship process. Her main tasks are to define the requirements to successfully complete the project, to provide the student with feedback on products of his internship, in particularly this *Project Initiation Document*, the *Thesis*, and possibly the *Research documentation*. She will also be involved with the grading of the products and she can be approached regarding problems that cannot be solved at the company / with the company supervisor. The school supervisor will visit the company both at the start of the project and at the end of it, during these visits she will monitor the project progression and process.

Company supervisor – Merijn de Jonge

The company supervisor will offer support with regards to the definition and progression of the project. He also acts as client of the project and will, in accordance with the student, determine the scope and requirements of the project.

He will also provide feedback on product documentation and will assess the final products in order to advise the school assessment committee with an appropriate grade. During the visits of the school advisor he will also be present. He will also act as contacted party when the student runs into any problems within or outside of his project.

Other support– Software-team

In the case of technical questions or problems the student can approach colleagues within Sorama, especially the software team may be able to offer support.

The intern – Koen Vrijdag

The intern is the one who is the main actor in the project. He will, among other things, compose a *Project Initiation Document*, perform the research and provide *Research documentation* and write a *Thesis* in order to complete the project. He will also develop the *Prototype Application*. Progression of the project is mainly the responsibility of the intern and in the case of problems he should show initiative to solve them.

PID APPENDIX A: GLOBAL PROJECT PLANNING

Each of the described products and research parts have been included in the following (visual) global planning. Exact time may differ and research focus subjects may be partially exchanged, exact planning will be determined using Scrum sprints, with the global planning in mind.

HTML5 Research																				
Calendar week	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	1	2	3
School week	2	3	4	5	6	7	8	-	9	10	11	12	13	14	15	16	-	-	17	18
Internship week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Orientation and PID	🟨	🟨	🟨	🟥																
Pre research	🟨	🟨	🟨																	
First company visit		🟨																		
Prototype App Definition			🟨	🟨																
Research part 1 (SPAs with HTML5)				🟨	🟨	🟨	🟨	▨												
Prototype Application				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	▨	▨							
Fontys 'terugkomdag'									🟦											
Research part 2 (HTML5 frontend)								🟨	🟨	🟨	🟨	▨								
Research documentation		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨							
Research end sprint												🟨	🟨							
Thesis				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥				
Thesis end sprint														🟨	🟨	🟥				
Second company visit																🟨	🟨	🟨		
Graduation hearing																			🟦	🟦
Round up time														▨	▨	▨	▨	▨	▨	▨

🟨	Planned	▨	Additional time
🟦	At school	🟥	Deadline

APPENDIX II

Additional SIS and Services Background

TABLE OF CONTENTS

Sorama Imaging System 2
Hardware 2
Sorama Acquisition Service 2
Silverlight Web Portal 2
Server 2
Using the Web Portal 3
Cross-Origin Resource Sharing 7
Introduction 7
Cross-Origin Resource Sharing 7
Cross-Domain Service Request 7
Cross-Domain Service Response 8
Credentialed requests 8
Cors on Acquisition 9
Adding headers 9
Message inspectors 9
Problem: Preflight Requests 10
Solution 10
Cors on Domain 10

SORAMA IMAGING SYSTEM

The platform that handles the sound measurement, processes its data and enables users to both control and review measurements, is referred to as Sorama Imaging System (or SIS). It consist of multiple distinguishable components, the ones of importance to this project are described in this chapter.

Hardware

To perform a basic sound measurement, the user needs both a Sorama Sound Camera, a DMADAQ, measurement PC, and cables to connect all components. The Sound Camera is consists of a grid of 1024 microphones that are used to record the measurement data. Via 12 UTP network cables, the immense amount of data is passed through to the DMADAQ, a device that both handles data processing and enables access by the Acquisition client on the measurement computer.

Sorama Acquisition Service

Application installed on the measurement computer that communicates that is used to control the measurement hardware, does some data processing, and exposes a couple of WCF web services hosted as Windows Service in order to expose the application and its functionality to the Silverlight Web Portal (and other potential applications). It has not user interface other than tray icon and the popup that shows the status of the hardware.

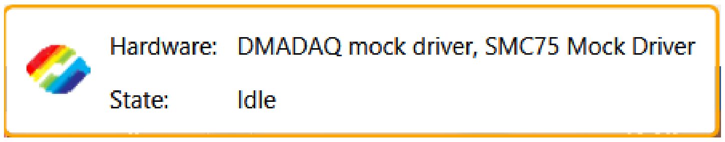

Hardware: DMADAQ mock driver, SMC75 Mock Driver
State: Idle

Figure 1. Status popup of the Sorama Acquisition Service

The WCF web services exposed include services for dedicated types of features. There is a service for Sound Acquisition, but also services for Hologram (sound image data) storage and control over the (optional) motion system that can be used to automatically position the sound camera. The default configuration of the services uses the SOAP protocol.

Silverlight Web Portal

Web application powered by Microsoft Silverlight. It allows configuration, execution, and analysis of sound measurements. This includes selection of measured sound frequency spectrum and focus on and comparison of specific frequencies, for which sound images over the duration of the measurement period (traditionally 1 second) can be viewed.

Server

User and measurement data are stored on a cloud server. They can be accessed via WCF RIA services, by default using MS-Binary encoded SOAP messages to communicate with their clients. This can be an issue when communicating via clients that are not .NET powered, since the use of MS encoding.

USING THE WEB PORTAL

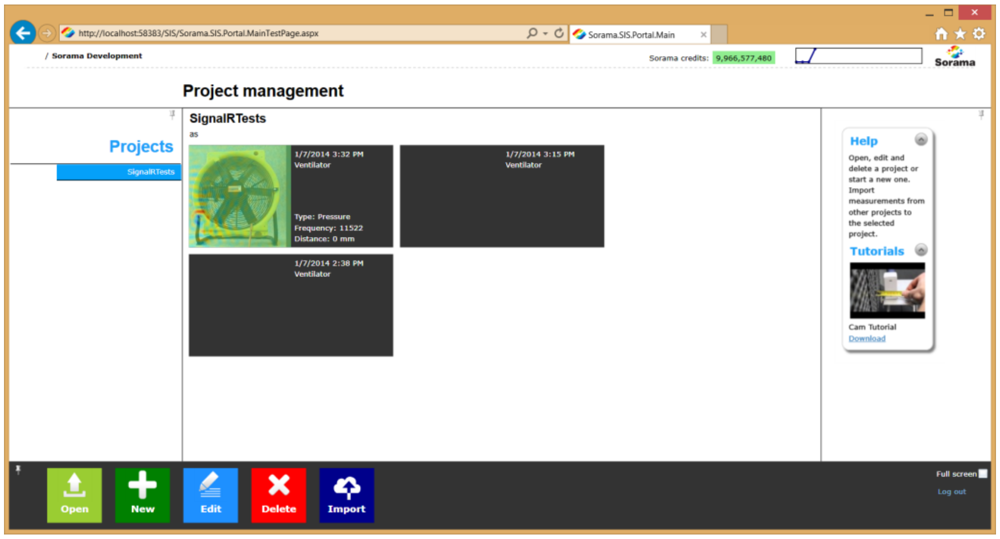


Figure 2. Projects Overview Page

This is the projects overview screen. The left bar contains a list of projects, the center area displays measurements for that project. It uses a design somewhat similar that of Windows 8, using tiles to display the measurements and actions in the bottom bar.

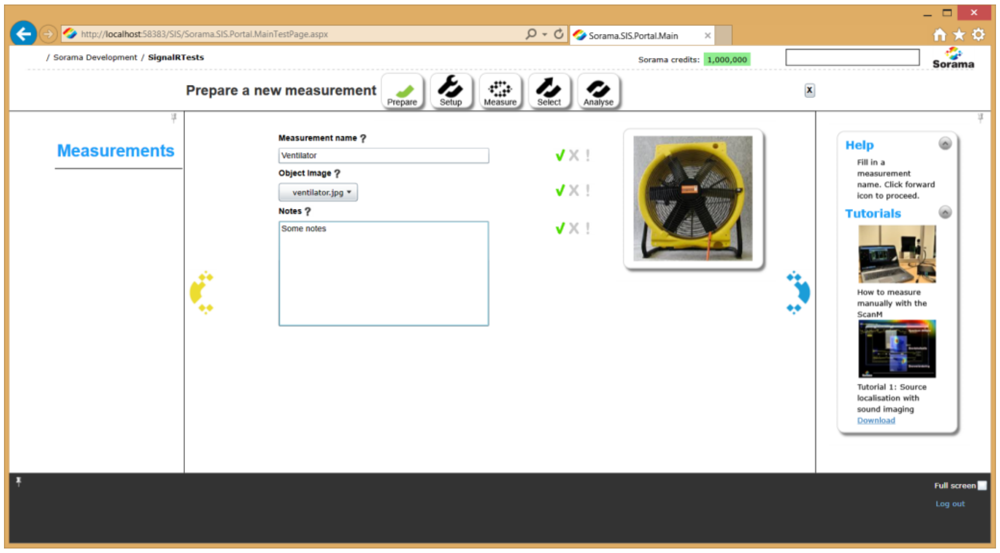
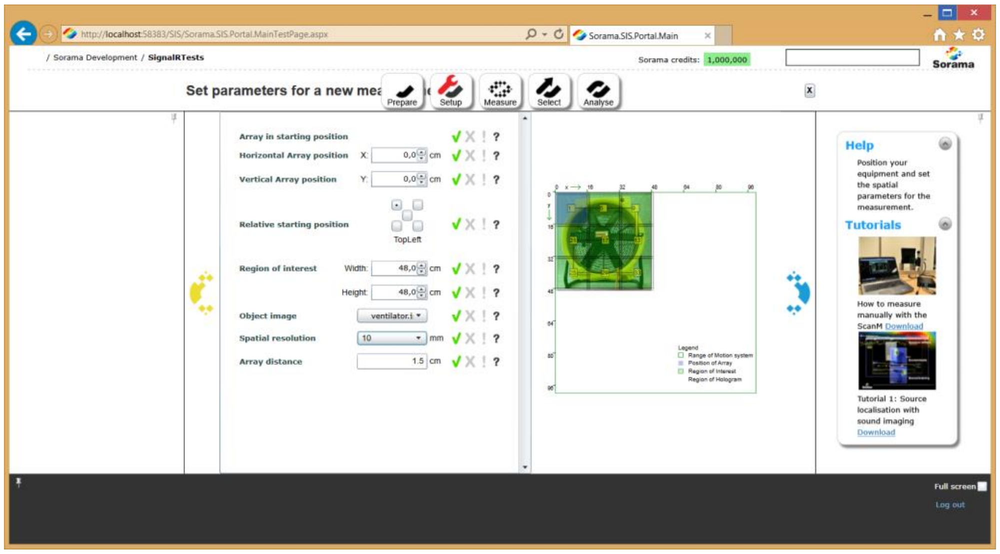

http://localhost:58383/SIS/Sorama.SIS.Portal.MainTestPage.aspx
Sorama.SIS.Portal.Main
/ Sorama Development / SignalRTests
Sorama credits: 1,000,000
Sorama
Prepare a new measurement
Prepare
Setup
Measure
Select
Analyse
Measurements
Measurement name ?
Ventilator
Object Image ?
ventilator.jpg
Notes ?
Some notes
Help
Fill in a measurement name. Click forward icon to proceed.
Tutorials
How to measure manually with the ScanM
Tutorial 1: Source localisation with sound imaging
Download
Full screen
Log out

Figure 3. Preparation screen

This screen allows preparation of a measurement with their name, object photo, and notes.



http://localhost:58383/SIS/Sorama.SIS.Portal.MainTestPage.aspx

Sorama.SIS.Portal.Main

/ Sorama Development / **SignalRTests**

Sorama credits: 1,000,000

Set parameters for a new mea

Array in starting position			
Horizontal Array position	X:	0,0	cm
Vertical Array position	Y:	0,0	cm
Relative starting position		TopLeft	
Region of interest	Width:	48,0	cm
	Height:	48,0	cm
Object image		ventilator.i	
Spatial resolution		10	mm
Array distance		1.5	cm

Help

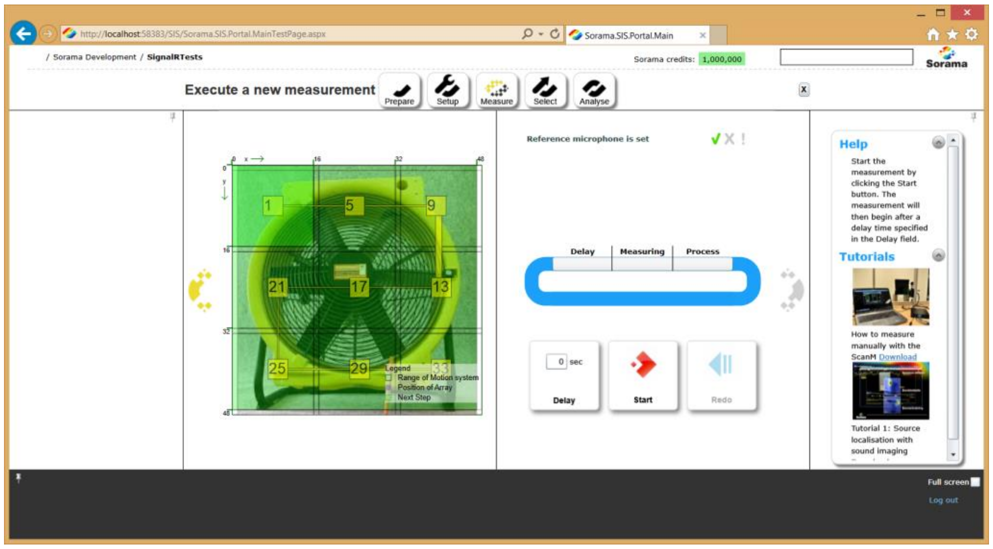
Position your equipment and set the spatial parameters for the measurement.

How to measure manually with the ScanM Download

Tutorial 1: Source localisation with sound imaging Download

Figure 4. Parameter config screen

This screens allows the user to set a number of measurement parameters. Parameters include the dimensions of the measurement and spatial resolution.



http://localhost:58383/SIS/Sorama.SIS.Portal.MainTestPage.aspx

Sorama.SIS.Portal.Main

/ Sorama Development / **SignalRTests**

Sorama credits: 1,000,000

Execute a new measurement

Reference microphone is set

0 sec

Help

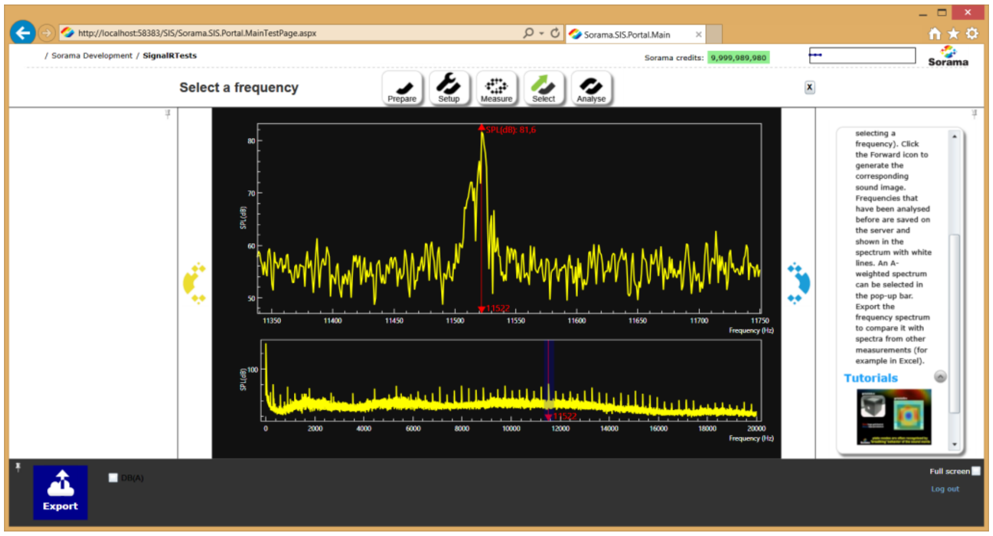
Start the measurement by clicking the Start button. The measurement will then begin after a delay time specified in the Delay field.

How to measure manually with the ScanM Download

Tutorial 1: Source localisation with sound imaging

Figure 5. Execution screen

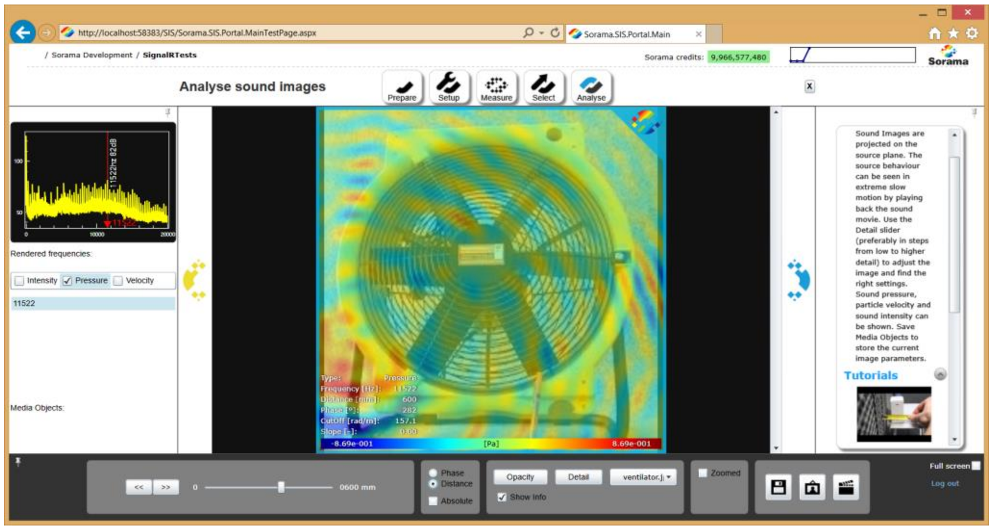
This screen is where the user can start a measurement and execute steps. Individual steps represented on the left, while the progression during a measuring step is displayed on the right.



selecting a frequency). Click the Forward icon to generate the corresponding sound image. Frequencies that have been analysed before are saved on the server and shown in the spectrum with white lines. An A-weighted spectrum can be selected in the pop-up bar. Export the frequency spectrum to compare it with spectra from other measurements (for example in Excel).

Figure 6. Frequency spectrum screen

After finalization of the measurement, the frequency spectrum of the measured data is shown. Bottom graph shows the full 20 kHz, the upper a selected range.



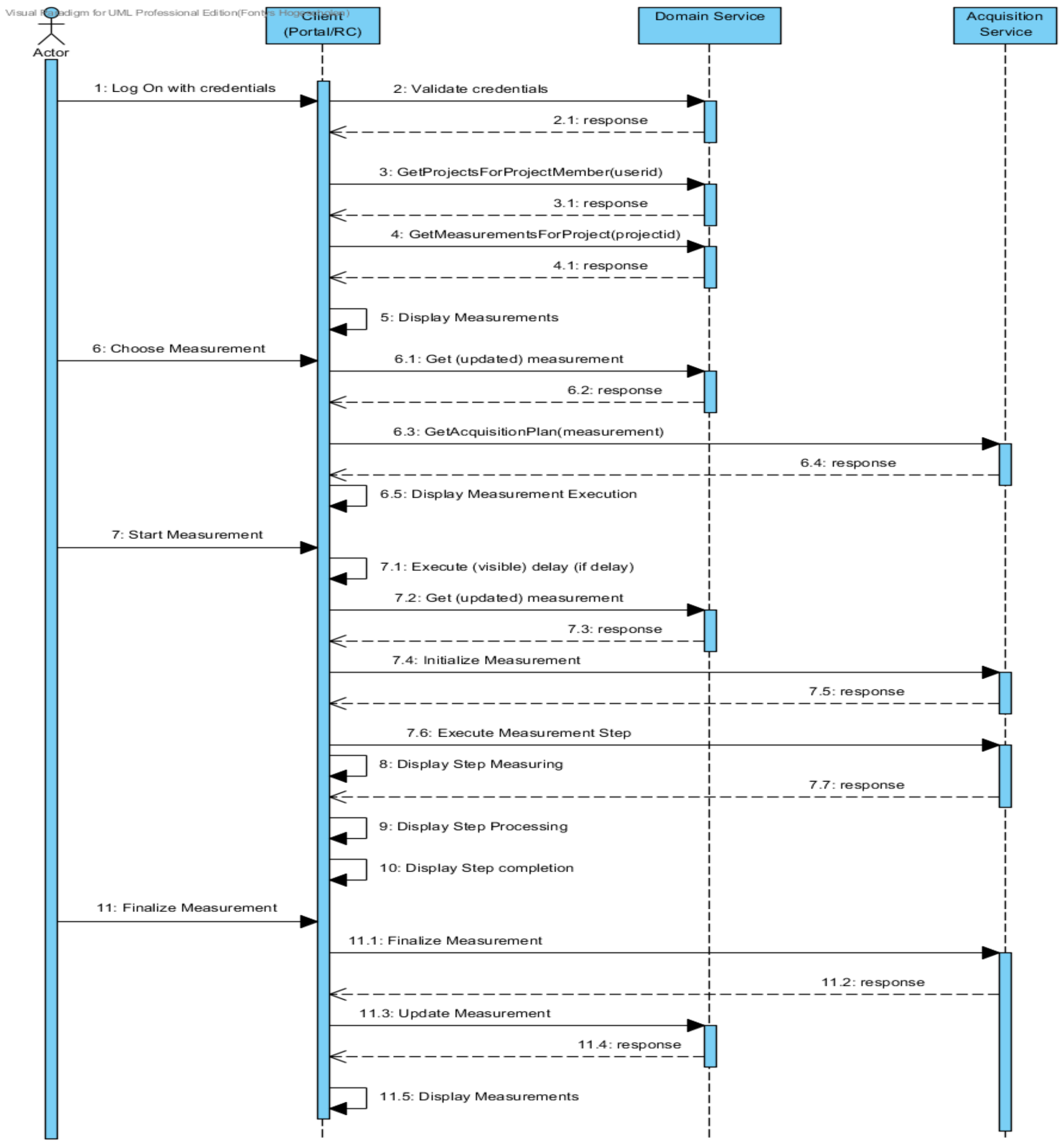
Sound Images are projected on the source plane. The source behaviour can be seen in extreme slow motion by playing back the sound movie. Use the Detail slider (preferably in steps from low to higher detail) to adjust the image and find the right settings. Sound pressure, particle velocity and sound intensity can be shown. Save Media Objects to store the current image parameters.

Figure 7. Sound Image screen

This screen displays a visual sound image at the selected certain frequency. Allows to change multiple parameters to focus on different aspects of the sound. Note that this image does not use real data, and the visual representation does not match the object.

COMMUNICATION SEQUENCE DIAGRAM

The following sequence diagram shows the communication between the Web Portal and both Domain Services and Acquisition Service when logging in and then performing a measurement. The pre-configuration process of the measurement is left out since it is not needed in the Remote Control application developed.


Visual Paradigm for UML Professional Edition(Fontys Hogeschool)
Actor
Client (Portal/RC)
Domain Service
Acquisition Service
1: Log On with credentials
2: Validate credentials
2.1: response
3: GetProjectsForProjectMember(userid)
3.1: response
4: GetMeasurementsForProject(projectid)
4.1: response
5: Display Measurements
6: Choose Measurement
6.1: Get (updated) measurement
6.2: response
6.3: GetAcquisitionPlan(measurement)
6.4: response
6.5: Display Measurement Execution
7: Start Measurement
7.1: Execute (visible) delay (if delay)
7.2: Get (updated) measurement
7.3: response
7.4: Initialize Measurement
7.5: response
7.6: Execute Measurement Step
8: Display Step Measuring
7.7: response
9: Display Step Processing
10: Display Step completion
11: Finalize Measurement
11.1: Finalize Measurement
11.2: response
11.3: Update Measurement
11.4: response
11.5: Display Measurements

The delay-portion of a measurement is executed by the client. The visual representations of 'measuring' and 'processing' are estimated by the client as well, this does (currently) not depend on the acquisition client as of this moment. Data is persisted on navigation. So on navigation towards the Preparation Screen, to the Measurement Execution Screen and after measurement finalization towards the Results screen, data is persisted.

CROSS-ORIGIN RESOURCE SHARING

Introduction

When developing a web application that consumes one or more external web services, the developer will encounter security issues that must be dealt with. This chapter will describe the essentials of dealing with this in a modern web application.

Back in the day, it was unwanted for web applications to access information on another domain. After all, they could retrieve the info they needed via their back-end, which resides on the same domain. Browsers, services, and protocols were designed for this kind of behavior, since allowing communication to alternative domains would open the browser and website for Cross Site Scripting (XSS) and Cross Site Forgery (XSF) attacks, which would be a very serious problem.

With 'the web' -and thereby also web applications- becoming more evolved, there was also an increase in demand of being able to access cross-domain services, even though this behavior was blocked by browsers. A technique that would allow to do this, is **Jsonp**, which stands for Json with Padding. It is a way to send/retrieve Json packages, wrapped in a unique handler and using unique scripts for each call.

CROSS-ORIGIN RESOURCE SHARING

With the introduction of HTML5, another, much more elegant way of communicating Cross Domain was introduced as well: **Cross-Origin Resource Sharing (CORS)**. In essence, this technique requires clients and services to agree on a set of HTTP Headers, which are included with every Http Request. If the client has included certain Headers, and the service dertermines that it is okay with these headers, the service will in turn have to add its own headers before it sends its reply. Finally, the browser will check if the service response in its turn includes the expected headers, after which the response is returned to the JavaScript that handles it.

Thus, to enable cross-domain communication, one needs to properly configure both the client application(s) and the web service(s) it needs to communicate to. If one controls both of these (which we do in the case of our project), this is not a problem. If one only controls the client however, it may be impossible to access it if the service has not been configured to allow your domain.

CROSS-DOMAIN SERVICE REQUEST

When sending HTTP Requests to a cross domain web service, most of the CORS-related work is done by the browser. If using JQuery/Ajax directly, one may need to specify that it communicates cross-domain. In any case, the browser distinguishes two types of requests:

1. Normal request
2. Unsafe request

Simple request, namely HTTP GET or POST requests without custom headers and without content, are considered by the browser as a 'Normal request', and it is sent to the service directly.

Other types of requests, such as POST requests with custom headers, attached content (such as cookies or data), and PUT, OPTIONS and DELETE requests, are considered as potentially unsafe. For these types of requests, the browser will first 'ask' the web service, if it can make the request. It does this by sending an HTTP OPTIONS request: a 'preflight' request. This preflight request defines what contents the actual request will contain, in order for the web service to be able to determine if this is allowed. If it *is* allowed, the OPTIONS request is accepted and the browser sends the actual request. If it is not allowed, the preflighted request is rejected as a Bad Request and the actual request is not made.

CROSS-DOMAIN SERVICE RESPONSE

There is a series of headers related to CORS that must or may be used to configure certain aspects of the communication contracts. The most important header that needs to be included in any CORS server response is *Access-Control-Allow-Origin*, which lists the domains that are allowed to communicate with the service. If it is unknown how many and which clients will communicate to a service, or when it does not matter, the header's value can be set to '**',* which is a wildcard and states that *any* domain is allowed. If the client(s) are known however, it may be wise to limit it to a specific address or IP.

Other CORS headers that may need to be set on a service response are the following:

- *Access-Control-Allow-Methods:* defines which methods a HTTP Request may use (GET, POST, OPTIONS, PUT, DELETE)
- *Access-Control-Allow-Headers*: defines which headers a HTTP Request may include (such as Content-Type, data related headers and custom headers)
- *Access-Control-Max-Age*: defines how long the client is allowed to access the service until it may need to request permission again

CREDENTIALED REQUESTS

A credentialed request, i.e. a request that contains a *cookie*, must be dealt with specifically. Since a cookie is a type of data, it will always result in a preflight OPTIONS request, in which it (automatically) will state that the actual request will contain credentials. The service must then also include the *Access-Control-Allow-Credentials* header set to *true*. Also, and this is something that may lead to annoyances as a developer, the *Access-Control-Allow-Origin* may not be set to**,* but *must* be set to a specific domain, more specifically the domain that made the request. This can be achieved by simply setting the Request Origin as allowed origin on the service. However, when developing, one should be aware that *Access-Control-Allow-Origin* may not be set to *localhost*. The reason for this is simply because *localhost* can be easily be 'spoofed', and thus would be a high security risk if it could be used. Localhost with an additional Port number is in most cases allowed though, since it is considered to be a different domain.

Cors on Acquisition

The services on the Acquisition Client are WCF Web Services, hosted as a Windows Service.

ADDING HEADERS

One way of adding headers to a Request/Response in WCF is with the following simple line of code:

```
    WebOperationContext.Current.OutgoingResponse.Headers.Add("Access-
Control-Allow-Origin", "*");
```

This could be called from the accessed methods directly or indirectly. WCF provides more elegant ways however, as we will describe now.

MESSAGE INSPECTORS

Since WCF does not expose any code or behavior to easily add headers, other than the implementation technique described above, it may be desirable to extend WCF's behavior. One of WCF's great features, is that it offers extend-ability for (some) of its shortcomings. For example, it allows developers to write custom ServiceBehaviors, which can be added to a web service and extend its functionality and behavior. WCF also allows a developer to 'hook' their own code into certain moments in the WCF communication pipeline. These 'hooked' portions of code can be referred to as interceptors or runtime extensions.

The following image shows the extend-ability points in the WCF Dispatcher. As visible in the image, the earliest extend-ability point after receiving a Message (a request) is the Message Inspection. This is where messages get tunneled through and may for example be manipulated, rejected or used to trigger other code.

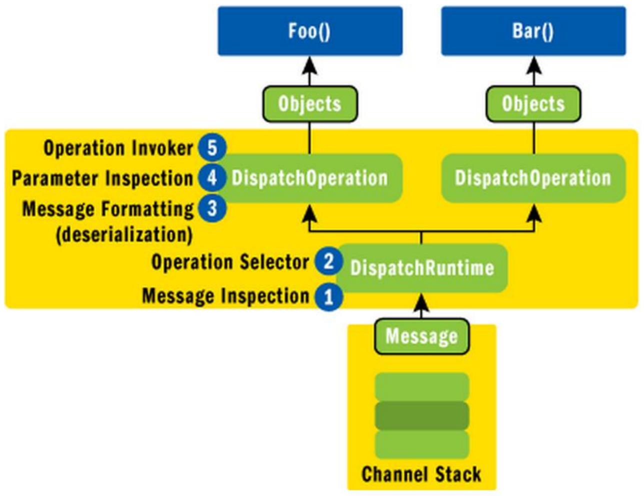


Figure 8. Shows how Message Inspectors can hook into the early part of the WCF channels. *(Händevik, 2011)*

This extend-ability point (Message Inspection) is most likely the most ideal place to inspect and set headers, since it is the first (accessible) point where messages enter the service, and the last place they can be accessed before they are sent out in a response.

PROBLEM: PREFLIGHT REQUESTS

Since cross-domain Http POST requests (with custom headers and/or data) are preceded by a preflight OPTIONS call (in most browsers), the service should be configured to correctly handle these requests.

However, this turned out to be a problem in WCF. As it appeared, OPTIONS requests are rejected by WCF even before they enter the Message Inspection. The reason for this is most likely a design decision; after all, SOAP Envelopes are always send in POST requests, so every other type of request cannot be a SOAP request and therefore not what the BasicHttpBound service (SOAP) should be expecting. From a design point of view, this does not seem an illogical decision. Cross-Origin Resource Sharing had not even been specified when WCF version X was released.

In a blog post about CORS in WCF, Carlos Figueira describes (Figueira, 2012) a method about extending WCF's behavior such that a WebHttpBound service (a Restful service) can handle preflight requests by having a custom ServiceBehavior create specific OPTIONS methods for each method that has been annotated as being CORS-enabled. This led me to the following idea: if OPTIONS-mapped operations could be added in code for a WebHttpBinding, then maybe also for a BasicHttpBinding. The answer for this was disappointing, as it turned out that it is not possible to set the Http Method for a BasicHttpBound service operation.

There seemed only one good option left: Adding a WebHttpBinding to communicate to, without affecting the existing BasicHttpBinding.

SOLUTION

With the code described on the internal website it is possible to add a WebHttpBinding (Restful) web service that consumes and produces JSON packages. Doing this it**is** possible to (without much effort) add functionality to handle preflight requests. This can be done using the method described by Carlos Figueira, or with only one additional method, by adding a WebInvoke mehod mapped to OPTIONS and an empty/wildcard UriTemplate, such that it is triggered by *all* OPTIONS requests and responds with the CORS headers accordingly.

Solution: Add WebHttpBinding (RESTful) Endpoint with either general OPTIONS or ServiceBehavior that 'enables' specific methods as being Cors Enabled.

Cors on Domain

To enable Cross-Origin Resource Sharing on the WCF Domain Service(s), only a slight addition needs to be made. The file Global.asax, which resides in the root folder of *Sorama.SIS.Portal.WebPortal.MVC4*, contains the method *Application_BeginRequest*. This method is called every time a new Http Request is received. This therefore seems like the ideal place to set the needed headers. For Http OPTIONS requests it should set the allowd methods and headers an possibly the Max Age and Cache Control. For all other requests it should set the allowed origin(s) and also set that Credentials are allowed (since most requests require a user to be signed in, and the login request itself also needs this header to be allowed.

```
void Application_BeginRequest(object sender, EventArgs e){

HttpContext.Current.Response.AddHeader("Access-Control-Allow-Origin",
"*");

var origin =  HttpContext.Current.Request.Headers.Get("Origin");
if (origin != null){
   if(!origin.Contains("localhost"))
      HttpContext.Current.Response.AddHeader("Access-Control-Allow-
       Origin", origin);
}

HttpContext.Current.Response.AddHeader("Access-Control-Allow-
     Credentials", "true");

if (HttpContext.Current.Request.HttpMethod == "OPTIONS"){
      HttpContext.Current.Response.AddHeader("Cache-Control", "no-
           cache");
      HttpContext.Current.Response.AddHeader("Access-Control-Allow-
           Methods", "GET, POST, OPTIONS");
      HttpContext.Current.Response.AddHeader("Access-Control-Allow-
           Headers", "Content-Type, Accept");
      HttpContext.Current.Response.AddHeader("Access-Control-Max-Age",
           "1728000");
     HttpContext.Current.Response.End();
   }
}
```

Note that here, the Allowed Header is set to the requests Origin (since a wildcard cannot be used when also using credentials). In the future, it may be desirable to set the allowed Origin to a specific domain in order to protect the service from third parties.

However, when this is enabled, a wildcard in the Allow-Origin header is forbidden. So one should use a specific domain instead (Mozilla, n.d).

The solution to this is to add your local IP to IIS Express server and run the project as Administrator.

REFERENCES

Figueira, C. (2012, May 14). *Implementing CORS support in WCF*. Retrieved from msdn.com: http://blogs.msdn.com/b/carlosfigueira/archive/2012/05/15/implementing-cors-support-in-wcf.aspx

Händevik, D. (2011, June 23). *Supporting Cross Origin Resource Sharing (CORS) requests in a WCF Rest service*. Retrieved from Developing DotNet Architect - Blogspot: Supporting Cross Origin Resource Sharing (CORS) requests in a WCF Rest service

Monsur Hossain, M. H. (n.d.). Retrieved from Enable Cors: http://enable-cors.org/

Mozilla. (n.d.). *HTTP access control (CORS)*. Retrieved from Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/HTTP/Access_control_CORS

APPENDIX III

Remote Control UI Design

General design goals:

- Design that **resembles the Web Portal** design. I.e. same color pattern and similar navigation and actions. This should benefit recognizability and usability together with Web Portal.
- **Flat design** like Windows 8 "Metro", just like the portal is based on. No "3d" elements or buttons, in contrast to many web sites that (currently!) use an iOS6 like design with 3d-like components.
- **Mobile First** design, meaning that the design is from a mobile point of view (all elements should fit on a mobile screen and work with touch gestures, i.e. no mouse-over dependent features).
- **Minimalistic** design. "Less is more". No unnecessary use of space-filling components and no unnecessary actions and/or navigation.

U1: Splash Screen

Shown while loading the main application code and content. Shows logo and loading indicator. **Possibly also title "Remote Control"?**

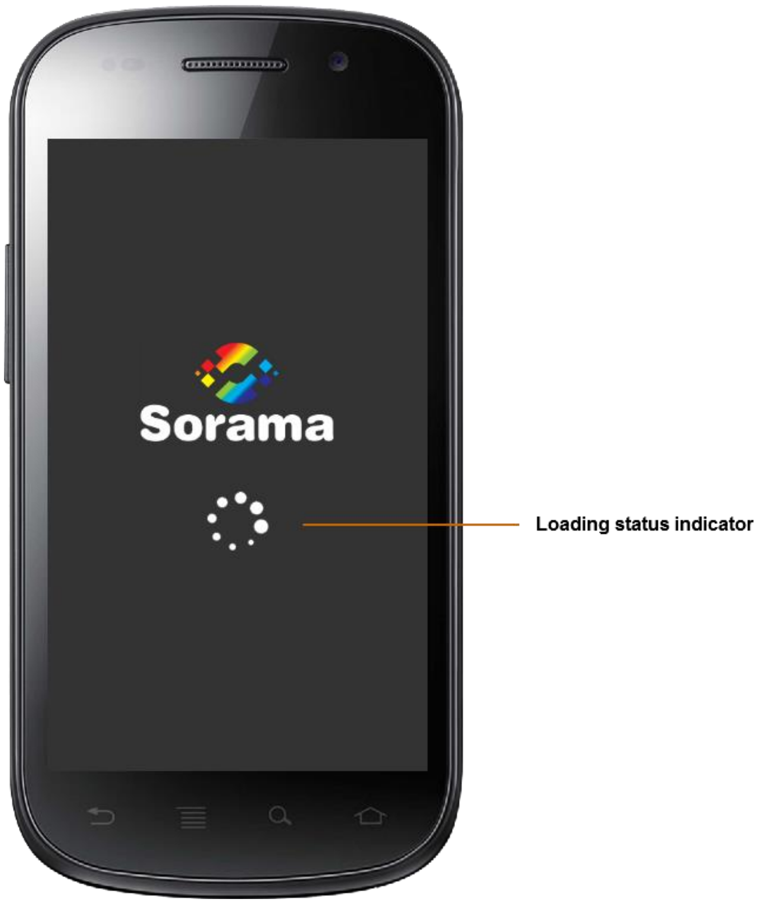


Image 1. HTML5 App splash screen

U2: Login

To ensure a “Single Page Application” experience in which logging in is part of the work flow, we start this experience starting with the log in page. This manifests both in already rooting the menu bar at the top and the views title “Log in” being displayed on the same location on which the title will be displayed on other views. Also, the view-specific controls are grouped within a panel which is padded a few pixels from the borders.

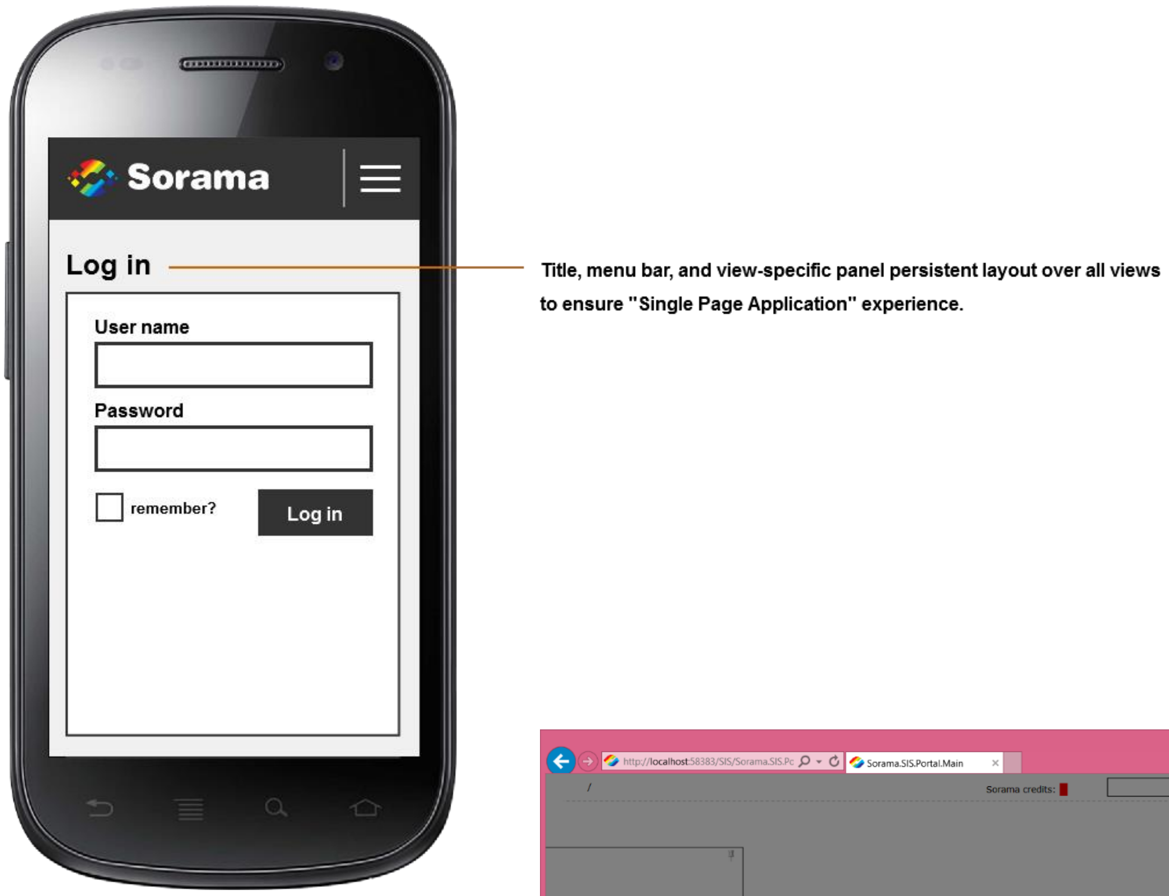

Sorama
Log in
User name
Password
remember?
Log in
Title, menu bar, and view-specific panel persistent layout over all views to ensure "Single Page Application" experience.

Image 2. HTML5 App login screen

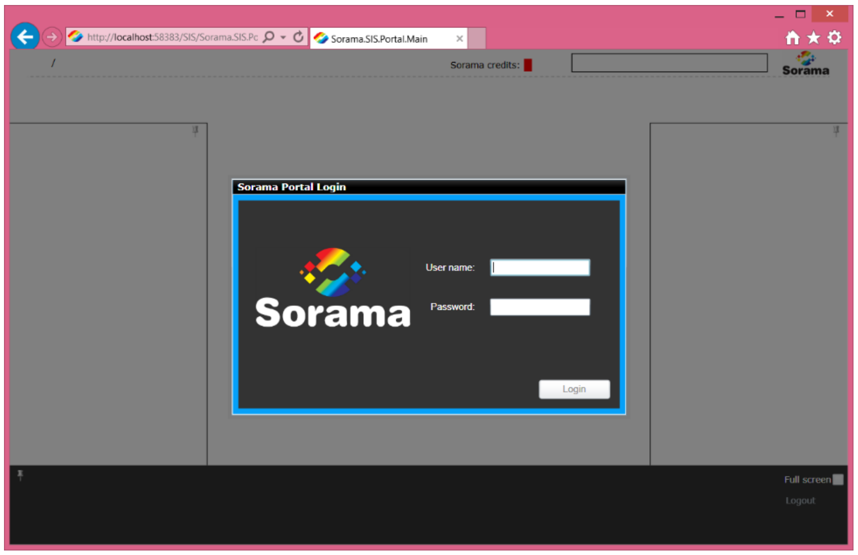
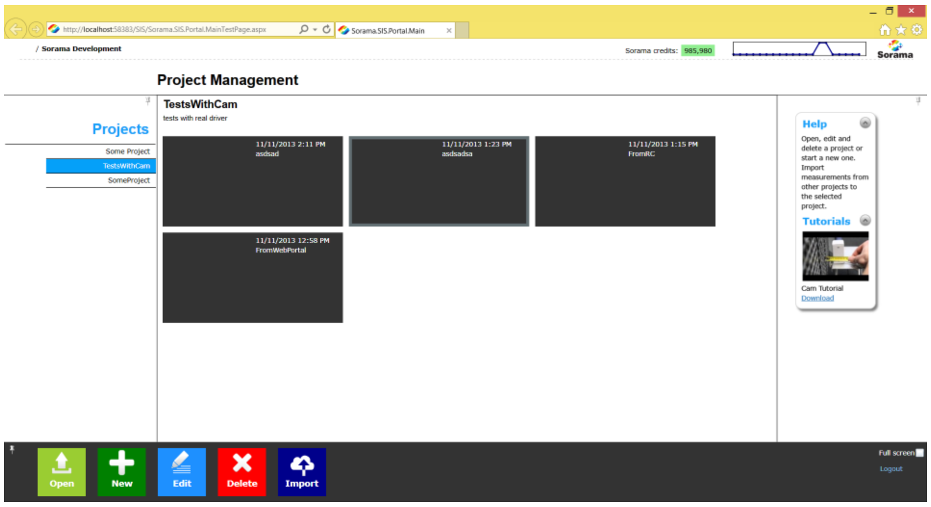


Image 3: Login in Silverlight Web Portal

U3: Measurements Overview

In the Silverlight Web Portal, Measurements are shown for the selected project as follows:



Open, edit and delete a project or start a new one. Import measurements from other projects to the selected project.

Image 4: Measurements per project in the Silverlight Web Portal

Since only “pending” measurements will be shown, the list of projects with “pending” measurements will most likely not be very long. For this reason it seems like a good and navigation-saving idea to combine the “Projects Overview” and “Measurement Overview” from the Web Portal into a single overview for the Remote Control app.

To limit the “pending” list to only recently created measurements, and not also measurements that were aborted during the creation process, we could limit the shown items to measurements younger than 1 day.

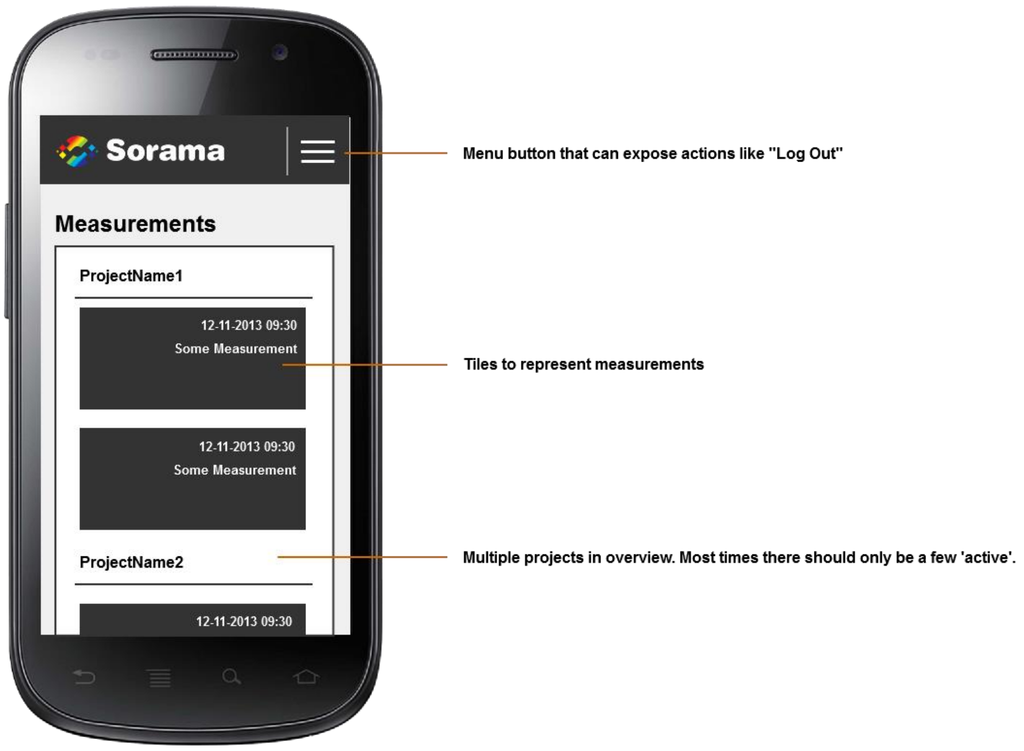

Sorama
Measurements
ProjectName1
12-11-2013 09:30
Some Measurement
12-11-2013 09:30
Some Measurement
ProjectName2
12-11-2013 09:30
Menu button that can expose actions like "Log Out"
Tiles to represent measurements
Multiple projects in overview. Most times there should only be a few 'active'.

Image 5. HTML5 App measurements overview

U4: Measurement Execution

Now this is the most interesting view both in its graphic and feature wise (relative) complexity. Let us first break down the Execution Screen in the Silverlight Web Portal:

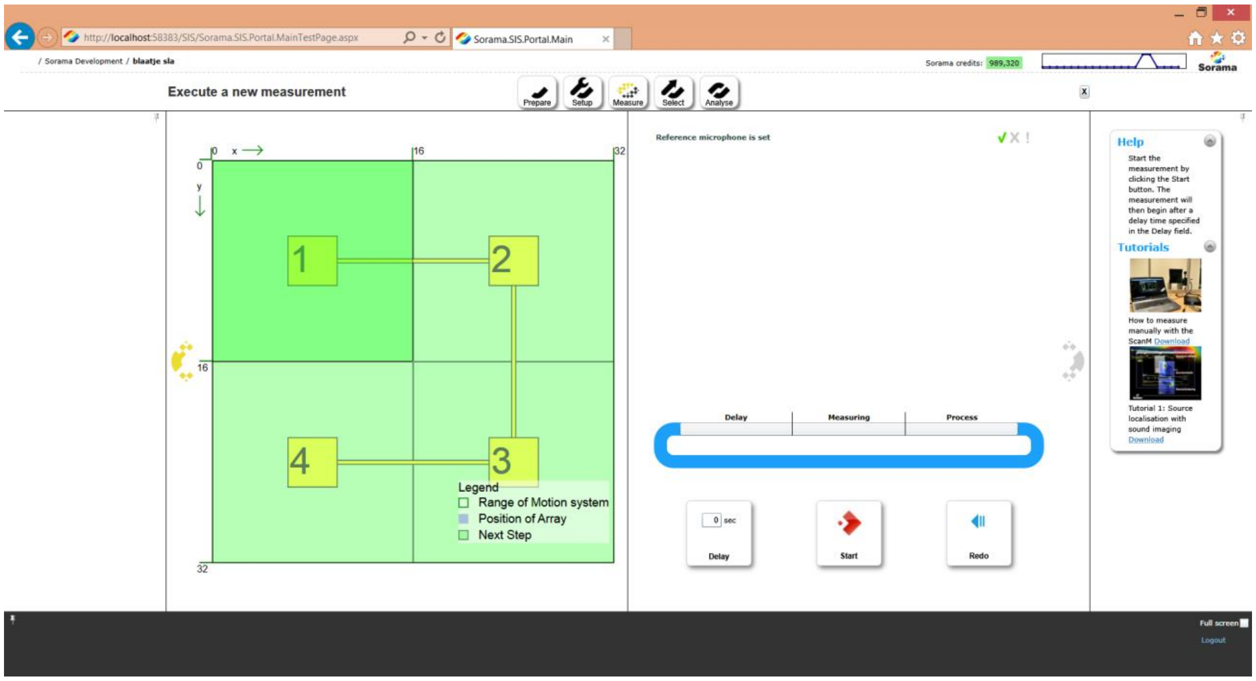


Image 6. Silverlight Web Portal measurement execution screen

Components:

- Start button.
- Redo button.
- Previous button navigating back to the "Measurement Setup" view (the previous view).
- Next button blocked until the last step has been executed, representing "measurement finalization" and navigating towards the Frequency Selector screen.
- Delay textbox to set a delay that is executed before a step is executed.
- Visual grid representing measurement "route" and selected and running execution steps.
- Progress bar showing a "faked" progression of the measurement. (Fake meaning that it bases its time on a prediction instead of true progression.)

Start & redo button – When giving the user the ability to effectively choose the to-be-executed step, they can choose a previously executed step as well and a dedicated "redo" button would then be redundant. Can be *combined into a single button*.

Previous button – By *implementing browser navigation* the user can navigate by using these *instead*. This reduces components and may even present as a more "expected" behavior as most users expect a web site / web application to navigate to the previous page/view by using this button.

Next Button – Is needed. But maybe the “start/redo” button can be *recycled* to show “Next” or “Finalize” after all have been executed. When reselecting another step at this point, the button would return to its “start/redo” behavior until that step has been either executed or deselected.

Delay textbox – *Redundant* seeing as this is implemented in the portal to allow the user to “move away from the measurement PC” and the remote control will allow him to “already be away from the measurement PC”.

Progress Bar – Since we focus on a mobile-capable application. It is important that components do not need more space than required to avoid scrolling and padding. The “circular progress bar” in the Web Portal is nice, but there are possibilities that are more compact or do not even need another component.

- A single bar as progression bar. Requires less vertical space.
- No bar, but use the “busy icon” to indicate that is “doing something” and in this case “executing a step”.
- Use the Visual Grid to show the progression**. My proposal**: Instead of having the “executing step” display a solid blue color, have it “fill up” gradually, indicating how far the step is executing.

Visual Grid – The most interesting component. In the Web Portal it works as follows:

- All measurement steps are distinguished by green rectangles, with a number and a connected line illustrating the route in which they have to be executed.

- The “current” measurement step is highlighted with a brighter green color.

- The most recently executed step is displayed in blue, another type of blue while executing.

- In-between execution steps, a repeated animation shows which step will be executed next.

- User cannot interact with the grid or alter the execution route or “selected” execution step.

Visual representation of all states, with from left to right:

1. Before executing a measurement step, first “selected”. **2.** While executing first step. **3.** After executing first step, repeated animation towards second step. **4.** After re-executing first step, no more repeated animation, but second step marked as “selected”. **5.** After all steps have been executed.

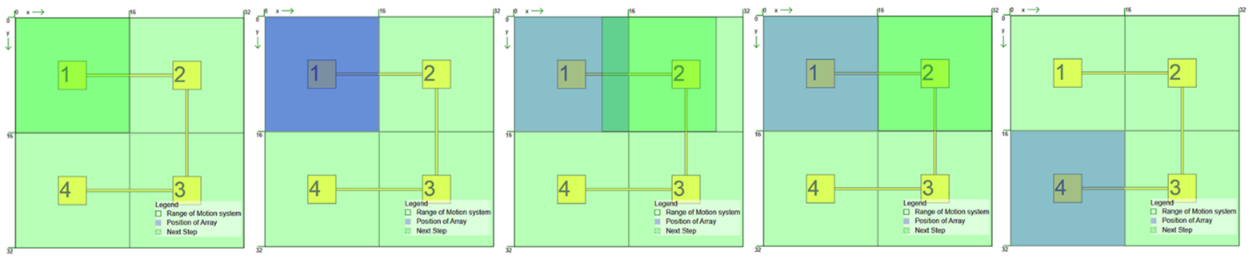


Image 7. Execution animation in Silverlight Web Portal

Since it is desired that the user can (optionally) indicate which step it wants to execute alternative to the default route (as discussed with Merijn), some alterations to the (familiar) behavior may be desirable as well.

Animation: Instead of repeatedly illustrating the route from the current step to the next step, I suggest that this is illustrated only once or twice. Continuously displaying the animation may suggest it is "only" an animation and not interactive.

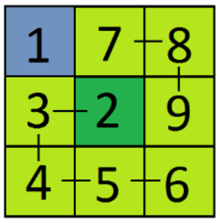
Executed marker: Instead of only marking the most recently executed step as "executed" (light blue), I suggest that all previously executed steps are marked as executed. Since we want to allow the user to select which step to execute, it should be useful to know at any point during the execution process which have been executed.

Executing marker: Instead of marking the currently-executing step with dark blue, I suggest that this is animated as a square that fills up from bottom to top, effectively becoming the loading/status bar.'

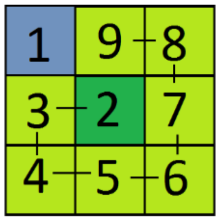
Alternative route: Any step can be selected for execution.

This could either or not impact the suggested route and effect the numbers and connectors displayed to show the execution path.

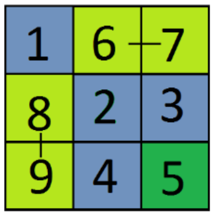
1. **Impact on suggested route:**
   - Using the same path determination as in Web Portal. Familiar route, but may lead to 'leaps' from one end of the area to another. As shown in this image, the route after selecting the center step continues towards the bottom like the usual Web Portal routing (with 2-6), but then it leaps back towards the top to continue 7-9.

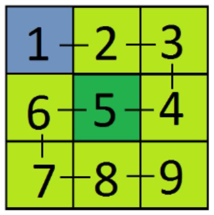


   - Using a 'smart' path determination to determine a more continuous path.



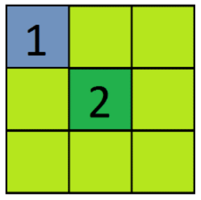
     In cases in which no optimal path can be determined, it could still fall back on its default routing behavior:





2. No impact on suggested route:
On the other hand, why bother predetermining the suggested route when the user has already chosen at the beginning to use an alternative route.

After completing step '9' in this example, it will continue with the first uncompleted step, which in this case is step '2'.



3. No suggested route when using alternative route

When the users chooses to deviate from the default path, another option is to disable the auto-suggested route altogether, and enter an "awaiting state" for the user to select *all* steps in the order he wants them to be executed.

Suggested option: 2. No impact on suggested route. This is the most familiar when compared to the Web Portal

Resulting design:

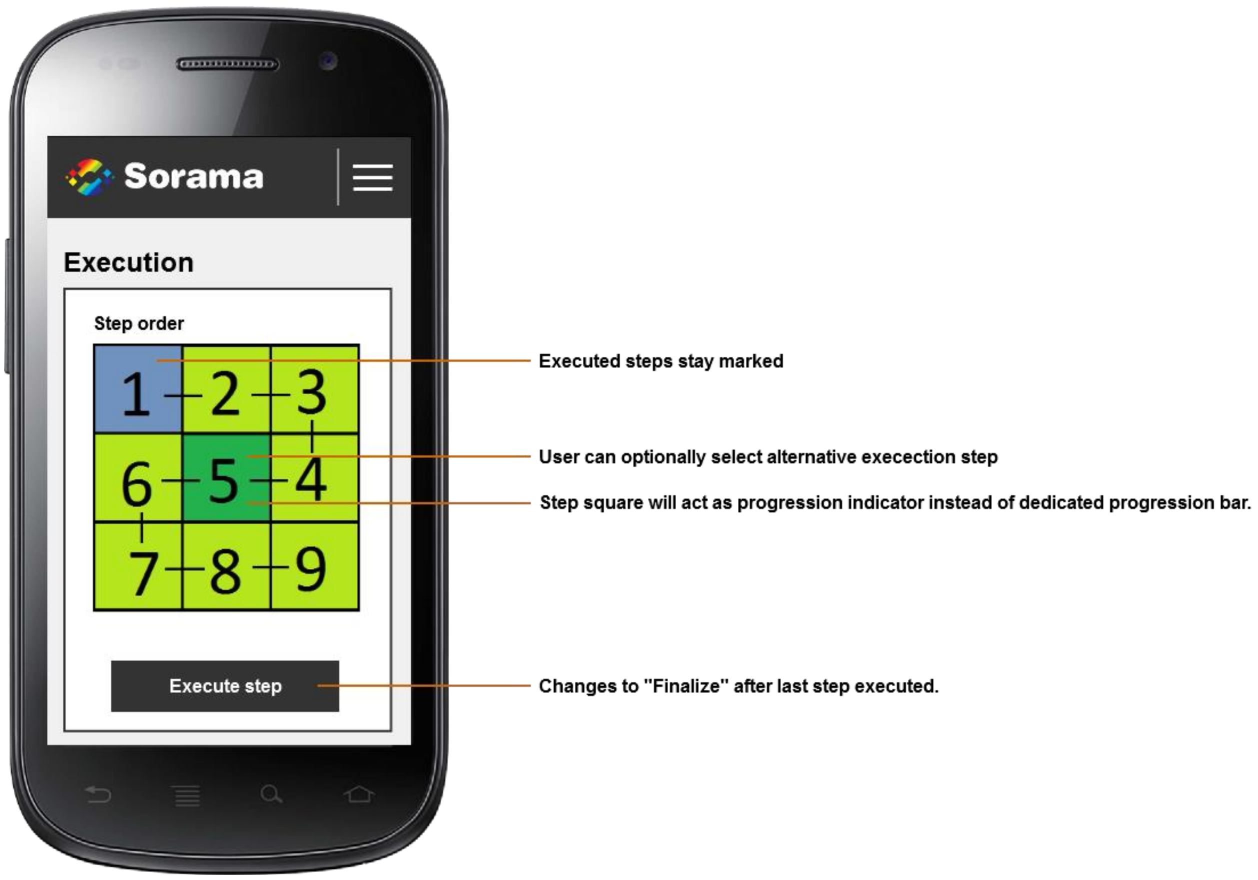

Sorama
Execution
Step order
1
2
3
6
5
4
7
8
9
Execute step
Executed steps stay marked
User can optionally select alternative execection step
Step square will act as progression indicator instead of dedicated progression bar.
Changes to "Finalize" after last step executed.

Image 8. HTML5 App measurement execution (simplified).

APPENDIX IV

Libraries, Frameworks and HTML5 features

TABLE OF CONTENTS

Libraries and Patterns 2
Knockout: MVVM Pattern 2
RequireJS: AMD pattern 2
JQuery: At the core of everything 3
SammyJS: Routing and Navigation 4
LESS: Simplifying and reducing CSS 4
Breeze: Entity State Management 4
Graphic libraries 5
Canvas Drawing Libraries 6
Data Plotting Libraries 8
HTML5 Features and Implementation Status 8
Introduction 8
Native touch events 8
Semantic Elements 8
App Cache and Web Storage 9
Improved HTML Forms and Advanced features 9
Server sent events (SSE) 9
Microphone and Camera input 9
WebGL 10
Motion (accelerometer & gyroscope) 10
GeoLocation 10
Web Sockets 11
Bluetooth 11
File API 11
Web Workers 11
References 12

LIBRARIES AND PATTERNS

This chapter lists and describes some libraries JavaScript patterns that were researched to some extent in during this project. Some of them are used either researched Durandal or Angular SPA frameworks, should be included specifically when used.

Knockout: MVVM Pattern

Knockout is one of the most popular libraries in modern web development. It transforms the traditionally static HTML into one that is dynamic, using declarative data binding. This allows associating DOM elements with model data in a way similar to the way that Silverlight maps XAML elements to (either or not) accessible properties in-code. It Durandal's core plugin for implementing the Model-View-ViewModel structure. Knockout handles automatic refreshes of the UI when properties are changed, and it also supports dependency tracking and templating. Due to its popularity there are also a number supplementary libraries that expand on its functionality.

Figure 1 and Figure 2 show an example of a View binding to an Observable Array rooted in a ViewModel as how this is implemented by Knockout. Note that the observable array is mapped to an empty array in this example. Updating this array at any time with projects, also results in an automatic UI update.

```
<section data-bind="foreach: projects">
    <div data-bind="text: Name" />
</section>
```

Figure 1. Simple Knockout data binding sample: The View mapping a section to an array of projects.

```
var projects = ko.observableArray([]);
```

Figure 2. The corresponding object in the ViewModel that it maps to.

RequireJS: AMD pattern

A broadly used design pattern in JavaScript development is the *Module Pattern*, which defines a way of encapsulating pieces of JavaScript code (Figure 3).

```
 (function () {
   myGlobal = function () {};
}());
```

Figure 3 Example of JavaScript module that encapsulates a function

This relies on using global variables to attach the exported values onto, and also limits the loading strategies since any dependencies are assumed to have been loaded already. RequireJS solves these problems using the *Asynchronous Module Definition (AMD) pattern* which allows registering a module and its dependencies to a factory. Upon accessing a module, RequireJS handles (asynchronous) dependency module loading and awaits them all to be loaded before executing the factory function.

An example of its usage is shown in Figure 4.

```
//Calling define with a name, dependency array and a factory function
define( ‘SimpleModule’, ['plugins/http', domain/services’],

     //Define the module value by returning a value.
    function (http, services) {

       /* Private functions and properties */
       function someMethod() { … }
       var someVar = … ;
       var anotherVar = …;

       /* Return object defines public functions and properties */
       return {
          SomeMethod: someMethod,
          SomeVar: someVar
       }
    }
);
```

Figure 4. Sample code of JavaScript AMD module as implemented by RequireJS.

This defines a module named ‘SimpleModule’ that has dependencies to ‘plugins/http’ and ‘domain/services’, which will be injected into the arguments ‘http’ and ‘services’. It exports a JavaScript object that exposes a private method and variable.

In Durandal the modules are defined using RequireJS, which allows addressing individual modules from different JavaScript files. It distinguishes modules that export an object (singleton), and modules that export a function (multiple instances possible).

JQuery: At the core of everything

Core library that introduces many features such as simplified DOM traversal and manipulation, event handling, animation, and simplified *Ajax*. Seems to be at the core of many other libraries, however thanks to other libraries such as Knockout and Durandal, the need to use JQuery itself directly is not as high as in legacy web development, which was known for jQuery dependent coding.

PROMISE DESIGN PATTERN

JQuery also implements a Promises design pattern, which is a pattern that allows chaining asynchronously executed methods. Traditional JavaScript leads deeply nested code when chaining callback handlers, a promise pattern allows reducing this and thus results in cleaner code. An asynchronous method that is chained like this is considered as a 'Deferred' object, meaning that it will fire callbacks or gain contents at a later stage (after the asynchronous method has been completed).

```
 Service.GetSomeData()
    .complete(dataRetrieved) // dataRetrieved called on success
    .fail(errorHandler)      // errorHandler called on fail
    .always(function(data){
          //code that executes both on error and on fail.
});
```

Figure 5. JQuery's implementation of Promise pattern

The example in Figure 5 illustrates an example of using JQuery's promises to chain functions. A deferred method GetSomeData is called on a service, and both a success and error handler are subscribed to its completion. There are also other statements such as 'always' that is called in every case, and 'then' to chain multiple deferred methods. A popular library that also implements a promise pattern that is similar but more feature rich, is 'Q'.

SammyJS: Routing and Navigation

Since Single Page Applications consist of a single page, traditional browser navigation cannot be used since this would navigate to another page. In order to introduce some kind of navigation that can also be handled by the dedicated browser navigation buttons (back, forward), one could use hashtag based routing (#-routing). Hashtags are not considered to be part of the URL, but can be used by the application. SammyJS uses this to edit the URL when navigating between views, and also uses it to navigate to the correct view upon accessing such a hashtag based URL directly.

LESS: Simplifying and reducing CSS

CSS is great for creating dynamic web site layouts that can be changed quickly. However traditionally it would require a lot of redundant code to set simple properties like a max width that should apply on multiple DOM classes. LESS improves on this by essentially lowing more dynamic definitions within CSS such as variables, operations and functions.

Breeze: Entity State Management

Breeze is a library that is dedicated into management of entities. It can be used to query WebAPI web services (or with much more work, other services), and it will automatically treat the retrieved models as entities. It implements a LINQ like query structure. Manually retrieving this data will only result into JSON objects (presented as simple JavaScript objects), whom do not have relations to child objects (but clones) and must be changetracked manually. Breeze handles all of this, and more. It uses Q for asynchronous data retrieval.

```
// define query on Customer entity.
var query = breeze.EntityQuery
              .from("Customers")
              .where("CompanyName", "startsWith", "A")
              .orderBy("CompanyName");

// execute it and attach success and fail handler to the promise.
manager.executeQuery(query).then(successHandler);
```

Figure 6. Querying with Breeze

```
// save all changes (if there are any)
if (manager.hasChanges()) {
    manager.saveChanges().then(saveSucceeded).fail(saveFailed);
}

// listen for any change to a customer
customer.entityAspect.propertyChanged.subscribe(somethingHappened);
```

Figure 7. Change tracking and subscription

GRAPHIC LIBRARIES

While HTML5 introduces more advanced graphic capabilities with the new <canvas> element and the improved <svg> element, using these directly requires relatively low level programming. There are a lot of JavaScript libraries out there that improve on using graphics and in this paragraph we will introduce and describe some of them.

When it comes to Canvas versus SVG based graphics, Canvas is in most cases the best option due to it having immediate mode drawing and as a result higher performance in scaling and manipulating objects. It allows more performance and scalability with more complex and interactive graphics.

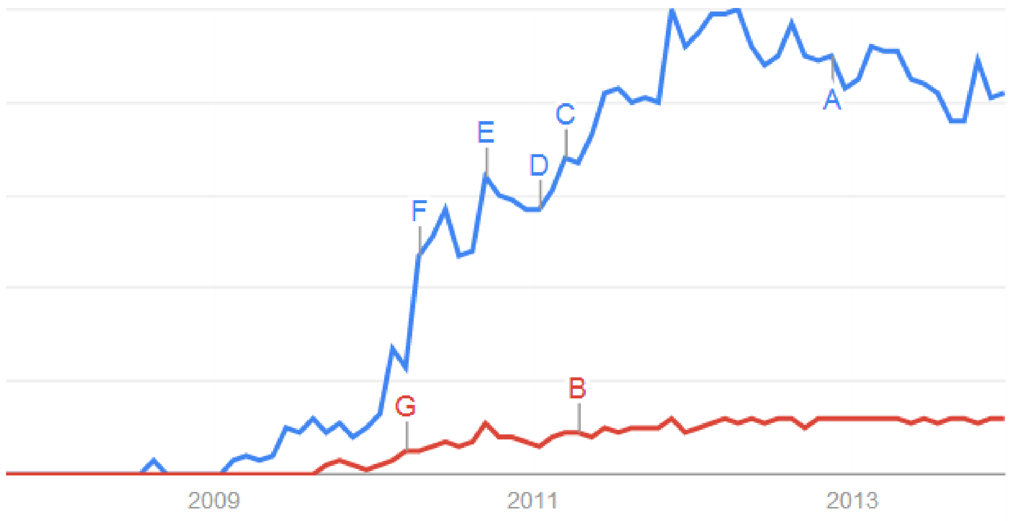


Figure 7. Interest in respectively Canvas and SVG **(Google Trends, 2013)**

Since HTML5 Canvas is a relatively young technology, the libraries that extend these are consequently also young. However there is quite a wide range of libraries already that benefit from active development and active communities. The most important question is: which of these libraries does fit our needs the best? Which should be answered by investigating the project's needs. For this project, it is fundamental to fulfill the needs for the Remote Control application (which will be elaborated on in *Chapter 6. HTML5 Remote Control Application)*, and find the library allows to fulfill its needs. For example the Web Portal embeds interactive graphs with large data sets, there are specific libraries that focus on these. For more advanced rendering opposed to relatively simple animated graphics there are also specific libraries.

Canvas Drawing Libraries

First we focus on general 2d Canvas libraries, of which there are also quite a lot out there. The most popular seem the following however.

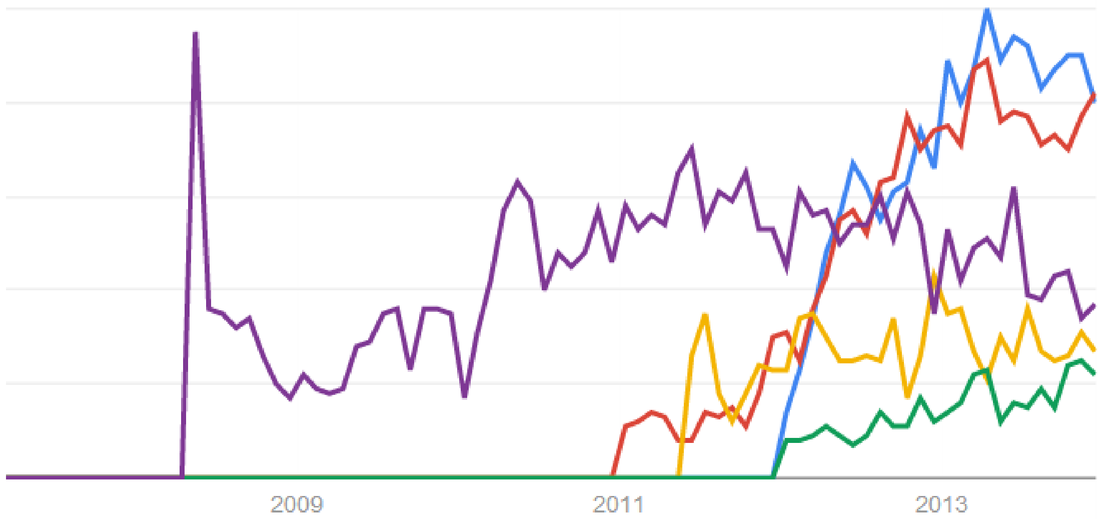


Figure 8. Processing.JS vs EaselJS vs KineticJS vs Fabric.js vs Paper.js **(Google Trends, 2013)**

ProcessingJS has been a long time popular library, but with the rise of HTML5's Canvas, its throne has been taken by EaselJS and KineticJS. Performance comparisons that can be found on generated on performance testers such as JSPerf.com, indicate that there are no major performance wise differences between these two, although EaselJS seems to perform slightly better in this case.

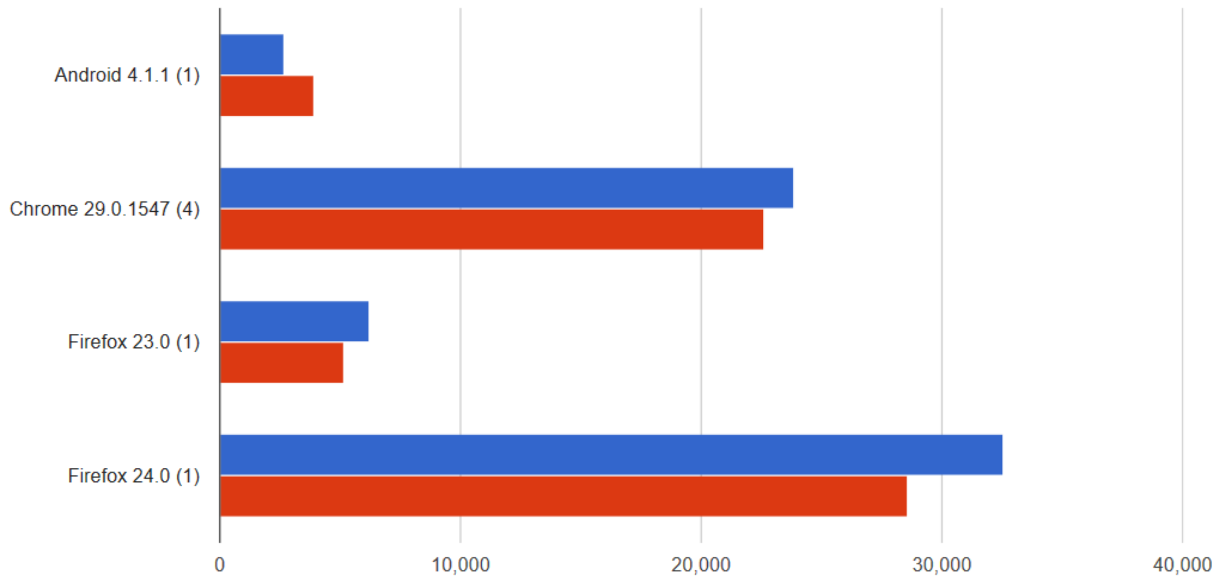


Figure 9. EaselJS versus KineticJS circle rendering performance (higher is better) **(JSPerf.com, sd)**

EaselJS offers some more advantages than KineticJS however. For one, it is part of a solution that also includes TweenJS, a library to easily animate objects integrated with EaselJS, which should be considered to be used when implementing animations within the graphics. Secondly, its architecture has been inspired by

ActionScript, which is a language with high focus on maintaining and manipulating objects in a graphic stage. Lastly, while offering both a solid documentation, EaselJS's examples and dedicated community forum seem

Data Plotting Libraries

There are a lot of data plotting libraries out there that offer all kinds of graphs. While this project had no real focus on these libraries since they were of no use to the Remote Control, during the orientation period the library DyGraphs was used to create a graph based on Sorama data. It could handle the large data relatively well and perform quick redraws on manipulation, but it was (with the used configuration) not as responsive as the graph used in the Web Portal. The library D3.js seems one of the most popular data plotting libraries. Though no investigation into this library has been performed, it can be noted that it offers a lot of possibilities and has a lot of community support.

HTML5 FEATURES AND IMPLEMENTATION STATUS

Introduction

Since the HTML5 specification is not final as of yet, the new features are not implemented yet by each browser manufacturer, and some implementations are based on manufacturers' assumptions since due to specifications being not comprehensive enough. This document will cover some of the (current) implementation statuses. Most of the data is retrieved from CanIUse.com, a website specialized in managing the implementation status between browser versions. For a more precise view on the implementation state of a feature I refer to this source.

Native touch events

Since HTML4 it has been possible for websites to trigger events natively. These events ranged from media events to input and window manipulation events, but there had been no inclusion of touch events. Mobile browsers of course (partially) overcome this issue by treating a touch event like a regular mouse or keyboard event, but if a website component requires specific touch gestures like pinch-to-zoom, these browsers offer no elegant solution. HTML5 solves this problem by adding native touch events as well. While these new native events are restricted to touchstart, touchend,touchmove, and touchcancel, these can be combined to support extensive gestures and multi-touch. There seem to be quite some libraries out there already that offer this out-of-the-box.

Semantic Elements

In addition to the more feature rich elements that have been added, HTML5 adds a large amount of new semantic elements**.** Nowadays, most (non HTML5) websites form their layout using a large amount of <div> elements, which form the website layout using CSS and rooting <div> elements inside each other. While this works, it is not very elegant. HTML5 introduces some tags to add a bit more layout managing and code readability (and thus maintainability). Among these, are elements such as <header>, <footer> and

<aside> to specify where parts of the website are nested. Another element is <time> to differentiate date/time elements (to allow, for example, more easily base these on ones preferred format). There are many more, a more extensive list can online.

App Cache and Web Storage

Another great feature added with HTML5 is App Cache support. This makes a web application, or parts of it, able to run offline, depending on which resources -such as pages, images and style sheets- the programmer has configured to be cached. The great thing about this, is that it can help offer a fluid app experience, even when the internet connection is not stable. If certain features are internet dependent, it is possible to configure to add fallback mechanisms, such as presenting the user that he should try again when the connection has returned. In a somewhat related topic, HTML5 also adds Web Storage, which is similar to, but much securer and faster than, Cookie Storage. It differentiates local and session storage.

Improved HTML Forms and Advanced features

While the aforementioned HTML5 features seem like the most interesting improvements in the new HTML specification, there is of course a lot more to it. Among other features are improved HTML Forms with additional tags that add useful things like auto-completion and validation on <form> and <input> elements. There will also be a range of more advanced features added, on which will elaborated in the next paragraph.

Server sent events (SSE)

HTML5 adds native support for server sent events. This was achievable in past specifications, but the browser / web application had to request if there were any updates. Now events can be caught automatically.

Microphone and Camera input

HTML5 has a specification that should allow web applications to request microphone and camera input streams. However, this is still a working draft specification, with changes still being made (at the time of writing (16-sept-2013) the most recent change was 03-sept-2013). As a result this features is available only in a select number of browsers and their implementations seem (at initial sight) not complete. (Partial) support exists in Firefox, Chrome, Blackberry Browser, Chrome for Android.

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	10.0	24.0	30.0	6.1	17.0	6.1		4.1	7.0	
Current	11.0	25.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		26.0	32.0		18.0			4.4		

WebGL

WebGL is a JavaScript API that allows hardware accelerated 2D and 3D graphics rendering. This might be an interesting technology to use. Currently not all browsers support this technology, but it seems that this will change with near future versions. Currently supported by Firefox, Chrome, Safari, Opera, Opera Mobile, Blackberry browser and Firefox for Android. In beta versions of the respective browsers, Internet Explorer 11 and Chrome for Android 30 also support this.

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	10.0	24.0	30.0	6.1		6.1		4.1	7.0	
Current	11.0	25.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		26.0	32.0		18.0			4.4		

Motion (accelerometer & gyroscope)

There is specification for device orientation. However current implementations are either lacking or partial.

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	10.0	24.0	30.0	6.1		6.1		4.1	7.0	
Current	11.0	25.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		26.0	32.0		18.0			4.4		

GeoLocation

Implemented in most current browsers, with the exception of Opera Mini. Internet Explorer supports it since version 9.0.

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	10.0	24.0	30.0	6.1		6.1		4.1	7.0	
Current	11.0	25.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		26.0	32.0		18.0			4.4		

Web Sockets

Native UTP socket support for web applications. This enables a lot of interesting scenarios. Supported by pretty much all current browsers, Internet Explorer supports it since version 10.0.

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	10.0	24.0	30.0	6.1		6.1		4.1	7.0	
Current	11.0	25.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		26.0	32.0		18.0			4.4		

Bluetooth

As of now, it appears that Bluetooth will not be included in the specification of HTML5. While an old specification does actually mention Bluetooth (W3C, 2008), this has been removed at a later stage (W3C, 2012).

File API

There is both specification for, and already broad implementation of the File API which allows file access and manipulation through a web browser. Some services like GMail and Outlook already exploit this API to allow drag-and-drop of files into a web application.

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	10.0	24.0	30.0	6.1		6.1		4.1	7.0	
Current	11.0	25.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		26.0	32.0		18.0			4.4		

Web Workers

One of the disadvantages of an HTML powered web application, is that JavaScript traditionally is single-threaded. This means that when the application requires to do some more intensive processing, that the application well become unresponsive until that processing has been completed. HTML5 introduces Web Workers, which is a type of multi-threading. It has however some limitations compared to C# threading. For example, they cannot share resources. A web worker can be given an input and may return an output, but it cannot access the UI for example. Communication towards and from the web worker can also only be done by passing JSON serialized objects. For some scenario's this may be an issue.

REFERENCES

Bootstrap. (n.d.). *Bootstrap*. Retrieved from GetBootstrap.com: http://getbootstrap.com/

Breeze. (n.d.). *Rich ddata for JavaScript apps is a Breeze*. Retrieved from Breeze: http://www.breezejs.com/

Can I Use. (n.d.). *Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers.* Retrieved from CanIUse.com: http://caniuse.com/

Google Trends. (2013, december). *Google Trends*. Retrieved from Google Trends: http://www.google.com/trends/

JSPerf.com. (n.d.). Retrieved from JSPerf.com: http://jsperf.com

Knockout. (n.d.). Retrieved from KnockoutJS.com: http://knockoutjs.com/

Osmani, A. (2012). *Learning JavaScript Design Patterns.* Retrieved from AddyOsmani.com: http://addyosmani.com/resources/essentialjsdesignpatterns/book/

The jQuery Foundation. (n.d.). *JQuery - Write less, do more.* Retrieved from JQuery: http://jquery.com/

W3C. (1999, December 24). *HTML 4.01 Specification*. Retrieved from W3.org: http://www.w3.org/TR/1999/REC-html401-19991224/

W3C. (2008, January 22). *A vocabulary and associated APIs for HTML and XHTML*. Retrieved from W3.org: http://www.w3.org/TR/2008/WD-html5-20080122

W3C. (2012, December 17). *HTML5 W3C Candidate Recommendation*. Retrieved from W3.org: http://www.w3.org/TR/2012/CR-html5-20121217/

W3Schools. (n.d.). Retrieved from W3Schools: http://www.w3schools.com/html/html5_intro.asp