Next Generation Graphical User Interface for IPTV

JOAKIM SVENSSON

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2013

Next Generation Graphical User Interface for IPTV

J O A K I M S V E N S S O N

2D1028, Master's Thesis in Media Technology (30 ECTS credits)
Degree Programme in Computer Science 270 credits
Royal Institute of Technology year 2013
Supervisor at CSC was Marko Turpeinen
Examiner was Alex Jonsson

TRITA-CSC-E 2013:002
ISRN-KTH/CSC/E--13/002--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.kth.se/csc

Abstract

It is common practice to use web technologies when creating graphical user interfaces for IPTV set-top boxes. The graphical user interface in TeliaSonera's IPTV service is built with HTML, CSS and Javascript. Lately focus has shifted from just showing video to also provide richer user experiences – coming from requirements of new services and the ongoing shift in resolution from standard definition to high definition. However, the set-top box is a device with very limited resources and it is important to assure good performance while allowing richer user interfaces.

Against that background the thesis asks "how can a web-based graphical user interface for IPTV set-top boxes, such as that of TeliaSonera, be improved and prepared for future IPTV services and richer user experiences?"

To provide an answer two studies were made; one of web technologies and one of future IPTV services. The technologies were evaluated in a Pugh matrix against a set of criteria (the most important being performance, platform independence, and resolution independence) as well as against the results of the IPTV services study.

SVG (Scalable Vector Graphics) scored best in the evaluation. The SVG renderer performed significantly better on the set-top box than the Mozilla web browser that TeliaSonera currently used. To put the technology to test the HTML based user interface was ported to SVG with expectations of a more responsive and graphically rich user interface. The result was satisfying. Loading times had been halved, the user interface works in both SD and HD resolution and richer graphics were added in form of gradients and a skinning system that allow easy change of appearance.

The thesis recommends SVG when building a rich graphical user interface for a set-top box.

Nästa generation grafiskt användargränssnitt för IPTV

Sammanfattning

Grafiska användargränssnitt för IPTV set-topboxar byggs vanligtvis med webbtekniker. Det grafiska användargränssnittet i TeliaSoneras IPTV-tjänst är byggt med HTML, CSS och Javascript. På senare tid har fokus gått från att enbart visa video till att även tillhandahålla en förbättrad användarupplevelse. Detta är en konsekvens av krav som kommer från nya tjänster och den pågående förändringen från standardupplösning till HD-upplösning. Dock har set-topboxen begränsade resurser och det är viktigt att kunna säkerställa bra prestanda samtidigt som användarupplevelsen förbättras.

Mot denna bakgrund ställer rapporten frågan ”hur kan ett webbaserat grafiskt användargränssnitt för IPTV set-topboxar, såsom det från TeliaSonera, förbättras och förberedas för framtida IPTV-tjänster och bättre användarupplevelser?”

För att svara på frågan har två undersökningar gjorts; en om webbtekniker och en om framtida IPTV-tjänster. Teknikerna utvärderades i en Pughs matris mot en mängd kriterier (de viktigaste är prestanda, plattforms- och upplösningsoberoende) samt mot resultatet från undersökning om IPTV-tjänster.

SVG (Scalable Vector Graphics) fick högst poäng i utvärderingen. SVG-renderaren presterade betydligt bättre på set-topboxen än webbläsaren som TeliaSonera använde för tillfället. För att testa tekniken ytterligare gjordes en portning av det HTML baserade användargränssnittet till SVG, med förväntningar på ett mer responsivt och grafiskt effektfullt användargränssnitt. Resultatet uppfyllde förväntningarna. Laddningstiden halverades, användargränssnittet fungerar i både SD- och HD-upplösning och grafiska effekter lades till i from av gradienter och ett ”skinning”-system som tillåter att enkelt ändra utseende på användargränssnittet.

Rapporten avslutas med att rekommendera SVG som teknik för att bygga grafiska användargränssnitt för set-topboxar.

Foreword

I would like to thank my supervisors at TeliaSonera, Per-Ola Wester and Annika Kilegran, for supporting me throughout the thesis. I would also like to thank Niels Bosma, at the time at Motorola, for being of great support when building the prototype.

A special thanks to my boss, Jessica Lindberg, for encouraging me to finalize the thesis even though I would be absent from work for a while.

Finally, I would like to give a big thanks to my family; Nadia, Lennart, Gunnar, Marcella and Sofia for reminding, supporting and encouraging me to finalize this report.

Abbreviations

Table over abbreviations	
IPTV	**I**nternet **P**rotocol **T**elevision
STB	**S**et-**t**op **b**ox
GUI	**G**raphical **U**ser **I**nterface
UI	**U**ser **I**nterface
SD	**S**tandard **D**efinition
HD	**H**igh **D**efinition
OS	**O**perating **S**ystem
AJAX	**A**synchronous **J**avaScript **A**nd **X**ML
W3C	**W**orld **W**ide **W**eb **C**onsortium
DOM	**D**ocument **O**bject **M**odel
DLNA	**D**igital **L**iving **N**etwork **A**lliance
UPnP	**U**niversal **P**lug and **P**lay
DRM	**D**igital **R**ights **M**anagement
MPEG	**M**oving **P**ictures **E**xperts **G**roup
CPU	**C**omputational **P**rocessing **U**nit
DSP	**D**igital **S**ignal **P**rocessor
EPG	**E**lectronic **P**rogram **G**uide
VoIP	**V**oice **o**ver **IP**
VOD	**V**ideo **O**n **D**emand
CE	**C**onsumer **E**lectronics
PVR	**P**ersonal **V**ideo **R**ecorder
ITF	**I**PTV **T**erminal **F**unction

Tables

Figures

Contents

Chapter 1
Introduction

TeliaSonera is the leading IPTV operator in Sweden. The graphical user interface (GUI) used in their set-top boxes (STBs) is built with web technologies; HTML, CSS and Javascript.

The company now wants to improve their STB GUI to become faster, withstand the transition from standard definition (SD) graphics to high definition (HD) graphics and also be able to manage new IPTV services as they may arrive in the near to mid future.

1.1 Problem description

Common practice today when creating GUIs for IPTV STBs is to use web technologies; they have been sufficient for providing the functionality required the user interface, they are easy to learn, widely adopted and has a large developer community. The user interface can run in a web browser, which makes it easy to move across different platforms.

Lately it has become important to provide a rich user experience apart from showing video, which is main function of the STB. The STB has dedicated hardware to handle video, but the overall performance is limited. In fact, the STB is a small computer with very limited resources. To provide a richer user interface the performance of the STB and the technology that runs on it is a crucial factor.

The research question for this thesis is defined as follows:

> *How can a web-based GUI for IPTV STBs, such as that of TeliaSonera, be improved and prepared for future IPTV services and richer user experiences?*

A summary of future IPTV services is derived from interviews with leading players in the IPTV STB and service area. A study of available web technologies is then

performed with focus on criteria developed in collaboration with IPTV experts at TeliaSonera.

The web technologies are evaluated with respect to the criteria and the new requirement from future IPTV services, and finally a prototype is constructed with the best suited technology.

1.2 Objective

The purpose of the thesis is to present a technology that is well suited for a modern browser-base STB GUI and apply it on a real-world example. The thesis will both give theoretical understanding of what is expected of such a technology and put it into practice to verify the result in a real-world complex system.

1.3 Criteria and scope

Five criteria have been developed together with experts in the IPTV group at TeliaSonera. The criteria form the focus area of the technology study and are used when evaluating the technologies.

Performance The ability to have a fast user interface with minimum response time to user interaction.

Platform independence The user interface should be consistent across multiple terminals with different hardware capabilities.

Resolution independence A better user experience when moving from SD to HD resolution.

Rich user experience The ability to easily customize the appearance of the user interface (skinning), the ability to provide richer graphical effects, such as animations.

Development community A user interface built with tools and technologies that have a large development community facilitates development and maintenance.

The technology research is limited to web based technologies, viewable in a browser, and because of limitations in the lab environment only the technologies that are able to run on the available equipment will be tested.

1.4 Methodology

The methodology used throughout the thesis is a mixture of focused interviews, literature studies, decision making and software implementation.

Chapter 2 Set-top box and IPTV services study is mostly based on focused interviews. That is, the questions are not made up on beforehand; instead the interview focuses on some specific areas that are of interest and lets the interviewee speak more of less freely about these [1]. In this case the areas are STB history and future IPTV services. The interviews are made with three IPTV STB manufactures – ADB, Amino and Motorola – and an IPTV service developer company - Accedo Broadband. Besides interviews whitepapers from the Open IPTV Forum are studied.

Chapter 3 Technology study is based on a literature study over available web-technologies. The literature mostly consists of specifications and papers collected from the different technologies' web sites. To gain knowledge about what technologies exist, online dictionaries, news groups and forums have been used. The criteria from 1.3 Criteria and scope are defined more thoroughly and serve as a basis for what material is presented.

The results from Chapter 2 are analyzed regarding what impact they will have on a STB GUI. Each technology from Chapter 3 is then evaluated with respect to:

- Whether the technology supports the IPTV services.
- Whether the technology fulfills the criteria defined earlier.

The standard format for evaluation in decision theory is called a decision matrix [2]. It evaluates and prioritizes a list of options against a set of weighted criteria. The process of a decision-matrix model may vary depending on sources, but usually the same base structure is used [2] [3] [4].

1. Identify the decision to me made.
2. Identify criteria that are of importance for the decision.
3. Assign individual weights to each criterion.
4. Identify the possible alternatives to be compared.
5. Score the alternatives for each criterion using a decision matrix or another evaluation tool.
6. Choose the alternative with the highest score.

The decision matrix is an L-shaped matrix which tabulates the alternatives to the criteria. It then scores each alternative relative to each criterion according to a predefined scoring system. The Pugh matrix is a special form of a decision matrix that uses one of the alternatives as reference [2]. Every other alternative is then scored relatively the reference using a scale of "worse, same or better" (or in more steps) [2].

The Pugh matrix is fitting in this case because a reference alternative already exists: the current implementation of the GUI.

The results from the evaluation are then applied in Chapter 5 Case TeliaSonera. A prototype based on the current GUI is implemented. The implementation process consists of three steps:

1. Define requirements on the implementation derived from the criteria.
2. Analyze the current GUI to identify what parts (if any) can be reused (the lowest common denominator of the new and old technology).
3. Implement.

Finally the result of the prototype implementation is presented with focus on the requirements.

1.4.1 Qualitative and quantitative methods

Mostly, this thesis is based on quantitative data. The results rely on measurable data, such as capabilities and characteristics of the various technologies. But the most important decisions come from qualitative data that is derived from interviews and discussions. These decisions determine how the quantitative data is used and what data is most important. An example is the technology evaluation. The evaluation is made using quantitative data, but the different criteria and their weights that control the outcome are selected based on qualitative data.

1.5 Related work

In the doctorial thesis "A Graphics Software Architecture for High-End Interactive TV Terminals" [5] Pablo Cesar suggests an architecture for DVB TV terminals based on the Java GEM[1] standard. Though the graphics layers are based on Java he suggests that the client should be able to render XML-based content. However, XML content is only suggested for simple (non interactive) applications and the graphics profiles use XForms in favor of a scripting language.

In the master thesis "STB application development based on modern web technology" [6] Niels Bosma investigates how modern web technologies can inspire development methods of set-top box portals. He states that the browser environment is preferred due to its flexibility, but has drawbacks, such as large memory footprint and low performance. After concluding that STB-portals could benefit of modern web technologies, he creates an application framework to facilitate building graphical STB-portals.

[1] http://www.mhp.org/introduction.htm

The thesis "A feasibility study of building Set-top box user interfaces using Scalable Vector Graphics" [7] by Fredrik Vinkvist concludes that SVG is suited to build a STB GUI. Vinkvist creates an application framework that facilitates development and addresses SVG's shortcomings, such as lack of other layout modes than absolute positioning. However, he never had the opportunity to run the technology on an STB, leaving performance tests undone.

1.6 Equipment

The hardware used is a Motorola VIP 1910 STB. The STB runs an adapted version of the Mozilla 1.7 web browser and the Ekioh SVG renderer[2].

1.7 Set-top box architecture

The architecture of Motorola 1910 STB is described in the diagram below. The parts that are of especial interest in this thesis are the two top most layers. First the web page, where the GUI application resides, and second the web browser in which the GUI application is executed after the web page has been loaded by the web browser. The web page can control STB functionality through a Javascript API exposed by the web browser or plugins in the web browser. The browser and plugins in turn communicate with the STB middleware using C/C++ APIs.

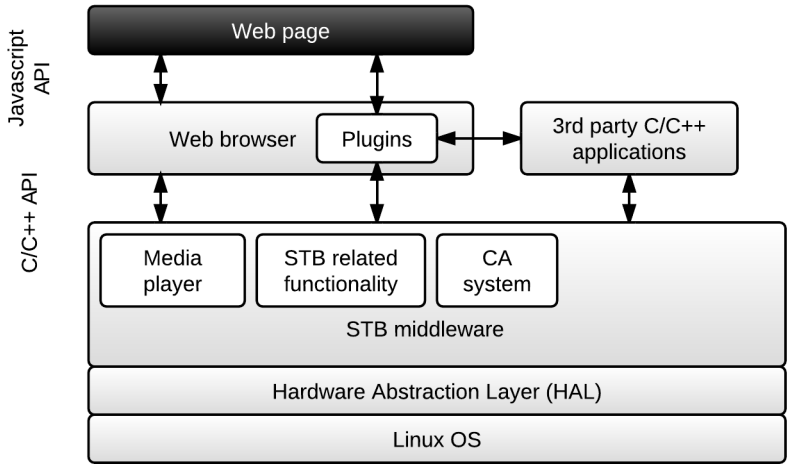

Figure 1-1 Overview of a Motorola STB architecture

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

1.8 Note on performance

It is hard to predict performance of a technology because it depends on many factors. If an application is heavy on computations, performance of the code execution environment is of most importance. If an application is heavy on graphics, performance of the rendering engine is of most importance. Tests can be created to measure different types of performance, such as the Sunspider [8] Javascript benchmark that tests pure Javascript performance, or the GUIMark [9] benchmarks that test the rendering engines of a few technologies.

Results from GUIMark show that both Flash and HTML produce vastly different results on Windows, OSX and Linux, all running on the same device [10]. To make these types of tests meaningful, it is important to have them run on the target device, with the software available on that platform. When performance tests are made in this thesis, only the technologies that currently can run on the STB are tested. This may leave a hole in the comparison, but I find it better than to draw conclusions from unrelated results.

1.9 Terminology and definitions

1.9.1 Graphical user interfaces

In 2004 the Linux Information Project[3] defines a graphical user interface (GUI) as "... a human-computer interface (i.e., a way for humans to interact with computers) that uses windows, icons and menus and which can be manipulated by a mouse (and often to a limited extent by a keyboard as well)." [11]

In the STB case the above definition still holds true with the addition that a remote control is used instead of a mouse.

1.9.2 Web technology

This thesis divides web technologies into two groups; native and plugin based. Server-side technologies are not considered here.

A technology that can run in a web browser is here considered a native web technology. Examples are presentational technologies such as HTML and CSS and the scripting language Javascript.

[3] http://www.linfo.org/

A technology that can run in a web browser by appending a plugin to a web page is here considered a plugin based web technology. Java applets and Adobe Flash are examples.

The main difference between these two is the need for additional software that a plugin based technology needs. There are also security aspects to consider – in this case, where the user interface engine is a web browser, a native web technology is sandboxed to the browser environment while a plugin may have access to other parts of the system.

1.9.3 Browser environment

The browser environment refers to the execution context of a web browser, where the web technology is executed.

Compared to a native environment, which here is considered the operating systems application layer, a technology executed in the browser environment can only access features that the browser exposes, hence not interfere with low level functionality.

Chapter 2
Set-top box and IPTV services study

This chapter seeks an understanding on how IPTV user interfaces and the set-top box hardware have evolved so far and how they will evolve in the next couple of years. The goal is to describe the present situation and provide suggestions of future IPTV services that are relevant for TeliaSonera. The study is mostly based on interviews, which have focused on two areas: set-top box performance and IPTV services.

A historical view over the IPTV set-top box's evolution was created, with CPU performance as main topic. This includes past and present models, as well as the next one or two generation's models. The information will give an understanding on what is possible with coming set-top boxes and can be used to determine how graphical intense the services can be.

Regarding IPTV services, the intension is to understand where the business is heading, to understand what kind of services that are coming and see what impact these will have on the user interface.

The set-top box manufacturers that have been interviewed are Motorola, Tilgin and ADB, all with quite extensive experience in the IPTV industry. An additional interview was performed with Accedo Broadband, a company that provides interactive applications and on-demand content to IPTV platforms.

2.1 Interview with Motorola

At Motorola, Hans Vind was interviewed [12]. He works as senior product specialist and has long experience in the consumer electronics business. He has also previously worked with human-computer interaction.

2.1.1 Set-top box history

Hans Vind explains that the industry's focus from 2005 to 2008 has been on the transition from MPEG-2 to MPEG-4. MPEG-4 enables HD content on the set-top box but requires about four times the data set compared to SD content. Work on encryption has been prioritized and time has been spent on finding approaches to keep channel switch times down. Today, a resolution of 1920x1080i is achieved at 60 fps. The goal is 1920x1080p, which is the Blu-ray standard.

The main CPU had a clock frequency around 300 MHz during the last three to four years. It is first now, when the MPEG-4 problems are basically solved, that focus has moved to increase the performance of the set-top box, Hans Vind continues. The current model, the Motorola IP-STB 1900-9, has a CPU of 266 MHz, two DSPs for video decoding of 400 MHz and 128 MB of RAM of which 32 MB is used for video memory. The next model, to be shipped during 2009 has an improved CPU, faster DSPs and double RAM. Performance is improved even more in future models, with two CPUs and native support for OpenGL and OpenVG. Table 2-1 shows the set-top box history at Motorola.

Hans Vind does not believe that the set-top box will cease to exist. He admits that it probably will be hidden away somewhere but as long as there are large TV operators and content providers that promote themselves by delivering TV content, the set-top box will always prevail. The reason is that otherwise, if the set-top box would be integrated in the television, the end customer could only choose from a limited set of television sets since not all television manufacturers would and could provide support for all TV operators' products. A possibility would be a common interface where the set-top box can be plugged in. Motorola has a working prototype of an IP-STB 1900-9 set-top box that is of the size of a PCMA card. The problem at the moment is that there is no such standard. The Common interface slot that exists on modern TVs does not support video transmission. So for the next five to six years we will have set-top boxes as we know them today. Then we will see, Hans Vind says.

2.1.2 Future IPTV services

Motorola has support for four environments for user interfaces in their set-top boxes. The first approach is using a web browser with HTML and JavaScript to present the user interface. The second is using an SVG renderer with SVG and JavaScript. The third is Adobe Flash, which can be implemented as a standalone user interface renderer or as a web browser plug-in. The last approach is Espial EVO, a proprietary solution which compiles HTML and JavaScript.

Preferably, Motorola likes to support standard technologies, i.e. the browser environment with HTML and JavaScript, SVG and JavaScript or native solutions that operates directly towards the set-top box. Should there be demands from customers

about proprietary technologies Motorola will look into it, but they recommend standardized technologies.

The big advantage of the browser environment, according to Hans Vind, is that it uses the same technologies as web services on the Internet, which makes it easy to integrate them with the set-top box. Hans Vind believes that the browser environment is the future just because it allows Web 2.0 services into the set-top box with minimal effort.

He states that the greatest desire at the moment from Motorola's customers is 3D graphics, and efforts are made to include support for that in upcoming set-top boxes. A problem is, though, that the scope of use for 3D graphics is still unclear and the set-top box has far from the computational power that is needed to model a full 3D user interface.

2.2 Interview with Amino (Tilgin)

At Amino Joakim Larsson was interviewed. Joakim is a senior systems architect at Amino (previously at Tilgin) and has 15 years experience in the set-top box industry.

2.2.1 Set-top box history

Tilgin launched its first IPTV set-top box in the year 2000. It was based on x86 architecture with a CPU at 266 MHz. In 2003 they launched its successor Mood 300, now based on a MPIS processor at 324 MHz. For the Mood 300 they implemented graphics acceleration support in a DSP for the purpose of allowing more advanced user interface graphics. But as the industry started to demand support for HD content, which the Mood 300 did not support, it got abandoned for a HD compatible box. In 2005 Mood 400 was released, which is the set-top box that is still used in 2008. It is based on a ST Mirco chipset with a 350 MHz processor and two DSPs of 600 MHz.

The average lifetime for a set-top box, according to Joakim Larsson, is three to four years. During that time the set-top box companies are dependent of that box and cannot afford to release new products every time a new chipset or new demands emerges.

Joakim Larsson says that Tilgin has held discussions with chipset manufacturers regarding increasing performance in the CPUs for a long time. It is just recently that efforts to do so have been made from the chipset manufacturers. The next generation of Tilgin set-top boxes (now labeled Amino), which is set to be release during 2009, will include a slightly better chipset.

2.2.2 Future IPTV services

Tilgin has decided to hold back on investing in new display technologies until the industry seems to have settled for one or a couple of approaches. They have contracted Opera to use their browser as user interface renderer in the meantime.

The next generation set-top box portal, according to Joakim Larsson, will be a widget based system. The operators will provide a portal application with an implemented widget engine. They will also provide standard widgets for services such as EPGs, VoIP and TV content. Third-party providers will then be allowed develop their own widgets for the portal – may it be games, online stores or Web 2.0 services – which will be made available to the users by the operator.

There will probably be a quantity of widget engines based on various technologies, but the overall system will be standardized. Widgets created for a specific engine may be used on all portal systems that implement that widget engine. Joakim Larsson says that this way the set-top box will become a portal to many different services available through the Internet. The system will probably resemble Windows Vista's gadget bar or Mac OS X's Dashboard, and will therefore be familiar to a large part of the end users.

Joakim Larsson presumes that it probably will be a long transition time from a browser environment to a widget-based environment. It is also likely that there will be a widget engine war, and until the winner is known not much can be said about system requirements.

Regarding 3D features, Joakim Larsson just as Hans Vind realizes that today's set-top boxes have insufficient performance to model such a user interface.

2.3 Interview with ADB

At ADB Marcin Pakula was interviewed [13]. Marcin Pakula works as Director of technical marketing at ADB.

2.3.1 Set-top box history

The latest ADB set-top box (as of 2008) is based on the ST Micro 7109 chipset. ADB's approach of building set-top boxes differs somewhat from those of Tilgin and Motorola. According to Marcin Pakula, ADB replaces the microcode from the DSP that handles the audio decoding with code for 3D graphics acceleration to improve performance. To do so they tend to wait a bit longer before embracing a new chipset since they believe that they can get better performance out of it than their competitors.

Marcin believes that the set-top box will get a more prominent role in the future. It will become the central point of the living room, which will provide a universal user

interface, through which it would be possible to connect to and control UPnP or DLNA enabled devices in the home.

2.3.2 Future IPTV services

ADB uses the browser environment as a platform for their user interfaces. Marcin Pakula is of the opinion that a web browser that renders HTML and JavaScript is sufficient for creating a set-top box user interface. At the moment they use Mozilla Firefox. On the other hand, he notes that there is a problem with the browser solution because there are no standardized APIs; they depend on both the set-top box vendors and the browser vendors. Marcin Pakula realizes that there is a need for a standard here and feels that the Open IPTV Forum's initiative to create such a standard is the right way to go.

He sees two major functions for the set-top box in the near future: home networking and Internet content. In the first case the set-top box could act as the central point in the living room. Through it one could access all UPnP and DLNA enabled devices connected to the home network, such as DVD players, cameras, computers etc. He claims that the technology exists today, but what is lacking is proper user interfaces. Because the number of devices connected to the home network may be very large, it is likely to become a problem for the user to find the media he is looking for. Most of the devices have their own internal catalogue structure and having to browse through all devices when looking for the media will be cumbersome and very time consuming. He says that there is need for an application that solves this problem and feels that this is an area that has been given too little attention.

Regarding viewing Internet content in the set-top box, he claims that we do not want to use the set-top box for browsing the Internet in the way we are used to in the PC environment since there is no keyboard or mouse available. What we do want, however, is having the set-top box presenting the content of the Internet to us in a way adapted for the big screen and the remote control. ADB is looking at different possibilities to bring Web 2.0 services like Facebook and YouTube into the set-top box.

Marcin Pakula then explains that in many countries there are a lot of information generated by communities about local traffic, weather and such. To be able to present this information through the TV, using a simple user interface, is the future of television. When the user has access to these communities, sharing of content is the next step and the users would then want to upload data to the communities. The technology exists today, he says, but has not been addressed properly by middleware providers or operators.

2.4 Interview with Accedo Broadband

Ted Björling was interviewed at Accedo Broadband. He is Manager of technology and platforms and has experience since the start-up of the company in 2004.

Accedo Broadband is mainly focusing on games for IPTV. Their product portfolio ranges from simple soduku games to more advanced multiplayer games. Most of their applications are based on HTML and JavaScript, but they have some Flash applications as well. Other technologies they are looking into are SVG, Espial EVO and Microsoft Mediaroom. Accedo Broadband investigates new technologies by customer demands and currently, according to Ted Björling, the biggest hype is around SVG.

He believes that community created applications and services are to come to the TV and gives Apple's AppStore as an inspirational model. An example could be an application that fetches information from Wikipedia about the current TV-show and presents the information to the user. Where this application will reside is yet to be determined, Ted Björling says. It could be the set-top box vendor that provides it, middleware vendor or a third-party vendor.

Accedo has done some research about Yahoo's Connected TV and entered a partnership with Yahoo. The idea behind this type of concept is that anyone should be able to create an application for the TV. The television experience will be more interactive if e.g. web services and games get connected to the TV. This is not a unique concept and most CE-vendors are working on similar systems. He thinks, though, that this concept takes a step away from the traditional TV operators since it provides many alternative ways for the user to get TV content. But the end users' expectations on content and services will increase as they get more accustomed to interactive TV services and these expectations must be met in order to keep the customers.

Accedo Broadband's vision is to follow the trends in the business and provide cutting edge solutions. Their main focus areas will be multiplayer games and Web 2.0 services. They have already, according to Ted Björling, made reference implementations that integrate against some Web 2.0 services. He believes that the time to market for these types of IPTV services is between one to three years. It is happening now, he says.

2.5 Open IPTV Forum services

The Open IPTV Forum has the purpose to produce end to end specifications for IPTV. The members come from all business areas associated with IPTV – network operators, content providers, service providers, CE manufacturers and network infrastructure providers [14]. Among these is TeliaSonera.

The Open IPTV Forum has decided upon using a special purpose web browser called CEA-2014 [15], developed by the Consumer Electronics Association. It uses CE-HTML, an extension to XHTML, and ECMAScript to render the user interface [16] and it has standardized APIs for communication with the IPTV devices [16].

The organization has produced a document that summarizes selected services and functions for its second release [17]. Some of those services are of interest for this chapter.

Among the services is PVR functionality with local or network based storage. The Open IPTV Forum suggests that the user should be able to remotely manage content and scheduling using a mobile device. A content guide should present the user with a list tailored to user preferences of scheduled, on demand and recorded content. A personalized channel service is also suggested, which would let the user create his own program line up based on his preferences, habits or the service provider's recommendations.

The Open IPTV Forum also suggests integration of IPTV with communication services such as voice and video telephony and chat with presence state indication. It is also suggested that it should be possible to deliver different components of the media stream to different devices. For instance could the audio stream be delivered to the mobile phone and the video stream to the television screen.

Moreover, it is suggested that interactive applications shall be allowed. These are defined as "applications that allow user interaction via the ITF[4] device or other user devices". This includes applications that are network-based and interact with the IPTV terminal device using web technologies, but also applications that reside in the home network. Mobile devices shall also be able to interact with an application connected to an IPTV service. An example could be to remotely schedule and manage the PVR with a mobile device, as suggested when the PVR functionality was described above.

Home networking is another service that is specified. It states that the IPTV user shall be able to access content stored on DLNA devices connected to the home network and also that IPTV content shall be offered to these devices. The user shall also be able to share content with other users, provided that the DRM policy allows it.

[4] IPTV Terminal Function

2.6 Summary

2.6.1 Set-top box survey

All interviews stated that the set-top boxes have not evolved so much considering CPU capacity during the IPTV-area. All focus has been on transitioning into the MPEG-4 standard that mainly concerns video decoding and is not handled by the CPU. According to all of the interviewees, a change in the industry is happening now. Focus is moved over to CPU performance and in the next models the CPU clock frequencies and the main memory are about doubled. The major set-top box chipset manufacturers are announcing new chipsets which confirms the change [18] [19].

The main chipset manufacturers used for IPTV set-top boxes are ST Microelectronics, Sigma Designs and Broadcom. Recently (late 2008) Intel decided to enter the CE device market by introducing a new chipset, based on the Intel x86 architecture [20].

The next generation chipsets from the some of the biggest players in the industry are listed in Table 2-3. Table 2-1 and Table 2-2 summarize the set-top box history at Motorola and Tilgin.

Table over Motorola set-top boxes

Model	*Chipset*
IP-STB 400	National Semiconductor G1 (CPU) Sigma Design 7400 (video decoder)
IP-STB 500	National Semiconductor Geode (CPU) Sigma Design 8401 (video decoder)
IP-STB 700	National Semiconductor Geode (CPU) Sigma Design 8405 (video decoder)
IP-STB 1500	ATI Xilleon 210D (comb. unit)
IP-STB 1900-9	ST Micro 7109 (comb. unit) Main CPU: 266 Mhz
IP-STB 19x3 (2009)	ST Micro 7105 (comb. unit) Main CPU: 450 Mhz

Next model (2010-2011)	Main CPUs: ~600 MHz x 2 Built-in support for OpenGL and OpenVG.

Table 2-1 Specifications for the Motorola set-top boxes

Table over Tilgin set-top boxes	
Model	*Chipset*
Mood 200	x86 266 MHz
Mood 300	MIPS 324 MHz
Mood 100	STMicro 200 MHz
Mood 400	STMicro 350 MHz
Next model (2009)	Improved CPU ~450 MHz

Table 2-2 Specifications for the Tilgin set-top boxes

Table over next generation chipsets		
Manufacturer	*Chipset*	*CPU*
STMicro	STi7105	ST40 450 MHz
Broadcom	BCM7405	MIPS 400 MHz
SigmaDesign	SMP8640	MIPS 333 MHz
Intel	CE 3100	Pentium M 800+ MHz

Table 2-3 Next generation's chipsets form some of the major CE chipset manufacturers.

2.6.2 IPTV services

Five major types of services have been identified based on the interviews and the Open IPTV Forum.

Internet content In common for all the interviews is the belief that Web 2.0 services will play a big part in the future of IPTV. The Open IPTV Forum is supporting this approach in their suggestion about interactive applications. It also suggests that users should be able to communicate with each other through video conferencing and chat.

Third-party applications Accedo Broadband suggests community created applications in the likes of the Apple AppStore concept. Tilgin believes that third-party applications will be big when the set-top box portal becomes widget based.

Widget system Both Tilgin and Accedo Broadband suggest a widget system. The widget system is an application environment that will open up the set-top box to small standardized applications, so called widgets. The purpose is that these widgets can be created by anyone with access to the API for that system and that the widgets run in a plug-n-play manner when deployed in the system. This type of system does not only target the set-top box, but all types of consumer electronics that are connected to a TV. Both companies mention that Intel and Yahoo have come quite far on developing such a system.

Home networking The Open IPTV Forum wants to allow the user to access content of other DLNA devices in the home network and vice versa. ADB also suggests this approach.

Mobile devices The Open IPTV Forum suggests that services should be accessible from a mobile device. It also suggests that it should be possible to deliver different parts of the media stream to different devices. These two facts give the opportunity to extend the IPTV experience from one screen into many.

2.6.3 User interface technology

All three set-top box companies have chosen a browser based approach with HTML and JavaScript as user interface technology. Motorola claims that the browser environment is the best way to go since it uses the same technology as the Internet services and will allow for fast integration. ADB and Tilgin are of the same opinion and have both chosen web browsers user interface platforms. Motorola recommends SVG or HTML meanwhile ADB and Tilgin sticks with HTML for now. The Open IPTV Forum has chosen a similar approach with its CEA-2014 browser.

There is, according to Motorola, a desire for 3D graphics. Motorola is to support OpenGL in its future set-top boxes and ADB has implemented some 3D acceleration in its set-top boxes. Both companies seem to realize that there is some confusion about what the 3D graphics purpose is.

Chapter 3
Technology study

This chapter aims to document what web technologies could be of interest for a browser based set-top box GUI. It uses the criteria from section 1.3 Criteria and scope as a base when describing the technologies.

3.1 Criteria

Each criterion is ranked by importance on a scale of 1 to 5. The rank has been set after discussions with the IPTV group at TeliaSonera. The criteria are defined more thoroughly and a set of focus questions has been developed for each criterion. See Table 3-1 for definitions and questions.

Performance The performance of a web technology depends on both the hardware of the device and the runtime environment in which the technology is run. *Weight: 5.*

Platform independence For the same technology to work seamlessly on different devices, it needs to be able to run on different operating system without having to consider the devices specific hardware configurations. *Weight: 4.*

Resolution independence The user interface should render with good quality independent of the screen resolution. *Weight: 3.*

Rich user experience Clear separation between graphics and code is needed. It should be possible to create or change the graphical components of the user interface without altering the code and vice versa. The technology should support rich graphical effect, such as animations and transitions. W*eight: 3.*

Development support Resources in forms of tools and development communities available for the technology. *Weight: 2.*

Table of criteria for the technology study	
PERFORMANCE, WEIGHT 5	
Script performance	How well does the technology perform on the target platform?
Graphics rendering	How well/fast does the technology render on the target platform?
PLATFORM INDEPENDENCE, WEIGHT 4	
Runtime environment	What is the run time environment for the technology? Does the runtime exists for different browsers?
OS support	What OS's does the technology support?
RESOLUTION INDEPENDENCE, WEIGHT 3	
Graphics format	Does the technology support vector graphics?
RICH USER EXPERIENCE, WEIGHT 3	
Skinnability	How does the technology handle separation of graphics and program code to allow easy skinning of the user interface?
Graphical effects	Does the technology support animation and other graphical effects?
DEVELOPMENT SUPPORT, WEIGHT 2	
Development community	Does the technology have a big development community? Is it easy to find information and support?
Tools	Do good tools and frameworks exist?

Table 3-1 Definitions of the criteria used for the technology investigations.

3.2 Web technologies

3.2.1 HTML

HTML version 4.01 is the current W3C recommendation and has been so since 1999. [21](chapter 2.3). The vision when developing HTML has been that all kind of devices should be able to access information on the web, and that HTML documents should be platform and device independent [21](chapter 2.2.1).

HTML has support for text, bitmap images and various layout elements [21](chapter 14.1). HTML is close coupled with other W3C specifications; an HTML

document can be manipulated with Javascript by modifying the DOM [22] and the appearance of HTML elements can be change by including CSS style sheets [23].

There are numerous Javascript libraries that provide utility functionality for DOM manipulation, animation, cross-browser issues and more [24].

HTML5

HTML5 is the latest draft of HTML and extends and refines the language by adding some elements, removing and altering other. Among the most notably are the audio and video elements – allowing multimedia content – and the canvas element (see 3.2.2 HTML5 Canvas). HTML5 is often referred to as a collection of technologies, including various newer CSS specifications (such as media queries [25], the CSS images [26] and CSS Transitions [27]) that allow for more control over user interface styling (gradients, animation etc). New Javascript APIs, such as the Websockets API [28] and the File system API [29] are also considered part of HTML5, as well as WebGL [30] and more[5].

Support for HTML5 features varies between different browsers and versions [31]. To get a full picture an analysis of each feature is needed.

3.2.2 HTML5 Canvas

Canvas is an HTML element that represents a resolution dependent bitmap on which it is possible to draw graphics using JavaScript. Graphics may be constructed using a script, thus allowing for dynamic rendering and redrawing, or by inclusion of bitmaps. The canvas API does support text, but only when drawn on one line or along a path (not as a text area). When graphics are updated in a canvas-element the whole element is redrawn on the screen [32](chapter 4.8.11).

The WebGL specification allows for 3D graphics to be drawn on a canvas element [30].

To facilitate developing on the canvas element third party Javascript libraries have appeared that adds functionality on top of the Canvas API. For example, paper.js [33] adds a scene-graph, a DOM and utility functions to draw vector graphics.

The canvas element is supported by newer versions of all major browsers [31].

[5] http://www.html5rocks.com/en/

3.2.3 Scalable vector graphics

SVG is and an XML-based language that aims to provide rich graphics content. It describes two-dimensional vector graphics and text with support for animation, video and audio [34]. It is a W3C standard and currently exists in three different profiles (W3C recommendations) that aim different devices: SVG Full 1.1, SVG Tiny 1.2 and SVG Mobile 1.1 where the two latter are subsets of the first [34].

SVG is compatible with other W3C standards, i.e. it is as already mentioned based on XML, but it can also be used with a subset of CSS for styling, SMIL for animation and has a DOM for document manipulation with ECMAScript/Javascript [34].

As with HTML, SVG is not coupled with a runtime environment, but is dependent on browser makers to integrate native support for it. Browsers are however getting better support for SVG with recent versions, but there are still inconsistencies in the implementations [35].

SVG graphics can be authored in vector graphics editors, such as Inkscape[6]. A few Javascript libraries exist, e.g. Raphaël[7], which facilitates programmatic manipulation of the SVG elements.

3.2.4 JavaFX

JavaFX is according to Oracle their means to improve Java as a rich client platform. The company describes it as "a lightweight, hardware-accelerated Java UI platform for enterprise business applications" [36]. It supports hardware accelerated graphics, audio, video and embedding of HTML documents (using Webkit render) [37].

By reading the documentation [38] it is a bit unclear how JavaFX handles resolution independence. It seems that some components, UI controls and layouts are resizable, while other, such as text and shapes are not [39].

The technology supports separation of code and graphics, by the FXML markup language [40] and a CSS3 subset [41].

Since JavaFX uses the Java language it has a large set of tool to aid development. Oracle also provides some tools specially aimed at JavaFX[8]

[6] http://inkscape.org/

[7] http://raphaeljs.com/

[8] JavaFX tools: http://www.oracle.com/technetwork/java/javafx/tools/index.html

The JavaFX runtime is installed with the Java Runtime Environment [36], and has currently support for Windows with Mac OSX and Linux on the roadmap [42].

3.2.5 Adobe Shockwave Flash

Adobe Flash is a collection of technologies for creating multimedia content. It consists of the swf format, which is a binary format and is used to represent vector graphics, text, video and sound. Flash uses a scripting language called ActionScript (an implementation of the ECMAScript) to add interactivity and animation to the swf-files [43].

To display the swf-files a player such as Adobe Flash Player is required. It is the industry standard and according to Adobe it has a market share of 98 per cent and up, depending on market and version of the player [44]. The Flash Player is installed as a plugin to the web browser. Flash applications can also run outside of the browser, in the runtime environment Adobe Air[9].

Adobe provides tools and frameworks for authoring flash applications[10]. There also exist open source tools, such as the editor Flash develop[11].

3.2.6 Microsoft Silverlight

Silverlight is a cross platform implementation of Microsoft's .NET framework with focus on rich interactive applications and can be run as a plug-in to a web browser or as a native application [45]. It based on vector graphics [46] and supports video, audio, text and graphical effects (such as animations) [45].

It uses a subset of the WPF[12] for user interface elements and the markup language XAML[13] to define the elements [45]. A Silverlight application can either use a Javascript API with Javascript code executed in the browser, or use a managed API with languages such as C# or IronRuby running in Silverlight's runtime environment [47].

---

[9] Adobe AIR is a system runtime that allows to run Flash outside of the browser (https://www.adobe.com/products/air/faq.html)

[10] Flash tools: http://www.adobe.com/flashplatform/

[11] http://www.flashdevelop.org/

[12] Windows Presentation Foundation (http://msdn.microsoft.com/en-us/library/ms754130.aspx)

[13] Extensible Application Markup Language (http://msdn.microsoft.com/en-us/library/ms752059.aspx)

Microsoft provides authoring tools for Silverlight in their Visual Studio product[14].

Silverlight supports Windows and Mac OSX [48]. Linux is supported by the open source Moonlight[15] project. However, the Moonlight project does not keep up to speed with the Silverlight releases, currently being three versions behind [49].

[14] Silverlight tools: http://www.silverlight.net/downloads

[15] http://www.mono-project.com/

Chapter 4
Evaluation of findings

The findings of Chapter 2 Set-top box and IPTV services study are analyzed based on what impact and requirements they put on a STB user interface. Then, the technologies found in Chapter 3 Technology study are evaluated based on the criteria defined in section 1.1 Problem description and how well they handle the coming IPTV services.

4.1 Requirements put on the user interface by future IPTV services

4.1.1 Internet content

To support Internet content, such as Web 2.0 services, the set-top box must be able to fetch information from external web services. The web browser is ideal for that task, hence no extra work is needed to allow the user interface to communicate with Internet based services.

As for running Web 2.0 services, such as displaying micro blogging feeds or other types of user or community generated updates, the STB needs to be able to display these feeds in a timely manner. For other types of web apps, the STB must be able to draw the user interface for that application. All of these applications are, by definition, built to run in a web browser. With that in mind it should be possible to use them in a STB web browser.

The obstacle could be the performance of the STB. More graphics might be displayed on the screen at the same time, which would require the browser to display all the extra graphics in a smooth way. But adapting the applications to the hardware prerequisites and the fact that next generation's set-top boxes get better performance, is most likely sufficient to bring Internet content to the set-top box and satisfyingly present it to the user.

For streaming video services, such as YouTube, the user interface is not affected other than that the quality and the resolution of the streaming content may differ between sources and may look bad when displayed on a big screen. The set-top box's (or the browser's) video player must have the most common video and audio codecs installed in order to playback the content.

4.1.2 Widget system

The purpose of a widget system is to enable small applications to run independently on the system and these applications could easily be created by third-party providers.

The widget engine would replace the web browser as user interface renderer. It will therefore be up to the STB manufacturer to support the widget engine in the same way that the web browser is supported today (see section 1.7 Set-top box architecture), with access to lower-level features such as the video player and graphics libraries and communication protocols.

Hardware requirements for set-top boxes with a widget system will depend on the implementation. The Yahoo Connected TV system, for example, builds on Yahoo Widgets which was originally created for the desktop environment [50]. The system is built to run on the Intel CE3100 chipset [51], which is the most powerful chipset on the market at the moment. But as shown in 2.2.1 Set-top box history, more capable chipsets are emerging which implicates that hardware requirements will be of lesser concern if this type of system will become standard for IPTV user interfaces.

The impact this type of system will have on a STB user interface is substantial. The widget system is also likely to dictate what technology and GUI frameworks to use.

4.1.3 Third-party applications

A widget system (see above) is one way of allowing third party applications. Another way would be to create a custom third-party system based on the current user interface. How to create a third-party system is subject for a thesis of its own and will not be handled here. The crucial part, though, is to provide an API for third-party developers that grants access to various features needed to create the application. The user interface technology must provide the tools to enable such an API and allow external code to run in it.

4.1.4 Home networking

The required functionality to allow the set-top box to communicate with e.g. DLNA devices, is something that resides in layers below the user interface, and needs to be addressed by the set-top box manufacturers.

Marcin Pakula identifies the biggest problem in using the set-top box as the home network's central point to be the user interfaces (see 2.3.2 Future IPTV services). He means that for the idea to work the user interfaces must be designed to handle the task of gathering all media, distributed on the various devices, and present it to the user in a straightforward way. That may be a difficult task, but it is mainly a design and usability problem.

The user interface technology must be able to communicate with e.g. the DLNA client in the STB or directly to the DLNA server. However that is more of an API issue to solve than a technology issue.

4.1.5 Mobile devices

The purpose of using a mobile device to interact with the IPTV service is to divide the user interface into more screens. For instance, it might be easier to type a message using the mobile phone instead of the remote control, or perhaps the user would want to interact with the EPG list on the mobile phone while continuing watching the program on the big screen.

In both cases the mobile device can have its own GUI which is separate from the STB GUI. In the second example, the user merely consumes the service on the mobile device in parallel, with no interaction with the STB user interface. In the first example, the mobile device interacts with the user interface of the STB. In that case a communication channel between the STB and mobile device must be established. It does not, however, put any new requirements on the STB GUI.

4.2 Technology evaluation

The web technologies found in Chapter 3 Technology study are here evaluated using a decision matrix. As criteria the result of section 4.1 Requirements put on the user interface by future IPTV services are used as well as the criteria from Section 3.1 Criteria. The results from section 4.1 are given the weight 1.

As described in section 1.4 Methodology, the decision matrix is an L-shaped matrix that tabulates the alternatives to the criteria. It then scores each alternative relative to each criterion according to a predefined scoring system. For the purpose of this thesis a Pugh matrix is used.

The Pugh matrix scoring system is based on the principle of "worse, same or better", where 0 is same or equally good, -1 is worse and +1 one is better. It is also possible to add extra levels to the scoring, to get a fine scale [2].

To add a bit of flexibility, the scoring system used here consists of a 5 five-point scale, -2, -1, 0, +1, +2, where 0 means that the alternative is rated equal to the

datum. +1 means better, +2 means significantly better, -1 means worse and -2 means significantly worse.

4.2.1 Platform independence

Platform independence comparison		
Technology	*Motivation*	*Score*
HTML	HTML is supported natively in all web browsers. Virtually all operating systems have a web browser available.	Datum
HTML5 Canvas	The Canvas element is part of HTML and is thus also designed to be platform independent. It is not yet supported in all browsers but is gaining continuously gaining more support.	0
SVG	SVG is like HTML a W3C specification and designed to be platform independent. Browser support is growing for each version. However implementations differ in features and in some cases also behavior.	-1
Flash	Flash requires a third-party plugin to run. With that in place it is platform independent. However, for Flash to be used on a new operating system additional software is needed.	-2
Silverlight	Silverlight does not have any official support for Linux (except for the Moonlight project).	-2
JavaFX	JavaFX 2 is so far only supported on Windows, having OSX and Linux on the roadmap. Eventual support for set-top boxes or other embedded devices are not	-2

Table 4-1 Platform independence comparison

4.2.2 Resolution independence

Resolution independence comparison		
Technology	*Motivation*	*Score*
HTML	HTML is bitmapped, hence it is resolution dependent. Units are in absolute sizes (pixels),	Datum

HTML5 Canvas	Even though canvas is drawn on a bitmap and is resolution dependent it can be used to draw vector graphics with help from third party libraries.	+1
SVG	SVG is resolution independent by definition.	+2
Flash	Flash, just as SVG, is based on vector graphics and resolution independent	+2
Silverlight	Silverlight also uses vector graphics	+2
JavaFX	JavaFX has support for vector graphics, but it was unclear how vector shapes behaved when resizing a document, a more thorough study is needed to conclude the final score	+1

Table 4-2 Resolution independence comparison

4.2.3 Rich user experience

Rich user experience comparison		
Technology	*Motivation*	*Score*
HTML	HTML (in combination with Javascript and CSS) provides the tools for good separation between graphics and code. Graphical effects such as animation can be achieved partly with CSS and party with Javascript.	Datum
HTML5 Canvas	Graphics drawn on a canvas must be done programmatically and does not provide the same clear separation between code and graphics (and styling).	-1
SVG	SVG provides good separation because it, just as HTML, is designed to work standalone. It has good support for animations and other graphical effects.	+1
Flash	The Adobe Flex framework provides good support for this. Flash is designed to give a rich user experience.	+1
Silverlight	The XAML markup language provides good separation. Silverlight have rich graphical effects build into it.	+1

JavaFX	JavaFX's FXML markup and CSS-support provides good separation. It is bundled with rich graphical effects.	+1

Table 4-3 Rich user experience comparison

4.2.4 Development support

HTML has a huge development base and community, but so does .NET, Flash and Java. Flash, Silverlight and JavaFX are given higher scores than HTML due to the fact that they all ship with dedicated development tools.

Development support comparison		
Technology	*Motivation*	*Score*
HTML	There is a huge development community around HTML (and CSS and Javascript)	Datum
HTML5 Canvas	As support for the canvas element grows, the community around it is also growing.	-1
SVG	SVG being a mature technology have not nearly as a large developer base as HTML, neither are the many tools and frameworks. The fact that SVG works with Javascript weights up this criteria.	-1
Flash	The flash community is very mature and has a stable development environment.	+1
Silverlight	The .NET community is available for Silverlight users, with mature tools.	+1
JavaFX	The technology is not as mature as Flash and HTML, but having the whole Java community and tools available weights up.	+1

Table 4-4 Development support comparison

4.2.5 Future IPTV services

HTML5 Canvas has no multimedia support, but because it is a part of HTML, it is assumed that the HTML5 audio and video elements can be used in combination with the canvas element to display multimedia.

Internet content

All technologies have the networking functionality to communicate with Web services. All technologies get score 0.

Widget system

None of these technologies provides a widget system by default. The task of creating such a system with either technology is too big to be handled here, and could be a subject for a thesis of its own. No scores are given.

Third party applications

As with the widgets system, providing a system for third party application is subject for a thesis of its own. None of the technologies have native support, but there is nothing that indicates that it should not be possible to create such a system. No scores are given.

Home networking

All technologies are suited to present this type of content (audio, video, images). Also, further study of what audio and video codec's are available for the technologies on different platforms is needed to make a fair comparison. All get score 0.

Mobile devices

As stated in 4.1.5 Mobile devices, no extra requirements is put on the STB GUI. No score is given.

4.2.6 First evaluation results

SVG, Flash and Silverlight score equally well after the weights have been applied.

Technology comparison matrix						
Criterion	*Weight*	*Canvas*	*SVG*	*Flash*	*Silverlight*	*JavaFX*
Platform independence	4	0 (0)	-4 (-1)	-8 (-2)	-8 (-2)	-8 (-2)
Resolution independence	3	+3 (+1)	+6 (+2)	+6 (+2)	+6 (+2)	+3 (+1)
Rich user experience	3	-6 (-2)	+3 (+1)	+3 (+1)	+3 (+1)	+3 (+1)
Development support	2	-2 (-1)	-2 (-1)	+2 (+1)	+2 (+1)	+2 (+1)
Internet content	1	0	0	0	0	0
Third party applications	1	0	0	0	0	0
TOTAL		**-3**	**+3**	**+3**	**+3**	**0**

Table 4-5 Results of the technology evaluation

4.3 Performance test

Testing performance only makes sense if it is done on the target device (see 1.8 Note on performance). At the time SVG was the only technology besides HTML that was available in the lab environment. It was decided to start the performance tests on these two technologies, while efforts were made to get additional equipment to the lab.

The STB used for the tests is the Motorola 1910 (see section 1.6 Equipment). It runs the Mozilla 1.7 browser (HTML), and the Ekioh SVG browser (SVG). The Ekioh browser has support for the SVG Tiny 1.2 specification, extended some features from the SVG 1.1 specification and some proprietary features [52].

4.3.1 Javascript benchmarks

JavaScript handles the application logic and much of the interactivity, so it makes sense to test the JavaScript engines of the web browsers.

The Sunspider[16] JavaScript benchmark was used here. It tests core Javascript performance, and was run in both browsers on the STB.

Sunspider test	
Browser	*Time (ms)*
Mozilla 1.7	701358
Ekioh	201486

Table 4-6 Results of the Sunspider test on the Motorola 1910-9 STB running Mozilla 1.7 and Ekioh

4.3.2 Graphics rendering test

A simple graphics rendering test was created with the set-top box in mind. Inspired by the SVG Chamber[17], developed by Dr. David P. Dailey for his contribution to SVG Open 2007, which is a SVG graphics test to compare performance in different browsers [53], a simple graphics benchmark was created that plots different shapes and images on the screen and measures the time consumption. This was kept simpler than the SVG chamber in order for it to function with both HTML and SVG, considering the different graphical capabilities of the two.

The combination of the JavaScript benchmark and this test will give a good hint on how well the technologies perform.

The test was created in two versions, HTML and SVG. The HTML version draws div-tags and images. The div-tags are given random sizes and color-fills, while the images are scaled to an arbitrary size up to twice the original size. All elements are given a random opacity. The SVG version functions the same way but draws rectangles, circles or ellipses instead of div-tags. It is possible to set what area that should be plotted on, the number of elements to be drawn and turn opacity, fill and image scaling on and off. Three tests with different amounts of elements were done at both SD and HD resolutions. All random numbers are seeded to assure the same graphics are plotted for both technologies.

[16] http://www2.webkit.org/perf/sunspider-0.9/sunspider.html

[17] http://srufaculty.sru.edu/david.dailey/svg/SVGOpen2007/SVGChamber99.html

Graphics rendering test		
Test	*Ekioh (ms)*	*Mozilla 1.7 (ms)*
SD RESOLUTION (720 X 576)		
100 rectangles	1385	5443
100 rectangles and 100 images	2682	10829
500 rectangles	6834	27894
HD RESOLUTION (1280 X 720)		
100 rectangles	1432	-
100 rectangles and 100 images	2799	-
500 rectangles	7091	-

Table 4-7 Results from the graphics rendering test on the Motorola 1910-9 STB

4.3.3 Results

The Ekioh SVG browser gave significantly stronger results than the Mozilla 1.7 browser in both Javascript performance and graphics rendering. SVG scores +2.

4.4 Results of the technology evaluation

Unfortunately we were unable to provide an STB running Flash, Silverlight or JavaFX within the time frame of this thesis. However, the large performance gains SVG showed compared to HTML gives good confidence to continue with the prototype without having tested all technologies. The results of the technology comparison are presented in the Pugh matrix below.

Technology comparison matrix						
Criterion	*Weight*	*Canvas*	*SVG*	*Flash*	*Silverlight*	*JavaFX*
Performance	5	-	+10 (+2)	-	-	-
Platform independence	4	0 (0)	-4 (-1)	-8 (-2)	-8 (-2)	-8 (-2)
Resolution independence	3	+3 (+1)	+6 (+2)	+6 (+2)	+6 (+2)	+3 (+1)
Rich user experience	3	-6 (-2)	+3 (+1)	+3 (+1)	+3 (+1)	+3 (+1)
Development support	2	-2 (-1)	-2 (-1)	+2 (+1)	+2 (+1)	+2 (+1)
Internet content	1	0	0	0	0	0
Third party applications	1	0	0	0	0	0
TOTAL		**-3**	**+13**	**+3**	**+3**	**0**

Table 4-8 Results of the technology evaluation

Chapter 5
Case TeliaSonera

Based on the results of the technology evaluation it was decided that the current user interface should be ported to SVG. HTML and SVG share two core features - both have a DOM and have support for JavaScript. Because of those two facts most of the current GUI code can be reused.

This chapter explains how the user interface is constructed, how SVG can replace HTML and how the SVG port was performed.

5.1 Requirements

Listed here are the requirements of the new GUI. They are derived from the criteria used in the technology evaluation and are listed in priority order.

Requirements on the SVG GUI	
Requirement	*Description*
R1: Performance	The new GUI shall load faster and be more responsive.
R2: Platform independent	The new GUI shall run on all SVG browsers with enough SVG and Javascript support.
R3: Resolution independent	The new GUI shall be resolution independent. Graphics shall be sharp in HD.
R4: Skinnable	The new GUI shall support a way of separate its visual representation so that it can change its appearance without modifying the logic.
R5: Rich graphics	The new GUI shall use animation and richer graphics.

Table 5-1 Requirements on the SVG GUI

5.2 Limitations

During the SVG port no access to the set-top box functionality was possible because full support for the Ekioh SVG engine was not implemented by Motorola yet. It was not possible to use the set-top box's video and audio player and subtitle functionality. Neither was it possible to use encrypted content since the encryption system that TeliaSonera use was not accessible.

Therefore this chapter will only focus on porting the GUI graphics layers from HTML to SVG, not making the application logic to fully work with the new browser.

5.3 Overview of the current GUI

The TeliaSonera graphical user interface is a Javascript application. JavaScript is the core of the application and handles both the application logic and the graphical presentation (by manipulation of the DOM).

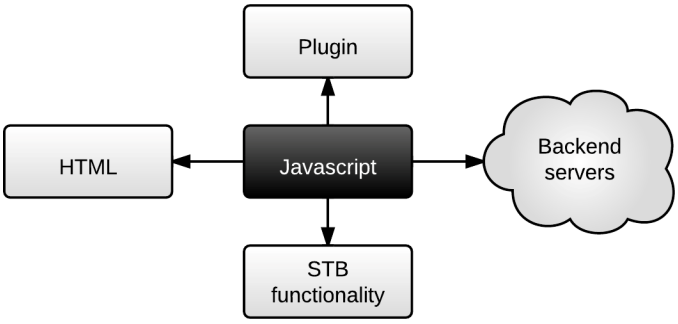

Figure 5-1 Javascript is the core of GUI system.

The graphics in the application is built on of two concepts; widgets and views. Widgets are JavaScript objects that consists of a DOM tree of HTML elements and various functions to alter the HTML elements. A view consists of a collection of widgets that together form the user interface. It also contains functionality to update the information shown on the screen. See Figure 5-2 for an overview of how the application is designed.

The STB exposes a Javascript API to allow control of lower level functions, such as writing to disk and controlling the media player. An npruntime[18] plugin handles

[18] More information about the npruntime API can be found at
https://developer.mozilla.org/En/Gecko_Plugin_API_Reference:Scripting_plugins

login to the backend server and fetches EPG-data through multicast, among other things.

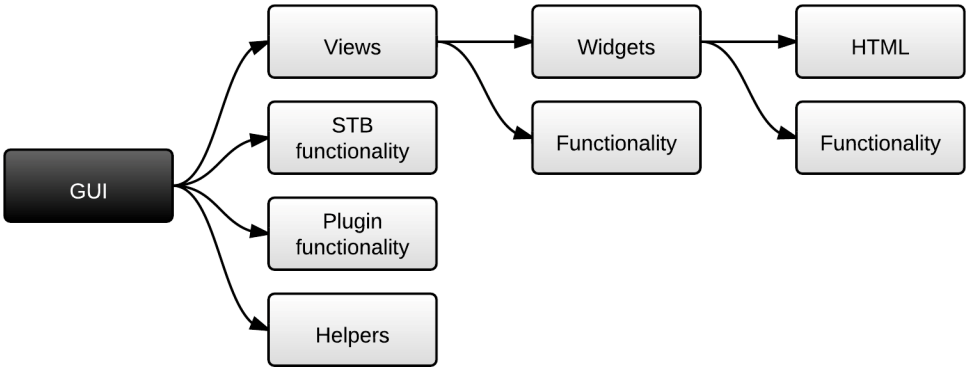

Figure 5-2 Overview of how the application is designed.

5.3.1 Widgets

There are at total of 12 widgets used in the system. They represent simple graphical components such as windows, images and text areas, as well as more complex elements such as lists, hot spots and input areas.

The widgets are all based on the HTML `div` element coupled with JavaScript functionality and CSS. They do not use HTML specific elements for text areas, input areas and so on; instead they use JavaScript and DOM manipulation to create similar functionality. Basic functionality for most widgets are the ability to set or change the widgets position, size, visibility, background color and background image. The appearance of the widgets can be controlled by assigning CSS classes. The user interface uses an external style-sheet document that contains these classes. All widgets use absolute positioning, which means the elements are positioned according to absolute pixel values rather than being part of the automatic layout flow[19].

Table over widgets	
Widget	*Description*
Window	The `Window` widget is the most basic widget. It can have a background image or color and is basically a container that other widgets can be appended to.

[19] http://www.w3.org/TR/CSS2/visuren.html#absolute-positioning

IBox	The `IBox` widget has the same purpose as a window widget – to contain other widgets. The IBox consist of an `iframe` that gets elements appended to it. The `iframe` is used as an optimization for the Mozilla 1.7 browser which slows down significantly when the DOM-tree gets too large.
TextArea	The `TextArea` widget is used for containing text. It can have a background image and a `Scrollbar` widget. It keeps track of the number pages is has and can be set to scroll automatically.
InputArea	Used for letting the user input text. Can be set to take SMS-styled input, i.e. using the letters assigned to the number buttons on the remote control, or password input where it only displays stars.
Image	Used for displaying images.
HotSpot	Used for shortcuts in the application. The `HotSpot` widget may be highlighted when selected and can have a list of neighbours of other `HotSpot` widgets that can be selected using the arrow buttons on the remote control. A `HotSpot` can show an image and has an action attribute that tells it what to do when clicked on.
ListItem	The `ListItem` consists of background color and text string. It provides functionality to change its text and appearance.
List	The `List` widget is used throughout the application. It takes a set of `ListItem` widgets and combines them to a list. It provides functionality for various ways to scroll the list, and to update it.
DefaultListHandler	The `DefaultListHandler` widget is not a graphical widget but helper to the List widget. It has functionality to control and update the `ListItem` widgets in the list.
Scrollbar	The `Scrollbar` widget comprises the scrollbar that may be added to Lists widgets and `TextArea` widgets that does not fit the screen.
Table	The `Table` widget is used to create an HTML table.

Table 5-2 Widgets in the GUI

5.3.2 Views

The purpose of the view is twofold: present the user interface and respond to user interaction with appropriate actions.

The views are JavaScript objects that holds a number of widgets. When a view gets initialized it creates widgets and appends them to the document tree. If a view is inactive it hides all of its widgets. When a view responds to user action, it is usually about updating itself and its widgets with new information, communicating with other objects in the system (i.e., the video player or the TV handler) or changing to another view.

5.3.3 Other

Except for widgets and views, other objects exists that control different features in application that are not directly connected to the visual appearance. These consist of a pre-loader object, a system object, the video player controller, set-top box environment and the plug-in wrapper plus some other helper objects.

5.4 SVG and HTML

Both SVG and HTML uses variations of the DOM standard to represent a document. SVG Tiny 1.2 uses the SVG uDOM – a subset to the full DOM – which has some features from both the DOM2 and the DOM3 specifications [54], but is kept smaller to improve performance on resource limited devices. Basic behavior, such as to create, insert and delete elements in the DOM are similar to the different versions. That means that the same approach to create the graphical elements can be used with SVG as with HTML.

Both technologies also support CSS. Even though SVG Tiny 1.2 does not support complete customization of its elements with CSS, most of the external CSS document can be reused. However, SVG Tiny 1.2 does not require the user agents to support external style sheets [55](chapter 6.2). In order to achieve full interoperability between SVG Tiny compatible browsers that feature should be used as little as possible. The Ekioh browser does have support for external style sheets, but this feature will have lesser importance in the SVG version, as described in section 5.6 Rough port.

5.4.1 The div and g elements

The most common element in the user interface is the HTML `div` element. The element can be used to hold other elements, functioning as a window, and have an optional background color or image. It can also hold text content and function as a text area. All widgets, expect the `Image` and the `Table` widgets, use the `div` element extensively.

The `div` element can be translated into the SVG `g` element when it acts as a window or container of other elements. The `g` element must be coupled with an SVG

shape, such as the `rect` element, or an `image` element in order to represent a background color or background image respectively.

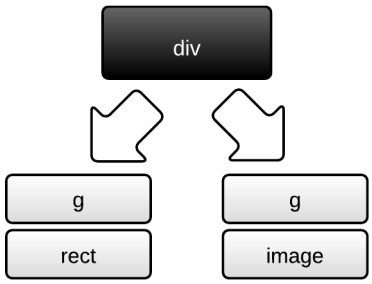

Figure 5-3 The `div` element with background color translated to SVG

When the `div` element contains text it has to be translated into a SVG `text` or `textArea` element, again coupled with a `rect` or `image` element to represent the background.

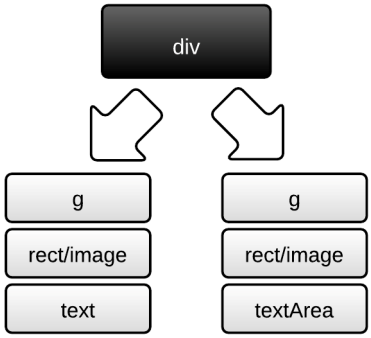

Figure 5-4 The `div` element with text content translated to SVG.

SVG Tiny 1.2 does not support clipping of content within a `g` element (or any other element) [54](chapter 12). The element has no `width` and `height` properties and the CSS property `overflow` does not apply. It is not possible to control the width and height in a `g` element – the element adapts its size after its content. The problem has been partly solved by setting the desired width and height to the `rect` element that represents the background. This way the visual appearance will be equivalent but no clipping will occur to content that exceeds the size of the `rect` element. The result may be unsatisfying if caution is not taken when appending elements to the g element.

5.5 Implementation process

The process to port the user interface to SVG has been divided into two steps. The first step, referred to here as the “rough port”, basically replaces all HTML elements with their SVG counterparts. This should result in a working SVG version of the user interface. The second step, the “template system”, utilizes the expressive power of SVG and use SVG files as templates that represent widgets and views.

5.6 Rough port

To translate the `div` element into SVG is as described in section 5.4 SVG and HTML fairly straight forward. Styling and rearranging of HTML elements (and widgets) is today handled by CSS, through manipulation of element's `style` attribute. Because not all display properties in SVG can be edited using CSS, it was decided to explicitly use SVG attributes instead of CSS properties in order to style and rearrange the elements in the SVG version. Positioning of SVG elements is defined by x and y coordinates, which maps very well to the absolute positioning model of HTML. Hence, all code that lays out widgets can be reused.

5.6.1 Widgets

The widgets were ported according to what is described in section 5.4.1 The div and g elements. Table 5-3 shows what elements are used in the HTML version for each widget and how they were translated into SVG.

The scroll functionality of the text area could not be implemented using standard SVG elements since SVG Tiny 1.2 does not support clipping of content. Ekioh implements a proprietary element, called `clippedGroup`, which only renders content that fits within its specified area [52]. Ekioh also implements an element called `scrollableGroup`, which inherits from `clippedGroup` and allows for fast scrolling of its content [52]. By using a `scrollableGroup`, the scrolling functionality of the `TextArea` widget was achieved.

5.6.2 Views

Because the `IBox` widget was discarded and replaced by the `Window` widget, all views using an `IBox` widget had to be modified.

Some views implement their own `ListItem` widgets, which may combine standard HTML elements with other widgets. These were ported in the same way as the rest of the widgets (see Table 5-3).

5.6.3 SVG helper object

An SVG helper object was created to facilitate repetitive tasks, such as to set size and position of a widget or to show and hide it. The object encapsulates SVG specific JavaScript calls for setting the concerned attributes.

In example, the method call

```
Telia.SVG.setSize(node, width, height);
```

replaces the following two calls

```
node.setAttributeNS(null, “width”, width);
node.setAttributeNS(null, “height”, height);
```

This way, the overall code size is kept down and most SVG specific functionality is kept at one place.

5.6.4 Layers

The HTML user interface uses the CSS `z-index` property to place the widgets in different layers. SVG does not support z-index. The `LayerManager` object mimics z-index by using root level g-elements that represent the different z-indexes. When a widget is created with a z-index, it is added to the g-element that represents that z-index.

5.6.5 Video

The video playback problem was solved by implementing a workaround that uses the `video` element instead of the set-top box's video player. The `video` element is capable of displaying video streams and can be used as a replacement to the video player in order to display unencrypted content.

The user interface is adapted to run with no video and sound in the Mozilla Firefox browser on a PC for debugging purposes. To allow potential support for additional platforms the application keeps all platform specific functionality separated from the rest of the application. The Firefox adaption replaces the video player with an `image` element and discards the video streams. That code was used as a base when the Ekioh version was created. The new version simply substitutes the `image` element with the SVG `video` element and redirects the video streams to that element. That way unencrypted video is supported.

Table over HTML and SVG elements used in the widgets			
	HTML elements		*SVG elements*
WINDOW			
	Used as container with background color or background image.	<g>	Used as the container.
		<rect>	Used for setting the size of the widget and as background color
		<image>	(Optional) Used as background image.
IBOX			
Deprecated. The `Window` widget will replace the IBox widget.			
TEXTAREA			
	Used as container for the widget with optional background image	<g>	Used as the container.
		<image>	(Optinal) Used as the background image.
	Used if the text area is scrollable. May contain a textNode.	<scrollableGroup>	Used if the text area is scrollable.
<textarea>	Used to contain text that should be able to line break.	<textArea>	Used to contain all text.
textNode	Used for text without line breaks		
INPUTAREA			
	Used as container for the widget with optional background image	<g>	Used as the container.
		<rect>	(Optinal) Used as the background.
	Used if the text area is scrollable. May contain a textNode.	<scrollableGroup>	Used if the text area is scrollable
	Used to contain the textNode.	<text>	Used to contain all text.
textNode	Contains the text	<tspan>	Contains the text
	Container for the blinking character	<tspan>	Contains the blinking character

textNode	Contains the blinking character		and its text content
	IMAGE		
new Image()	JavaScript function for creating an image.	<image>	The image.
	HOTSPOT		
new Image()	JavaScript function for creating an image. Represents the HotSport image. May have and background color that exceeds the image size.	<g>	Used as the container of the widget.
		<rect>	Used for setting the size of the widget and as background color
		<image>	Used for setting the size of the widget and as background image.
	LISTITEM		
	Used as container for the widget with background color.	<g>	Container for the widget.
		<rect>	Used for setting the size of the widget and as background color.
<textSpan>	Used to contain the textNode.	<textArea>	Contains the text.
textNode	Contains the text.		
	LIST		
	Container for the widget.	<g>	Container for the widget.
	DEFAULTLISTHANDLER		
-	No HTML elements used.	-	No SVG elements used.
	SCROLLBAR		
		<g>	Container for the widget.
	The scrollbar.	<rect>	The scrollbar.
	The slider.	<rect>	The slider.
	TABLE		
Deprecated. The table widget uses the HTML table elements to draw a table. Not used in SVG.			

Table 5-3 All widgets and how they were translated to SVG

5.7 Template system

The purpose of the template system is to make the GUI "skinnable". That is, to let the developers alter the appearance of the GUI without changing the code. The "look and feel" of widgets and views could be specified in SVG template files instead of in the JavaScript code. That would separate the visual appearance of the user interface from the application code and allow for multiple "skins" to be designed. It would further eliminate the need of having an engineer to apply every visual change, thus allowing a graphics designer to work directly with the SVG template files. Easy editing of the templates should be possible due to the fact that the SVG format is supported by popular vector graphics editors (see 3.2.3 Scalable vector graphics).

In the paper "SVG: A Key Element in Achieving Product Differentiation and Competitive Advantage in the DVB Market" [56] it is described how Cabot Communications integrated SVG into their DVB applications to allow their customers to easy alter the appearance of the user interface. Their applications are built in C++ and the user interface framework consists of a number of widgets, much like the widgets used in the TeliaSonera user interface. Each of these widgets is represented by an SVG file. The widgets themselves are divided into different parts, each part represented by a group of SVG elements and identified by the C++ application using the SVG elements' `id` attribute. The SVG elements are then bound to the C++ application at runtime. The Cabot SVG templates are based on a small subset of SVG Basic 1.1 that only includes different shapes. Cabot decided not to extend the SVG vocabulary in order to keep the system compliant to the SVG standard and as simple as possible. To tell the application where to position and how to render text, a `rect` element is used. It has a special id value that tells the application not to render actual element, but instead to use its attributes as instructions for text rendering. The model of using attributes of a specially named `rect` element is used throughout the template system in order to provide application specific information.

A similar concept may be used when creating a template system for the TeliaSonera user interface; let widgets and views be represented by individual SVG files and let the application identify the widgets using an id. The Ekioh engine is a more capable SVG renderer than what Cabot used, having full support for SVG Tiny 1.2, some SVG Full 1.1 features and a couple of proprietary extensions. Thus, there is e.g. no need to use a `rect` element to represent text formatting. SVG Tiny 1.2 also supports all of its elements to include attributes from foreign namespaces [55](chapter

19.1). That way, application specific information can be included as attributes of the concerned element instead of attributes of an extra `rect` element. Furthermore, the JavaScript `XMLHttpRequest`[20] (XHR) object is well suited to load the SVG template files into the application at runtime, since it can provide entire XML documents as DOM objects.

What is described above should be sufficient to create a template system for the TeliaSonera user interface. The following sections defines the structure of the SVG templates, describes how a template handler is created and how the widgets and views gains support for the templates.

5.7.1 Template structure

If the structure of the templates is based on the structure of the widgets, minimal effort should be needed to translate the templates into widgets. Table 5-3 shows that most widgets have a container (a `g` element) that holds all elements of the widget. That container element will be used to identify the widget in the template. To do that a new temporary namespace is used, named `telia`[21]. The definition of the `telia` namespace is:

```
xmlns:telia="http://iptvlogin.telia.se/2008/"
```

Every template element that should be recognized by the application has a `telia:id` attribute. If a widget contains more than one element, such as a `rect` element for the background, that element will have the same value of its `telia:id` as its parent plus an added colon and the word "bgNode". For instance, a `Window` widget created in the JavaScript code may contain the following lines of code:

```
this.dom = document.createElementNS(NS_SVG,"g");
bgNode = document.createElementNS(NS_SVG,"rect");
bgNode.setAttribute( "fill", "none");
Telia.SVG.setSize(bgNode,571,343);
this.dom.appendChild(bgNode);
```

That would render the following SVG structure:

```
<g>
  <rect fill="none" width="571" height="343"/>
</g>
```

That structure in an SVG template using the rules explained would be as follows:

---

[20] Read more at http://www.w3.org/TR/XMLHttpRequest/

[21] No definition resides on the URI. If TeliaSonera choose to implement this template structure, the namespace URI may be subject to change. It should also get an end point.

```
<g telia:id="window">
  <rect telia:id="window:bgNode" fill="none" width="571"
  height="343"/>
</g>
```

The id of the top element is of great importance for the view templates. It is used to connect a template element to a widget in the view.

The view template consists of several template elements. Every widget in the view must have a representation in the template. For instance, the SVG template element above could be a representation for following `Window` widget in a view:

```
this.window=
new Telia.Widgets.Window(0,0,571,343,1001,parentNode);
```

The name of the variable is matched to the `telia:id` attribute of the template element. Other attributes in the telia namespace exists, which provide the application with specific information about the widget they are attached to. For example, the `telia:type` is used if the widget is of a type that needs additional processing, such as the `List` or the `HotSpot` widgets. This will be further explained in the next section. An example of a SVG template can be found in Appendix A: SVG template file.

Template processing

All template functionality have been collected in a new JavaScript object, called `Templates`. This object consists of several objects designated for specific tasks in the template processing. The `Loader` object contains functionality to load the SVG templates into the application, using the XHR object. The `Template` object processes a loaded template file into template widgets. The result is then stored in the object. The template widgets serve as intermediate storage between the template element and the actual widget. Widgets that have special properties, such as action or navigation properties that cannot be specified by standard SVG, are stored as special template widgets. These special widgets are identified by the `telia:type` attribute. All other widgets are stored as normal template widgets. A list over the available template widgets can be found in Tabell 5-1. The processes of loading SVG template files and to process them into template widgets are shown in Figure 5-5 and Figure 5-6.

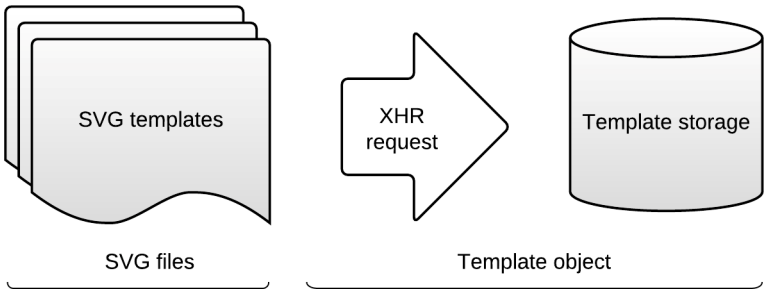

Figure 5-5 The process of loading the SVG template files

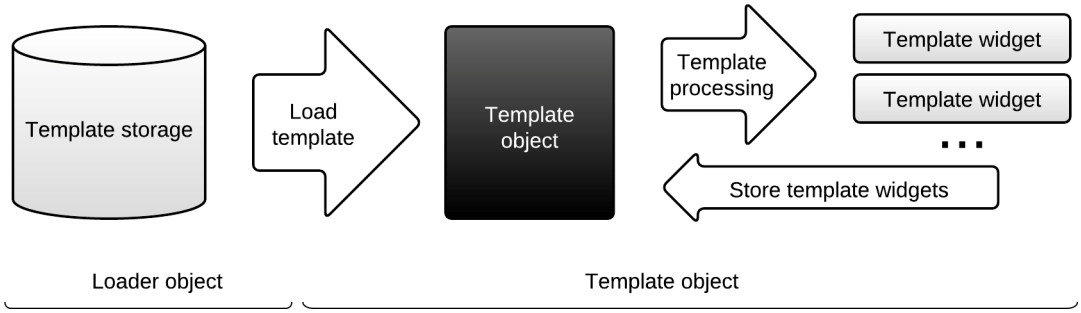

Figure 5-6 The template system processes the SVG files into template widgets.

When a view is initialized it creates a new `Template` object and tells it to process one of the loaded template files. The result is an object that contains various template widgets. When the view then continues in ordinary fashion to create its widgets, it passes along an extra variable that tells the widget what template to use. For example, the following code creates a template based on the SVG template called "Template".

```
var template = new Telia.Templates.Template("Template");
```

When the widgets are created in the view, the corresponding template widget is then passed along in the constructor. In the following code it is called "template.window".

```
this.window = new Telia.Widgets.Window(xPos,147, ..., tem-
plate.window);
```

If a template widget is left out of the constructor the widget will be created normally. That way the views are backward compatible.

In turn, when a widget is created, it starts by checking if the template parameter has been set. If that is the case the widget will use the elements in the template widget. If no template is passed the widget will be created normally, achieving backwards compatibility.

Table over template widget types

Template widget	*Description*
NormalWidget	The normal template widget contains a `g` element with a `telia:id` attribute and an finite number of child elements. The child elements may in turn have a `telia:id` attribute, specifying the relationship to the parent node. The available relationships are: `bgNode`, `bgImage`, `textNode`.

ListWidget	The list template widget is based on the NormalWidget. It is specified by setting the `telia:type` attribute to "list". It has additional attributes for specific list widget functionality, such as number of items to show, width and height of the list. The available properties are: `telia:width`, `telia:height`, `telia:items`, `telia:offset`, `telia:transform`
MultiItemListWidget	Based on the list widget, but for lists with predefined list items, where each list item is represented in the template.
HotSpotWidget	The `HotSpotWidget` is based on the `NormalWidget` and is specified by setting the `telia:type` attribute to "hotspot". It provides attributes to specify the navigation path among the HotSpots. The attributes are: `telia:neighbour-left`, `telia:neighbour-right`, `telia:neighbour-top`, `telia:neighbour-bottom`, `telia:transform`

Tabell 5-1 Describes the avaliable template widget types.

5.7.2 Default templates

The system has support for two types of widget templates. The first type is the one described above, where the template widgets are defined in a view template and sent to the widget on creation, thus a specific template intended for a specific widget instance. The second type functions as a generic template for a certain element in the user interface or for a certain widget type. This is called a default template. If a widget is created without the template parameter set, it first checks if there exists a default template for that widget type before it continues to create its elements. The purpose is to allow for customization of elements and widgets that should look the same throughout the user interface, without creating a full view template.

The widgets that have support for default templates are the ListItem and the Scrollbar widgets.

5.8 Results

The TeliaSonera user interface was successfully ported to SVG. The rough port resulted in a user interface that is almost identical to the HTML version. All widgets except for the Table-widget were ported. The difference in appearance lies in the graphics rendering, where Ekioh renders shapes (such as rounded corners) smoother than the Mozilla browser. The template system shows that it is possible to customize the appearance of a user interface. The loading time has improved to around 1/3 and graphics rendering is around 3 times faster. The application uses a

proprietary element of the Ekioh render to support a scrollable text area, which breaks the platform independence requirement.

5.8.1 R1: Performance

The startup time has been reduced from around 45 seconds to around 15 seconds (not counting the 15 seconds it takes for the login procedure) [22]. This corresponds to the results of the JavaScript performance test, shown in Table 4-6, which states that the Ekioh engine is around 3.5 times faster than the Mozilla 1.7 browser.

Table over startup time			
	Total time (s)	*Login time (s)*	*Diff (s)*
SVG (Ekioh)	30	15	15
HTML (Mozilla 1.7)	60	15	45

Table 5-4 Start up time comparison

A simple benchmark that measures the time to render the EPG view was performed. The EPG view is one of the more complex views, which should give a good enough indication of overall graphics performance. The results show that there is a clear improvement in rendering time that corresponds to the results of the graphics rendering test shown in Table 4-7.

Table over time to show the EPG view		
	Time SD (ms)	*Time HD (ms)*
SVG (Ekioh)	372	707
HTML (Mozilla 1.7)	1460	-

Table 5-5 EPG rendering time comparison

5.8.2 R2: Platform independence

The GUI runs well in Ekioh on the Motorola 1910 STB. However, some proprietary Ekioh specific functionality was used to compensate for the missing clipping functionality in SVG Tiny 1.2. The use of that functionality will break the code in other SVG browsers.

[22] The login procedure is handled by the npruntime plug-in and is not affected by the browser or the JavaScript engine.

5.8.3 R3: Resolution independence

All widgets, except for the `Image` widget, are now based on vector graphics. When the resolution is changed the SVG documents viewport is automatically scaled and all SVG elements inside it adapts to the new resolution. The user interface still consists of many bitmap images, which do not scale well. Many of these, such as icons and logotypes, can be translated into vector graphics which will make them resolution independent. Other, such as movie posters, cannot be represented by vector graphics due to the complexity of the graphics. These would have to be stored in different versions (for each resolution) or scaled to the appropriate resolution on the server side.

5.8.4 R4: Skinnability

The template system makes it possible to create different skins for the user interface. The template structure presented here is not necessarily the best possible solution, but one that is well suited for the current design of the user interface.

Figure 5-7 The user interface skinned using the original design.

Figure 5-8 The user interface skinned using an alternative design.

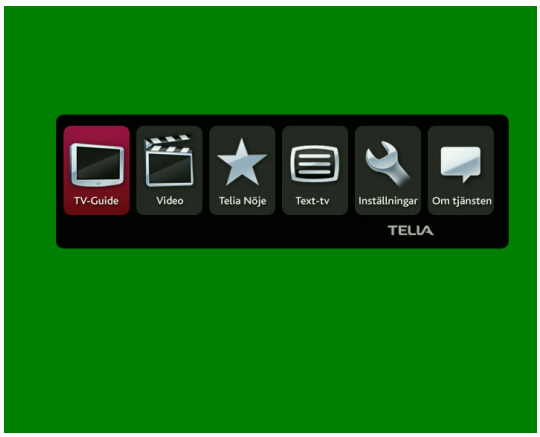

Figure 5-9 The main menu skinned using the original design.

Figure 5-10 The main menu skinned using an alternative design.

5.8.6 R5: Rich graphics

No animations were added to the GUI due to lack of time. Richer graphics were added in form of gradients on list items and backgrounds.

Chapter 6
Discussion and conclusion

This chapter briefly reflects over the method used, shares some thoughts over the future and concludes the work of the thesis.

6.1 Reflections on empirical data

Three of the four interviews, which form the basis for the IPTV services study, were performed with set-top box manufacturers. Because the intention was to create an understanding of how the IPTV set-top box market has evolved and what the future may hold in terms of services, it would have been interesting to have heard the opinions of some chipset manufacturers. Together with set-top manufacturers, application providers and business developers at TeliaSonera it would have covered the complete chain of IPTV service development and perhaps given a more nuanced picture of the situation.

6.2 Reflections on the evaluation method

The Pugh Matrix was chosen as evaluation method much to the fact that is a simple method that allows for high customization of multiple choices and criteria. One could argue that a method like "A process for context-based technology evaluation" [57], which promotes creation of model implementations to validate the criteria, would be a better choice because it will actually test the technology against each criterion. I must agree, but considering the time frame of a master thesis and the limited resources available during the work the decision-matrix method makes sense.

There is, however, an amount of uncertainty in the decision-matrix method; weighing the criteria differently may produce different results. Another uncertainty of this particular evaluation is the scoring of each criterion. Because the comparison is made at quite high level, looking at the overall characteristics of each technology, some scores are highly based on the author's perception of the case. An example is

the “Development support”-criterion, where I give Flash, Silverlight and JavaFX higher scores than HTML with motivation that they all ship with dedicated tools and utilities whereas with HTML and Javascript the developers have to piece together their own. Another writer might have valued the latter higher with the motivation that it allows more freedom and would thus produce a slightly different result.

It may seem that SVG was favored in the study because it was the only technology besides HTML that could be tested on a set-top box. That is unfortunate but I am not sure how I could have avoided the problem and still use a set-top box for the evaluation. As explained in the Introduction chapter it was necessary to perform the tests on the target hardware to get reliable results.

6.3 Thoughts on performance testing

When testing and comparing performance of different technologies, it is actually the runtime environments of the technologies that are compared. It is the Ekioh SVG renderer that renders SVG around 4 times faster than the Mozilla 1.7 browser renders HTML, and it is the Javascript engine in the Ekioh SVG browser that is around 3.5 times faster than the Javascript engine in the Mozilla 1.7 browser. It would therefore be of great interest to redo the performance tests when, or if, a more modern HTML browser becomes available for STBs.

6.4 Open IPTV Forum

The Open IPTV Forum is mentioned throughout this paper but is not included in the study other than its IPTV services and functions for release 2 document. The reason for this is that it is such a big initiative that it would have taken up a too large part of the study. Understanding the extent of the Open IPTV Forum specifications may be subject for a thesis by itself and since it consists of end-to-end solutions it does not actually fall in to the scope of this thesis, which is mostly concerned about the user interface layer. Regardless, the organization has published specifications for its first release and implementations are expected to be released soon [58]. It would of course be of interest to follow up on these implementations.

6.5 Improvements and future work

The prototype developed in Chapter 5 Case TeliaSonera is more or less a straight port from HTML to SVG. The GUI could be optimized for SVG by further analyzing the widgets and optimize them for performance, e.g., by including fewer elements and make extensive use of the SVG `use` element [55](chapter 5.6).

Two IPTV services – 3rd party applications and a widget system – where identified in the IPTV services study, but not used in the evaluation. Effort should be placed to deeper analyze the potential of these services and what impact they would have on the set-top box user interface.

I would recommend keeping an eye on the upcoming SVG 2.0 specification [59]. The specification is not yet published as a working draft, but it is explained that it aims to improve interoperability with SVG and HTML (and other W3C technologies). It is also stated that it will designed to work across platforms and devices, replacing the different SVG profiles that exists today with one. That would eliminate much of the problems with the current specifications. For instance, a notable issue that arose during the prototype development is the lack of clipping functionality in SVG Tiny 1.2, which forced me to use a proprietary element of the Ekioh renderer to create a scrollable text area. SVG Full 1.1 supports clipping but does not specify the `textarea` element – which is used extensively in the user interface. Having only one standard would be of great benefit for a developer of set-top box user interfaces.

From the interviews it appeared that work is being done about bringing 3D graphics into the set-top boxes. Hans Vind argued that the desire of having 3D graphics in the set-top box and the knowledge of what to do with the graphics does not add up. The customers are not sure whether they want the whole user interface to be modeled in 3D or if it is just 3D-like effects they are after. Many of these effects are actually 2D graphics using filter effects and transformations which make them look like 3D – so called 2.5D[23]. Effects such as the cover flow effect Apple uses in iTunes, or some of the Microsoft PowerPoint like transitions can be created using 2D graphics. In the paper "Achieving 3D Effects with SVG" it is explained how such effects can be created [60]. It would be interesting to study what impact pure 3D graphics (not 2.5D) would have on user interfaces for consumer electronics and that use cases they might have. If it is sufficient with those effects that e.g. SVG can provide, is it then worth the effort to use OpenGL, which requires much more effort by the programmer? It is an interesting topic that should be monitored and studied closer when hardware for 3D graphics acceleration starts to appear in the set-top boxes.

The results of the IPTV services study indicate that the set-top box will remain, for some years at least. Interesting would have been to compare it to some of today's high end home media devices, such as the Sony PlayStation 3. Sony released PlayTV in 2008, which is a DVD-T tuner with PVR functionality. Taking the step to IPTV should be possible since the PlayStation 3 is able to play most video for-

[23] http://en.wikipedia.org/wiki/2.5D

mats [61]. Content protection is an area that needs to be examined thoroughly in order to provide encrypted content. The idea is interesting, since IPTV presents the opportunity to bring the television content to basically any device with an Internet connection.

6.6 Conclusion

By choosing SVG as display technology the TeliaSonera IPTV GUI can be improved with better performance, resolution independence and richer graphics. SVG is likely to work well with upcoming IPTV services, such as Internet content and home networking. Two of the proposed services, a widget system and third party applications, where too large to analyze, thus require further study in order to understand what requirements they may put on the user interface technology.

It was a fairly simple task to replace HTML with SVG in the TeliaSonera IPTV GUI because they both share the DOM and support Javascript. The fact that TeliaSonera already had an abstraction layer over the HTML elements facilitated even more.

By choosing SVG, the company must accept to either use some proprietary elements or a mix between two specifications (if it needs support for scrollable areas in the user interface). Neither alternative ideal and can not be expected to work seamless on different SVG renderers. However, because TeliaSonera controls the hardware (the set-top box) their customers use, they also control what software runs on that hardware. That practically takes browser inconsistencies out of the equation.

For a company that is looking to build a rich graphical user interface for a set-top box, SVG should make a good choice as long as the company can accept the limitations described above.

6.7 Current relevance

This thesis was first started on in the spring of 2008. Most of the report was written during that time – the interviews, technology study and evaluation were performed and the prototype was developed, but the report was not finalized. In the spring of 2012 I decided to pick up where I left and quickly realized that some of the informa-

tion was outdated and irrelevant. I have had to update some sources and also revise the technology evaluation chapter. The results have not changed since 2008, though.

If the interviews had been performed today I am sure some other services would have been presented. However, some of the predictions have come true. For instance, a good example is the Samsung smart-tv platform[24]. It consists of a widget engine that runs third party applications. The platform has support for DLNA and can be controlled by applications on mobile devices.

The Android platform is another example of a similar system[25]. An interesting event worth following up on is Google's recent acquisition of Motorola Mobility [62]. Will this bring the Android ecosystem to the set-top box in large scale?

I believe that I would have given HTML5 more focus in the study and evaluation if they had been performed today. HTML5 and CSS3 provide tools for far more expressive graphics than HTML4 and CSS2. That together with the announcement from Adobe in late 2011 that Flash is discontinued for mobile devices [63] can be interpreted as the industry is moving towards open technologies for the web.

TeliaSonera has not yet, for different reasons, moved over to use SVG. One reason is that better hardware in new set-top boxes together with the introduction of the WebKit browser have made the performance gains of SVG smaller than earlier. Another reason is the promise of HTML5 and the large number of devices that supports it.

[24] http://www.samsung.com/se/smarttv/

[25] http://www.android.com

Bibliography

1. **Bell, Judith.** *Introduktion till forsknings-metodik (4th edition ed.)* . s.l. : Narayana Pres, 2006. 9789144046457.

2. **Tague, Nancy R.** Evaluation and Decision-Making Tools. *Quality Toolbox, Second Edition.* 2004, pp. 219-223.

3. **Dr. McDermott, David.** Rational decision making models. [Online] [Cited: January 30, 2009.] http://www.decision-making-confidence.com/rational-decision-making-models.html.

4. **Fülop, Jànos.** *Introduction to Decision Making Methods.* s.l. : Laboratory of Operations Research and Decision Systems, Computer and Automation Institute, Hungarian Academy of Sciences.

5. **Cesar, Pablo.** *A Graphics Software Architecture for High-End In- teractive TV Terminals.* Espoo : Helsinki University of Technology, 2005. TML-A12.

6. **Bosma, Niels.** *STB application development based on modern web technology.* Linköping : Linköping University, 2007. LITH-IDA-EX--07/018--SE.

7. **Vinkvist, Fredrik.** *A feasibility study of building Set-top box user interfaces using Scalable Vector Graphics.* Linköpings : Linköpings universitet, 2008. LITH-ISY-EX--08/4011--SE.

8. SunSpider JavaScript Benchmark. *SunSpider JavaScript Benchmark.* [Online] [Cited: 05 10, 2012.] http://www.webkit.org/perf/sunspider/sunspider.html.

9. **Christmann, Sean.** GUIMark Home. [Online] [Cited: January 30, 2009.] http://www.craftymind.com/guimark.

10. —. GUIMark 2. *CRAFTYMIND.* [Online] [Cited: 05 10, 2012.] http://www.craftymind.com/guimark2/.

11. GUI Definition. *The Linux Information Project.* [Online] October 1, 2004. [Cited: 04 24, 2012.] http://www.linfo.org/gui.html.

12. **Vind, Hans.** *Senior Product Specialist at Motorola.* [interv.] Joakim Svensson. 12 17, 2008.

13. **Pakula, Marcin.** *Director of Technical Marketing at Advanced Digital Broadcast SA.* 1 23, 2009.

14. **Open IPTV Forum.** About us. *Open IPTV Forum.* [Online] Open IPTV Forum. [Cited: 2 16, 2009.] http://www.openiptvforum.org/aboutus.html.

15. —. *Open IPTV Forum – Functional Architecture – V 1.0.* [Document] s.l. : Open IPTV Forum, 2007.

16. **Consumer Electronics Association.** *Web-based Protocol and Framework for Remote User Interface on UPnP™ Networks and the Internet (Web4CE).* [PDF document] s.l. : Consumer Electronics Association, Technology & Standards Department, 2007.

17. **Open IPTV Forum.** *Services and Functions for Release 2.* s.l. : Open IPTV Forum, 2008.

18. **Broadcom.** BCM7405. *Broadcom.* [Online] [Cited: 3 1, 2009.] http://www.broadcom.com/products/Cable/Cable-Set-Top-Box-Solutions/BCM7405.

19. **ST Micro.** STi7105. *ST Micro.* [Online] [Cited: 3 1, 2009.] http://www.st.com/stonline/products/literature/bd/15007/sti7105.pdf.

20. **Intel Corporation.** Intel Introduces First IA System on Chip for Consumer Electronics, Expands Internet to TV Experience. [Online] Intel Corporation, 8 20, 2008. [Cited: 2 18, 2009.] http://www.intel.com/pressroom/archive/releases/20080820comp_a.htm.

21. **W3C.** HTML 4.01 Specification. [Online] 1999. http://www.w3.org/TR/html401.

22. —. Document Object Model (DOM) Level 3 Core Specification. *W3C.* [Online] [Cited: 05 02, 2012.] http://www.w3.org/TR/DOM-Level-3-Core/.

23. —. Cascading Style Sheets (CSS) Snapshot 2010. *W3C.* [Online] [Cited: 05 02, 2012.] http://www.w3.org/TR/css-2010/.

24. **Wikipedia.** Comparison of JavaScript frameworks. *Wikipedia.* [Online] [Cited: 05 02, 2012.] http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks.

25. **W3C.** Media Queries. *W3C.* [Online] [Cited: 05 02, 2012.] http://www.w3.org/TR/css3-mediaqueries/.

26. —. CSS Image Values and Replaced Content Module Level 3. *W3C.* [Online] [Cited: 05 02, 2012.] http://www.w3.org/TR/css3-images/.

27. —. CSS Transitions. *W3C.* [Online] [Cited: 05 02, 2012.] http://www.w3.org/TR/css3-transitions/.

28. —. The WebSocket API. *W3C.* [Online] [Cited: 05 02, 2012.] http://www.w3.org/TR/websockets/.

29. —. File API: Directories and System. *W3C.* [Online] [Cited: 05 02, 2012.] http://www.w3.org/TR/file-system-api/.

30. **Khronos Group.** WebGL Specification. *WebGL - OpenGL ES 2.0 for the Web.* [Online] [Cited: 05 02, 2012.] https://www.khronos.org/registry/webgl/specs/1.0/.

31. **Deveria, Alexis.** Caniuse. *Caniuse.* [Online] [Cited: 05 02, 2012.] http://caniuse.com/.

32. **Web Hypertext Application Technology Working Group.** HTML 5 Draft recommendation. [Online] [Cited: January 30, 2009.] ttp://www.whatwg.org/specs/web-apps/current-work.

33. **Lehni, Juerg and Puckey, Jonathan.** About. *paper.js.* [Online] [Cited: 05 02, 2012.] http://paperjs.org/about/.

34. **W3C.** http://www.w3.org/Graphics/SVG/. *w3c.* [Online] [Cited: 05 01, 2012.] http://www.w3.org/Graphics/SVG/About.html.

35. **Schiller, Jeff.** Codedread - SVG support. *Codedread.* [Online] [Cited: 05 01, 2012.] http://www.codedread.com/svg-support.php.

36. **Oracle Corporation.** JavaFX - The Rich Client Platform. *JavaFX.* [Online] [Cited: 04 30, 2012.] http://www.oracle.com/technetwork/java/javafx/overview/index.html.

37. —. What Is JavaFX? *Oracle.* [Online] [Cited: 05 03, 2012.] http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm.

38. —. JavaFX 2.1 API. *JavaFX 2.1 API.* [Online] [Cited: 04 30, 2012.] http://docs.oracle.com/javafx/2/api/index.html.

39. —. Class Node. *JavaFX 2.1 API.* [Online] [Cited: 04 30, 2012.] http://docs.oracle.com/javafx/2/api/javafx/scene/Node.html#isResizable().

40. —. Introduction to FXML. *JavaFX 2 Documentation.* [Online] [Cited: 04 30, 2012.] http://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html#overview.

41. —. JavaFX CSS Reference Guide. *JavaFX 2 Documentation.* [Online] [Cited: 04 30, 2012.] http://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html.

42. —. JavaFX Roadmap. *Oracle Technology Network.* [Online] [Cited: 04 30, 2012.] http://www.oracle.com/technetwork/java/javafx/overview/roadmap-1446331.html.

43. **Adobe Systems Incorporated.** *SWF File Format Specification, Version 10.* [PDF] s.l. : Adobe Systems Incorporated, 2008.

44. —. SWF Technology Center. *Adobe Developer Connection.* [Online] [Cited: 04 30, 2012.] http://www.adobe.com/devnet/swf.html.

45. **Microsoft.** Silverlight Overview. *MSDN Library.* [Online] [Cited: 05 01, 2012.] http://msdn.microsoft.com/en-us/library/bb404700(v=vs.95).aspx.

46. —. WPF Graphics Rendering Overview. *MSDN Library.* [Online] [Cited: 05 01, 2012.] http://msdn.microsoft.com/en-us/library/ms748373.aspx.

47. —. Application and Programming Models. *MSDN Library.* [Online] [Cited: 05 01, 2012.] http://msdn.microsoft.com/en-us/library/cc903934(v=vs.95).aspx.

48. —. Get Microsoft Silverlight. *Microsoft.com.* [Online] [Cited: 05 01, 2012.] http://www.microsoft.com/getsilverlight/Get-Started/Install/Default.aspx.

49. MoonlightRoadmap. *mono-project.* [Online] [Cited: 05 01, 2012.] http://www.mono-project.com/MoonlightRoadmap.

50. **Yahoo.** *Yahoo! Connected TV.* [Online] [Cited: 3 1, 2009.] http://connectedtv.yahoo.com/.

51. **Intel Corporation.** Intel and Yahoo! to Bring the Internet to Television. *Intel press room.* [Online] 8 20, 2008. [Cited: 3 1, 2009.] http://www.intel.com/pressroom/archive/releases/20080820comp_b.htm.

52. **Ekioh Ltd.** *Ekioh SVG Extensions.* [Document] September 8, 2008.

53. **Dr. Dailey, David P.** A comparative analysis of some considerations for browser performance in SVG. [Online] [Cited: February 1, 2009.] http://www.svgopen.org/2007/papers/BrowserPerformanceMeasures/index.html.

54. **W3C.** Mobile SVG Profiles: SVG Tiny and SVG Basic. [Online] January 14, 2008. [Cited: February 8, 2009.] http://www.w3.org/TR/SVGMobile/.

55. —. Scalable Vector Graphics (SVG) Tiny 1.2 Specification. [Online] December 22, 2008. [Cited: January 30, 2009.] http://www.w3.org/TR/SVGMobile12.

56. **Prosser, Joe.** SVG: A Key Element in Achieving Product Differentiation and Competitive Advantage in the DVB Market. *SVG Open 2008.* [Online] August 2008. [Cited: 03 26, 2009.] http://svgopen.com/2008/papers/75-SVG_a_key_element_in_achieving_product_differentiation___competitive_advantage_in_the_DVB_market/.

57. **Lewis, Grace A and Wrage, Lutz.** *A Process for Context-Based Technology, Evaluation.* s.l. : Carnegie Mellon University, 2005.

58. **Open IPTV Forum.** Way now open for Widespread Use of IPTV - New Specifications Published. *Open IPTV Forum.* [Online] Open IPTV Forum, 01 08, 2009. [Cited: 03 01, 2009.] http://www.openiptvforum.org/PRESS/pressrelease_070109.html.

59. **W3C.** SVG 2.0 - Introduction. *SVG 2.0.* [Online] [Cited: 05 15, 2012.] http://dev.w3.org/SVG/profiles/2.0/publish/intro.html.

60. **Fujisawa, Dr. Jun and Grasso, Mr. Anthony.** Achieving 3D Effects with SVG. *SVG Open 2008.* [Online] 08 2008. [Cited: 03 01, 2009.] http://www.svgopen.org/2008/papers/86-Achieving_3D_Effects_with_SVG.

61. **Sony Computer Entertainment Inc.** PlayStation®3 System Software User's guide. *PlayStation®3 System Software User's guide.* [Online] [Cited: 05 15, 2012.] http://manuals.playstation.net/document/en/ps3/current/video/filetypes.html.

62. **Google.** Google to Acquire Motorola Mobility. *Google Investor Relations.* [Online] [Cited: 05 15, 2012.] http://investor.google.com/releases/2011/0815.html.

63. **Adobe Systems Incorporated.** Flash to Focus on PC Browsing and Mobile Apps; Adobe to More Aggressively Contribute to HTML5. *Adobe Featured Blogs.* [Online] 11 09, 2011. [Cited: 04 30, 2012.] http://blogs.adobe.com/conversations/2011/11/flash-focus.html.

64. **Sigma Design.** Sigma Designs brings Adobe Flash platform to consumer electronics in the digital home. *Sigma Designs.* [Online] 1 6, 2009. [Cited: 3 1, 2009.] http://sigmadesigns.com/public/Company/press_releases/090106b.pdf.

65. **Broadcom.** Adobe and Broadcom Bring the Adobe Flash Platform to TVs. *Broadcom.* [Online] 1 6, 2009. [Cited: 3 1, 2009.] http://www.broadcom.com/press/release.php?id=s357665&industry_id=4.

66. **Intel Corporation.** Intel and Adobe to Extend Flash Platform to TVs. *Intel Press Room.* [Online] 1 5, 2009. [Cited: 3 1, 2009.] http://www.intel.com/pressroom/archive/releases/20090105corp.htm.

Appendix A - SVG template file over the main menu view

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg"
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:ekioh="http://www.ekioh.com/2007/ekioh"
   xmlns:telia="http://iptvlogin.telia.se/2008/telia"
   viewport-fill="green">

<g id="MainNavigatorTemplate">

<g telia:id="navigatorBox" transform="matrix(1 0 0 1 0 0)">
<rect telia:id="navigatorBox:bgNode" width="241" height="576" rx="0" ry="0" opacity="0.8"/>

<g telia:id="hotspotx1y0" transform="matrix(1 0 0 1 0 50)">
    <rect telia:id="hotspotx1y0:bgNode" stroke="#333333" stroke-width="2"
        fill="url(#unselectedColor)" class="mainNavigatorHotspot" width="240" height="40" rx="0" ry="0"/>
    <textArea font-size="20" fill="white" x="55" height="35" display-align="center">TV-guide</textArea>
    <image telia:id="hotspotx1y0:bgImage" x="5" y="3" xlink:href="images/icons/tv_small2.png"/>
</g>
<g telia:id="hotspotx2y0" transform="matrix(1 0 0 1 0 95)">
    <rect telia:id="hotspotx2y0:bgNode"  stroke="#333333" stroke-width="2"
        fill="url(#unselectedColor)" class="mainNavigatorHotspot" width="240" height="40" rx="0" ry="0"/>
    <textArea font-size="20" fill="white"  x="55" height="35" display-align="center">Filmer</textArea>
    <image telia:id="hotspotx2y0:bgImage" x="5" y="3" xlink:href="images/icons/movies_small.png"/>
</g>
<g telia:id="hotspotx3y0" transform="matrix(1 0 0 1 0 140)">
    <rect telia:id="hotspotx3y0:bgNode"  stroke="#333333" stroke-width="2"
        fill="url(#unselectedColor)" class="mainNavigatorHotspot" width="240" height="40" rx="0" ry="0"/>
    <textArea font-size="20" fill="white"  x="55" height="35" display-align="center">Nöje</textArea>
    <image telia:id="hotspotx3y0:bgImage" x="5" y="3" xlink:href="images/icons/star_small.png"/>
</g>
<g telia:id="hotspotx4y0" transform="matrix(1 0 0 1 0 185)">
    <rect telia:id="hotspotx4y0:bgNode" stroke="#333333" stroke-width="2"
        fill="url(#unselectedColor)" class="mainNavigatorHotspot" width="240" height="40" rx="0" ry="0"/>
    <textArea font-size="20" fill="white"  x="55" height="35" display-align="center">Text-TV</textArea>
    <image telia:id="hotspotx4y0:bgImage" x="5" y="3" xlink:href="images/icons/texttv_small.png"/>
</g>
<g telia:id="hotspotx5y0" transform="matrix(1 0 0 1 0 230)">
    <rect telia:id="hotspotx5y0:bgNode" stroke="#333333" stroke-width="2"
        fill="url(#unselectedColor)" class="mainNavigatorHotspot" width="240" height="40" rx="0" ry="0"/>
    <textArea font-size="20" fill="white"  x="55" height="35" display-align="center">Inställningar</textArea>
    <image telia:id="hotspotx5y0:bgImage" x="5" y="3" xlink:href="images/icons/preferences_small.png"/>
</g>
<g telia:id="hotspotx6y0" transform="matrix(1 0 0 1 0 275)">
    <rect telia:id="hotspotx6y0:bgNode"  stroke="#333333" stroke-width="2"
        fill="url(#unselectedColor)" class="mainNavigatorHotspot" width="240" height="40" rx="0" ry="0"/>
    <textArea font-size="20" fill="white" x="55" height="35" display-align="center">Om tjänsten</textArea>
    <image telia:id="hotspotx6y0:bgImage" x="5" y="3" xlink:href="images/icons/about_small.png"/>
</g>
<image telia:id="logo" x="10" y="5" width="64" height="23" xlink:href="images/swe/company_icon.png"/>

</g>

<g telia:id="campaign" transform="matrix(1 0 0 1 0 375)">
    <g telia:id="hotspotx1y1" transform="matrix(1 0 0 1 0 0)">
        <rect telia:id="hotspotx1y1:bgNode" width="240" height="100" fill="#cccccc" rx="0" ry="0"/>
        <image telia:id="hotspotx1y1:bgImage" y="10" stroke="#cccccc" stroke-width="2" xlink:href=""/>
    </g>
</g>

</g>
</svg>
```

TRITA-CSC-E 2013:002
ISRN-KTH/CSC/E--13/002-SE
ISSN-1653-5715

www.kth.se