Cross-Platform Multimedia Contents through Model Transformations: The Digital TV Case

Aitor Rodríguez Alsina

Ph.D. Thesis Dissertation

Directed by

Jordi Carrabina Bordoll

Bellaterra, July 2012

PhD of the Computer Science Program of the Universitat Autònoma de Barcelona

Engineering School. Department of Microelectronics and Electronic Systems

Universitat Autònoma de Barcelona

Jordi Carrabina, lecturer in the Microelectronics and Electronic Systems Department of the “Universitat Autònoma de Barcelona”,

CERTIFIES:
That the present thesis has been realized under his direction by Aitor Rodríguez Alsina in partial fullment of the requirements of the Ph.D. degree on Computer Science by the “Universitat Autònoma de Barcelona”.

Bellaterra, July 2012.

To Oli, half of this work is yours

To my mother and brother

To Vito, a huge friend in a small format

ACKNOWLEDGEMENTS

This thesis, as well as all the work that is behind, would not been possible without the help and support of many people that with their knowledge, complicity and patience have contributed to the development of this research.

As a wise said in a famous speech, one can not connect the dots of life by looking forward; one can only connect them looking backwards and realize the actual sequence that has brought to one to a particular moment. When applying this statement to the present work, the hard moments that always come with any success are blurred in my mind with the sweet to form a unified feel of satisfaction by the end of a personal commitment.

I am sincerely grateful to Prof. Jordi Carrabina for his support and directions along this time, but especially for give me the opportunity to work in a leading research center that has been the cornerstone of my learning during the past 7 years. I have honestly to thank to him for many of the connected dots that bring me where I am today. This includes reached milestones, training, travels, meetings, projects, discussions, laughs, annoyances, joys, and those aspects that day by day help to forge a working and personal relationship. His help has been significant in the last stage of this dissertation, when his motivation and reviews have been determinant to reach the final at the planned time.

In this last development stage of this research, I've also received an incomparable support from Prof. Pilar Orero, self-styled the “mom of PhD students of CAIAC”, and Dr. Guillermo Talavera, one of the friends who has given me orientation from the point of view of somebody that has recently given my steps before.

It is difficult to make a complete list of all people that I would like to thank by its daily friendship and collaboration in the work place. I’ll try to form a non-complete list of them: Màrius Monton, Borja Martínez, Roger Puig, Joan Carles Chak, Héctor Delgado, Marc Moreno, Eduard Céspedes, Eric Teruel, Carlos Montero, Joan Garcia, Anna Vilaró, Sofia Arqués and Marta Viana among others. Please try to imagine that I have written some personal and sentimental lines for each of you and accept a huge hug from me.

On the personal side, the list is still bigger and I'm very glad for having the friends who still put up with me. At the risk of losing some of them for do not appearing here, I will try to give a non comprehensive list: Gerard, Toni, Raquel, Quim, Natalia, Montse, Cristian, Arantxa, Santi, Miki, Assun, Miquel, Olga, Jordi, … <put your name here>.

I would like to add also a special line to thank the short but helpful experience of the course University2Business (organized by the UAB) that I recently had the opportunity to share with really great people. They have taught me that team working is more than sharing knowledge to reach a milestone.

The more special grateful is definitely to my wife, Oli, for who I neither have enough space to thank her for being as she is, nor enough poetic writing style to address the words she deserve. Luckily, she knows what I mean.

Another special place is for my mother, who is the main responsible for being who I am today and who actually deserves my most sincerely grateful by his unconditional love, help and patience. I would like to mention my brother, whom I definitely appreciate despite our ups and downs.

I feel also a sincerely grateful to the family of my wife, who has accepted me happily from the first day. Please, accept my apologies for being disconnected the last months, but I have been writing the book you have in hands now.

The last, but not the least, friend that I want to remember here is to my little mate Vito. Sometimes, the connection that can be achieved between people and animals overcomes any expectation, especially when talking about a proud Jack Russell. His life has been attached to the present dissertation, time in which I cannot enumerate the number of laughs and good moments that he selflessly gave to me. He was born in Setember 2008, when I started this research, and died because a suddenly and serious illness while I was writing the last lines of this work. He will never read these lines of gratitude, but I hope that this small tribute serve to remind him beyond my personal memory.

ABSTRACT

The current diversity of internet-connected consumer devices enables an increasing variety of multimedia platforms for accessing audiovisual content, interactive services, games and all kind of user applications. The concept of digital TV is evolving from being an isolated unidirectional technology to become part of the ecosystem of services users consume in their multimedia home or nomadic environment. In this context, the convergence between interactive TV, Internet, and applications is a reality thanks to the family of smart devices that are available on the market. Smart TVs, smartphones, tablets, and game consoles allow users to access TV contents and interactive applications through multiple networks.

The number of different application formats and runtime environments that currently enable interactive TV services hinders the portability of applications developed for those environments. The lack of a globally unified and accepted standard for the deployment of interactive contents on TV platforms requires the use of special techniques to adapt those contents from one platform to another. Interactive content producers can save both time and resources when developing the same content for different platforms.

The main objective of this dissertation is the proposal and validation of a suitable methodology for the efficient generation and maintenance of cross-platform iTV applications, which allows the separation between the design process and its multiple implementations for different platforms. This thesis analyzes the current context of interactive TV and proposes a solution for generating "write once, adapt to anywhere" applications based on a portable content format for TV environments. The proposed methodology is validated through different application use cases that have been tested using the software framework developed in the course of this research. This includes the required modules for translating cross-platform descriptions into platform-specific iTV applications and an integrated development environment containing a visual editor for graphic user interfaces that stores the interface description in the portable content format.

The study of multimedia content synchronization in web-based platforms generated a secondary contribution for the development of subtitling systems based on HTML5. This proposal takes advantage of SVG and SMIL for synchronizing customizable video subtitles across web platforms. It also enables to reduce considerably the code required by an application for managing time issues. The results of the user experience evaluation for the proposed subtitling system show that SMIL time features allow an efficient management of subtitles across different HTML5 platforms without losing the synchronization between the presentation components.

“Any sufficiently advanced technology is indistinguishable from magic.”

Arthur C. Clarke

CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT iii

CONTENTS vii

INDEX OF FIGURES xi

INDEX OF TABLES xiii

ACRONYMS xiv

CHAPTER 1. INTRODUCTION 1

1.1. The Addressed Problem: Cross-Platform Applications for Interactive TV 2

1.2. Objectives 2

1.3. Methodology 3

1.4. Contributions 3

1.5. Structure of the dissertation 4

CHAPTER 2. BACKGROUND CONTEXT, STATE OF THE ART AND RELATED TECHNOLOGIES 7

2.1. Background Context 7

2.1.1. Interactive TV 7

2.1.2. Platform, Cross-Platform, Platform-Agnostic and Other Trends 11

2.1.3. Runtime environments 12

2.1.4. Progressive Enhancement 13

2.1.5. User experience 13

2.2. State of the Art 14

2.2.1. Current iTV Platforms and Middleware 14

2.2.2. Cross-Platform Content Description Formats for iTV 20

2.2.3. The Web as a Platform for TV Services 22

2.3. Reference Format for Content Authoring: DVB Portable Content Format 24

2.3.1. Introduction to the Content Format 24

2.3.2. Component Architecture 26

2.3.3. Behavior and User Interaction Capabilities 27

2.3.4. Structuring a Service Description 27

2.3.5. Managing Differences between Target Platforms 28

2.3.6. Media Synchronization 29

2.3.7. The Return Channel 29
2.3.8. Comparison with Other Languages: Justification of DVB-PCF 30
2.4. A Procedural Client Format: MHP 30
2.4.1. Introduction to the Multimedia Home Platform 31
2.4.2. Applications and Services 32
2.4.3. Platform Profiles 34
2.4.4. Deployment Status 34
2.5. A Declarative Client Format: HTML5 and New Cross-Platform Web Technologies 35
2.5.1. Introduction to HTML5 as a Set of Standards 35
2.5.2. Why HTML5? 36
2.5.3. Scalable Vector Graphics 39
2.5.4. Synchronized Multimedia Integration Language 40
2.5.5. Analysis of the Current Browser Support 42
2.6. Summary of the Chapter 44
CHAPTER 3. GENERATING CROSS-PLATFORM APPLICATIONS FOR INTERACTIVE TV 45
3.1. Methodology for Building Cross-Platform iTV Applications 45
3.1.1. Methodology Description 46
3.1.2. System Architecture for Production 50
3.1.3. Remarks 53
3.2. Framework for Creating and Managing Cross-Platform Applications 54
3.2.1. Overall Framework Architecture 54
3.2.2. Document Validation and Parsing: PCFValidator 55
3.2.3. The object Model: PCFModel 56
3.2.4. Adopting a Model-View-Controller Strategy: PCFSplitter 58
3.2.5. Scaling the application for Multiple Views: PCFScaler 58
3.2.6. The Document Viewer and Debugger: PCFViewer 59
3.2.7. From Production to the Client Platform: PCFTranscoder 60
3.2.8. Integrated Authoring Tool for Building iTV Applications: VisualPCF 61
3.2.9. DVB-PCF Examples 63
3.2.10. Remarks 63
3.3. Approach for Synchronizing Multimedia Subtitles across Web Platforms 64
3.3.1. Background 64
3.3.2. Approach Description 65
3.3.3. Analysis of the Current Platform Support 69
3.3.4. Remarks 72
3.4. Summary of the Chapter 72
CHAPTER 4. EXPERIMENTS AND VALIDATION 75
4.1. Experiments Driven for Validating the Proposed Methodology 75

4.1.1. Use Case description: Virtual Campus for iTV 76
4.1.2. Transcoding DVB-PCF for Declarative Runtime Environments: HTML5 76
4.1.3. Transcoding DVB-PCF for Procedural Runtime Environments: MHP 81
4.1.4. Results & Partial Conclusions 89
4.2. Validation of the Proposed Development Environment 89
4.2.1. Use case description: Personal Health Record Management for iTV 89
4.2.2. Development of a cross-platform iTV application: VisualPCF 90
4.2.3. Results & Partial Conclusions 94
4.3. Validation of the Approach for Synchronizing Subtitles Across Web Platforms 95
4.3.1. The SVG Subtitle Access Bar 95
4.3.2. Positioning Subtitles on Interactive TV 97
4.3.3. Presenting Subtitles on a Second Screen 98
4.3.4. User Experience Evaluation 100
4.3.5. Results & Partial Conclusions 102
4.4. Summary of the Chapter 102
CHAPTER 5. CONCLUSIONS 105
5.1. General Conclusions 105
5.2. Future Research 107
APPENDIX A. AUTHOR'S BIOGRAFY 113
APPENDIX B. AUTHOR'S KEY PUBLICATIONS 115
REFERENCES 119

INDEX OF FIGURES

Figure 1. Relationship between HbbTV and other existing standards. 16
Figure 2. Scope of the Portable Content Format (DVB-PCF).. 25
Figure 3. Code example with the basic DVB-PCF component structure................................ 26
Figure 4. Proposed document structure for defining DVB-PCF services. 28
Figure 5. The GEM/MHP overall architecture. 32
Figure 6. Map of iTV application formats and runtime environments.................................. 46
Figure 7. Mapping models for the translation between application formats. 47
Figure 8. Scenarios for production and deployment of cross-platform iTV services............. 49
Figure 9. Overall System and Sub-System Architecture for cross-platform iTV production. .. 51
Figure 10. System and subsystem architecture of the proposed software framework.......... 55
Figure 11. An example of the component hierarchy in DVB-PCF and PCFModel. 57
Figure 12. Relationship diagram between the structural items of DVB-PCF and PCF Model.. 57
Figure 13. PCFViewer 1.0 presenting a sample iTV application described in DVB-PCF........... 60
Figure 14. Use case diagram for VisualPCF. 62
Figure 15. Screenshot of VisualPCF editing a sample iTV application.................................... 63
Figure 16. Example of subtitles defined using the SubRip (.SRT) format. 65
Figure 17. Example of SVG text element with SMIL animation for presenting a subtitle....... 66
Figure 18. JavaScript code for linking the HTML5 video element and the SMIL container..... 67
Figure 19. Time-based strip diagram with two video pause operations................................ 67
Figure 20. Time-based strip diagram with two video seek operations. 68
Figure 21. Time-based strip diagram with a subtitle language change operation.................. 68
Figure 22. Time-based strip diagram with a screen export operation. 69
Figure 23. General container of a HTML5 transcoded application. 77
Figure 24. Example of the HTML5 application obtained from the DVB-PCF transcoding. 81
Figure 25. General container of a MHP transcoded application.. 82
Figure 26. Example of the MHP application obtained from the DVB-PCF transcoding........... 87
Figure 27. Mobile web applications obtained from DVB-PCF descriptions............................ 88
Figure 28. Architecture of the eHealth system developed for TV.. 89
Figure 29. Result of an iTV application design with VisualPCF.. 93

Figure 30. Browser validation of the application generated with VisualPCF......................... 93
Figure 31. The eHealth application presented on TV. 94
Figure 32. The SVG Subtitle Access Bar presented on a desktop browser. 96
Figure 33. Positioning of subtitles within an interactive TV environment. 98
Figure 34. Example of second screen application for subtitles on an Android 2.2 platform. . 99
Figure 35. Results of the user experience evaluation for second screen systems............... 101

INDEX OF TABLES

Table 1. Generic classification of interactive services for MHP platforms. 33
Table 2. SVG animation for an oscillating ellipse in SMIL and JavaScript. 41
Table 3. Browser support for the key features of HTML5. 43
Table 4. Browser support for the key features of SVG in HTML5. 44
Table 5. Support of the proposed solution in main desktop web browsers........................... 70
Table 6. Support of the proposed solution in main mobile web browsers. 71
Table 7. DVB-PCF component clustering for generating HTML5 code. 78
Table 8. DVB-PCF component clustering for generating MHP code. 83
Table 9. Key events that may be generated by an MHP receiver. 84

ACRONYMS

3GPP:	3rd Generation Partnership Project
ACAP:	Advanced Common Application Platform
AJAX:	Asynchronous JavaScript And XML
API:	Application Programming Interface
ARIB:	Association of Radio Industries and Businesses
ATIS:	Alliance for Telecommunications Industry solutions
AWT:	Abstract Window Toolkit
CEA:	Consumer Electronics Association
CE-HTML:	Consumer Electronics HTML
CSS:	Cascading Style Sheets
DAE:	Declarative Application Environment
DFXP:	Distribution Format eXchange Profile (related to TT-AF)
DLNA:	Digital Living Network Alliance
DOM:	Document Object Model
DRM:	Digital Rights Management
DSM-CC:	Digital Storage Media Command and Control
DVB:	Digital Video Broadcasting
DVB-PCF:	DVB Portable Content Format
DVR:	Digital Video Recorder
EPG:	Electronic Program Guide
ETSI:	European Telecommunications Standards Institute
GEM:	Globally Executable MHP
GIF:	Graphics Interchange Format
GPL:	GNU General Public License
HAVi:	Home Audio Video Interoperability
HbbTV:	Hybrid Broadcast Broadband TV
HD/SD:	High Definition / Standard Definition
HTML:	HyperText Markup Language
HTTP:	Hypertext Transfer Protocol
IEC:	International Electrotechnical Commission
IP:	Internet Protocol
IPTV:	Internet Protocol Television
ISDB:	Integrated Services Digital Broadcasting

ISO:	International Organization for Standardization
iTV:	Interactive Television
JPEG:	Joint Photographic Experts Group
MHEG:	Multimedia and Hypermedia information coding Expert Group
MHP:	Multimedia Home Platform
MIME:	Multipurpose Internet Mail Extensions
MPEG:	Moving Picture Experts Group
NCL:	Nested Content Language
NPT:	DSM-CC Normal Play Time
OASIS:	Advancement of Structured Information Standards
OCAP:	Open Cable Applications Platform
OIPF:	Open IPTV Forum
PAE:	Procedural Application Environment
PNG:	Portable Network Graphics
RSS:	Really Simple Syndication
SDK:	Software Development Kit
SMIL:	Synchronized Multimedia Integration Language
SMS:	Short Message Service
STB:	Set-Top Box
SVG:	Scalable Vector Graphics
TT-AF:	Timed Text Authoring Format
UIML:	User Interface Markup Language
UML:	Unified Modeling Language
URI:	Uniform Resource Identifier
URN:	Uniform Resource Name
VOD:	Video On Demand
W3C:	World Wide Web Consortium
WTVML:	Worldwide TV Mark-up Language
XHTML:	eXtensible HTML
XML:	eXtensible Markup Language
XMPP:	Extensible Messaging and Presence Protocol
XSLT:	Extensible Stylesheet Language Transformations

CHAPTER 1

INTRODUCTION

"If a man does not keep pace with his companions, perhaps it is because he hears a different drummer. Let him step to the music which he hears, however measured or far away."

Henry David Thoreau

Interactive Television (iTV) claims for new techniques for the development of multi-platform services and applications. On the one hand, 2012 seems to be the year for the takeoff of the connected TV systems (i.e. Smart TV or Hybrid TV), which promises a significant improvement in the way users consume enhanced services on TV. On the other hand, the rise and variety of connected devices for accessing to TV contents has generated a welter of application formats and runtime environments that hinders the portability of the generated applications.

While the industry does not reach a consensus on a unified and standardized format for the deployment of multimedia applications, some techniques and tools are necessary to adapt the interactive contents from one platform to another. With this kind of tools, interactive content producers can save both time and resources when developing the same content for different platforms, which usually requires virtually cut off production lines. This same problem is happening on the mobile context with quite similar results, where application producers suffer the consequences of a market split between big proprietary silos and the open web standards. Nobody can forecast accurately the result of the current application platform war, especially when these big actors do not present a clear commitment between open web standards and their own silo, keeping an eye on each side.

HTML5 has entered recently in this changing context providing a real alternative to the widespread of large variety of formats and runtime environments that are available for implementing iTV solutions. The rise of consumer Smart TVs and the emergence of new TV platforms as GoogleTV provide new ways of consuming iTV that allow full internet access and HTML5 applications. These platforms are added to the current scenario of consumer Set-top Boxes (STB) and Media Centers that also support HTML5 applications and websites but, however, have not reached a great market penetration because of a lack of a clear and attractive built-in solution. The main difference between HTML5 and the rest of iTV application formats is that HTML5 has been (and is being) designed for all type of devices and platforms, not specifically for TV, which means that it must provide the necessary resources to enable an automatic adaption according to the target device constraints. These capabilities make HTML5 an interesting application format for implementing multi-platform solutions.

When different application formats are not compatible between themselves, the number of conversions required to translate one application format to another increases exponentially with the number of formats requiring the implementation of $n(n-1)/2$ conversions when considering n different formats. A different, and possibly better approach, is to translate each application format into a new one that satisfies the requirements of the rest and reduces drastically the number of conversions. In this model, the number of conversions increases linearly with the number of target formats, which seems more convenient according to the amount and variety of application formats considered in the iTV context. One of the formats designed to fit this model was published in 2006 by the Digital Video Broadcasting (DVB) consortium with the name Portable Content Format (DVB-PCF).

The DVB-PCF provides a standard format for the description of an author's intended viewer experience of an interactive service, which can be translated as required for each target platform. After its publication, the viability of this standard was successfully demonstrated by the British Broadcasting Corporation Research and Development (BBC R&D) group and its automatic translation from DVB-PCF to MHEG-5, which is the main iTV application format in UK. However, the technical details of their implementation were not published and, at the time of writing this work, no further studies have provided a complete workflow for managing and translating DVB-PCF services.

1.1.The Addressed Problem: Cross-Platform Applications for Interactive TV

The problem addressed by this work is the lack of effective methodologies and tools for building and managing cross-platform iTV applications. Today's iTV platforms offer a wide variety of interactive services but different iTV platforms use different technologies for deploying interactive services hence a large amount of interactive content has to be developed bespoke for each platform. This is time consuming and results in high production costs that hinder the development of high-profile interactive applications and the proposal of clear business models. Furthermore, the variety and features of consumer devices that can potentially access to these iTV services argues for a production solution that takes into account the particularities of the target devices instead of the rise of complexity of the device software platforms.

1.2.Objectives

The main objective of this work is the proposal and validation of a suitable methodology for efficiently generating and maintaining cross-platform iTV applications, allowing to separate the design process to its multiple implementations in different platforms, according to their corresponding resources, display capabilities and human-computer interface constraints.

Some specific objectives can be detached from the above mentioned main objective:

- To propose the reference system architecture to apply the proposed methodology.
- To propose a suitable framework for building and managing cross-platforms iTV applications through the proposed system architecture. This set of tools should provide enough flexibility to be included in existing iTV production systems.
- To propose an integrated development environment as a monolithic solution for the management of cross-platform iTV services. This desktop application should include the proposed framework and provide comprehensive facilities to service developers for building and maintaining interactive services for iTV.
- To provide a set of experiments for validating the proposed solutions in different platforms.

1.3.Methodology

All the stages of the developed research have been supported by a permanent revision through scientific publications, scientific communications and research projects that have been validating the theoretical basis of this research, as well as the proposed implementations. Furthermore, the driven research has been closely followed and validated by the PhD Studies Committee of the Department of Microelectronics and Electronics Systems of the Universitat Autònoma de Barcelona, Spain.

1.4.Contributions

This section summarizes the major contributions resulting from the driven research. Section 5.3 of this document provides a deeper look inside the publications and communications related to the presented contributions.

- **Approach for building and maintaining cross-platform iTV services**: The developed research has provided a suitable methodology for the production of interactive services, including interactive applications, based on the standard DVB-PCF for the cross-platform definition of the structure, behavior and contents. The system architecture and workflow are also detailed in order to provide a clear solution for producing cross-platform iTV applications.
- **Proposed tools for implementing the previous approach**: In order to demonstrate the theoretical basis of the driven research, a set of tools has been developed as a framework for building and maintaining cross-platform iTV services. Firstly, a collection of small pieces of software have been built for solving specific issues in the management of services based on the standard DVB-PCF. They have been distributed as open source and can be used freely to build custom iTV production systems. Moreover, an integrated development environment has been developed as a unified solution for building cross-platform iTV services. This desktop environment embodies the proposed software framework and enables the service production through the

proposed methodology but its development still requires some efforts to make it installable on a real production system.

- **Approach for synchronize multimedia subtitles across web platforms**: Some interactive applications were developed during the experimentation and validation of the proposed methodology for building cross-platform iTV services. The design of one of them, regarding the real time synchronization between different multimedia contents, has generated an asset by itself. The video and subtitle synchronization on web based client devices through SVG and SMIL has generated an interesting and partially new way of presenting video subtitles on web platforms.

1.5.Structure of the dissertation

The rest of chapters of the present dissertation outline the different tasks carried out in this research work, ranging from the early information harvesting to the final conclusions:

- **Chapter 2. Background context, state of the art, and related technologies**: The second chapter introduces some theoretical concepts that are needed for a suitable understanding of this research. Moreover, the chapter analyses the state of the art for the addressed problem and provides a further introduction to the main technologies related to the present dissertation.
- **Chapter 3. Proposals and contributions in generating cross-platform applications for interactive TV**: This chapter concentrates on the work developed during the research work that is presented in this dissertation. Here, detailed explanations and remarks can be found about the proposed methodology for building cross-platform iTV Applications, the generated software framework for implementing this methodology and the integrated development environment generated for validating the proposed software framework. Moreover, a novel proposal for presenting video subtitles on web platforms is presented as an implementation example of multimedia content synchronization.
- **Chapter 4. Experiments and validation**: This chapter presents the experiments driven for validating the presented proposals and discusses the provided results. Firstly, the proposed methodology is evaluated through the study of different use cases for ensuring its viability, flexibility and effectiveness. Secondly, the developed software framework is validated in the generation of an eHealth application for a national research project[1]. Thereafter, the usability and capabilities of the proposed integrated development environment are validated through subjective evaluation tests in order to ensure a reasonable effectiveness of the tool, especially when used by non-experts. Finally, the proposed approach for synchronizing video subtitles on web platforms is implemented as a web application and is tested by users validating the synchronism correctness.

[1] "La eSalud al servicio del ciudadano". AVANZA TSI-020302-2009-85. Spain, 2011.

- **Chapter 5. Conclusions**: The final chapter summarizes the achieved results and exposes the conclusions of this dissertation. The future research that can follow this work is discussed and the publications and references related to the present research are detailed here.

CHAPTER 2

BACKGROUND CONTEXT, STATE OF THE ART AND RELATED TECHNOLOGIES

"If you know others and know yourself, you will not be imperiled in a hundred battles; if you do not know others but know yourself, you win one and lose one; if you do not know others and do not know yourself, you will be imperiled in every single battle."

General Sun Tzu ("The art of war", 6th century BC)

This chapter introduces the key theoretical concepts that are needed for a suitable understanding of this research. Moreover, the chapter analyses the state of the art for the addressed problem and provides a further introduction to the main technologies related to the present dissertation.

2.1.Background Context

This research has dealt with some aspects of technology that have evolved very rapidly in recent times. Interactive TV (in its current form of Smart TV or Hybrid TV), Progressive Enhancement, and User Experience, among others, evolve according to technology. Others, such as the concepts of platform and runtime environment are computer science fundamentals that are also related to the present work.

2.1.1. Interactive TV

The term of interactive television has been historically used for describing different technologies for the user interaction on TV systems. Smart TVs are the current generation of interactive television, but there are still many challenges to overcome.

Introduction

Traditionally, interactive TV (iTV) describes the number of techniques that enable viewers a certain level of interaction with TV contents. This definition comprises any kind of interactivity between the viewer and the TV set, the contents or the program being watched. It represents a continuum from that is considered:

- <u>Low interactivity</u>: In which viewers interact with the TV set and are able to change the channel and the volume or decide to turn on/off the TV set.
- <u>Moderate interactivity</u>: When interaction is with the contents of the TV program and viewers can access to on-demand content and participate somehow in the flow of

the program (e.g. chat and SMS systems, polls, questions, etc). This service is unidirectional.

- High interactivity: This enables the access to TV-related content for getting, for instance, enhanced information about what is on TV. This is the most challenging profile and it has not been until now fully developed with the inclusion of web technologies that iTV systems can clearly reach this level of interactivity.

Although interactivity may seem new according to the press and popular perception, the fact is that scientific publications have been appearing on a regular basis for some time now. Marketing efforts by the new iTV platforms are a reality and has been so for a while. The fact is that the history of iTV covers the last five decades. During this time, the number of successes of iTV has been accompanied by many failures due to the both the lack of maturity of related technologies and infrastructures, and the lack of suitable contents or the absence of a clear market demand. However, the idea behind the iTV paradigm causes still a great interest to the broadcasting industry and the scientific community. Moreover, the technological revolution of the past years regarding Internet, social networks, mobile communications and content business models seems to give support for a new wave of TV interactive digital media. Recent years have brought many changes in the way people consume multimedia contents, especially on mobile platforms, where platform capabilities are blended with Internet resources to provide greater functionalities and enhanced user experience. This positive disruptive innovation can be also translated to the TV context. Since TV platforms include Internet services in addition to broadcasted contents, iTV can be understood as "the merger between conventional television and new interactive information and communication technologies" 0.

In order to better understand the current context and the many challenges in iTV that are lying ahead, it is necessary to have a brief summary of his history and evolution.

Brief History

The background of the modern iTV era goes back to the 1920s and the telephony companies, when the first experiments with video telephony were carried out. Bell Telephone Laboratories succeeded in sending and receiving pictures via the telephone lines and, in the 1960s, the first telephone where the person who was talking could be seen was presented to the market. Although this video telephony product was expected to be very well received by users, especially in the business world, it never caught on because of some technical weaknesses related to the own device usability and the fact that most people do not felt comfortable with the idea of being seen when talking by telephone [2]. This was the first false-start of the video telephony, which was not actually spread until early the 2000s with the 3G irruption and the mobile technology.

The first actual approach to iTV was driven in the 1970s as a cable TV system based on analogical transmission, offered as a TV service with a channel pack and accessed from the STB provided by the provider company. The system was known as QUBE and offered a limited set of interactive channels that enabled the user participation in some game shows, sports events and polls [3]. Once again the world was expecting the revolution of this kind of new interactive service but its deployment success was overestimated. Although the service got many subscribers, it was too expensive to argue the maintenance of the return channel and could not compete with the audience of the big TV networks. In 1984 the system was closed down which meant the second big false-start of the interactive media era, the first for iTV as it is known today.

In the 80s, some iTV services were launched with varying success but the key factor of the period was the simplicity of the offered services in contrast to the ambitious proposals of the previous services. Regarding iTV, the main legacy of this period were the interactive text and the participation on broadcast programs via telephone services, which exists still today in an evolved form. On the one hand, some cable companies of the 80s offered services with two-way transmission capabilities providing interactive teletext and simple graphics. On the other hand, telephone companies developed STB for accessing to TV contents and interactive services via the telephone line. Both cases were well received by the users but not enough to provide the expected revenues to its promoters, mainly due to the costs of the related technology. During the 80s the technology evolution also enabled the reduction of its costs. Cable and telephone companies could enhance and spread their services but the interactivity provided was limited, and the services offered did not match user expectations.

In the early 90s, the first significant user studies with iTV revealed that most users were more interested in what the service allowed to do than in what was the subject area of the service. It was learned that “people want to have fun, have something, learn something, and tell somebody about it” and that “the service must be simple to use for the consumer. They want television, not computers, and they do not want to use anything that implies even a hint about complex pc-operations” [4]. The Time Warner’s so-called Full Service Network (FSN) was an ambitious iTV project providing the full spectrum of interactive services. As others, it crashed spectacularly, just two years later, but the responsible of the project reported some interesting learned lessons as “1) the service itself must be available to the consumers free of charge; 2) different gradual price models do not work; 3) Video-on-Demand (VOD) is a very popular use of the technology; 4) people want really simple interactive possibilities” [5].

In the late 90s, the great success of Internet led to seriously consider that the World Wide Web in connection with the computer capabilities were the key recipe for an actual success of iTV as a "Marriage of convenience", as stated by M. Krantz’s in 1997 [6]. The convergence of the iTV world with internet is a tendency that has been evolving since beginning of the millennium, period in which a number of standardization agencies and international industrial consortiums (i.e. DVB) around digital television and iTV were consolidated. From this period also remains the switch off of analogue television, which is still pending in some countries, and some other trends that are still in continuous evolution as the personalized television via STBs and the cross-platform return channel.

A STB – understood as a device that connected to the television set and an external source of signal enables the access to television contents and their interactive services – allows the connection to broadcast signals as well as satellite, cable and Internet. It has taken many different forms and its definition is sometimes blurred by the increasing deployment of Digital Video Recorders (DVRs), multimedia discs and satellite or cable tuner boxes with hard discs, network or USB interfaces built-in. The interactive services provided usually depend on the built-in applications, the return channel connectivity and the offer of the platform provider.

The cross-platform return channel refers to the multiple channels that the television programs have been using for promoting user participation in their shows. It began with simple phone calls and emails addressed to the television program followed by SMS services, which reached such popularity that this kind of television was termed SMS TV [7]. This kind of iTV enabled (and enables) instant messages for participating in games, polls and chats via the mobile phone, which introduced the mobility in the world of iTV. Those services also generate a revenue volume for broadcasters that is sometimes bigger than the obtained by advertising. Later, with the spread of the social networks and smartphones, this tendency has evolved into a cloud of interactive services that take advantage of the new technologies for enhancing user engagement.

The convergence with internet also brought the addition of a web browser to fully access the Web and the mix up of web contents with television programs. Philips NetTV was one of the first systems of hybrid TV sets with an integrated browser for accessing to the web without restrictions, followed by others manufacturers such as LG, Sony and Sharp among others. This kind of television was termed Enhanced TV and although it sounded as if this could be the next iTV revolution, the lessons learned during the previous experiences helped to bring prudence to the generated expectations. Previously, iTV was expected to be a 'revolution' but some years ago it was learnt that iTV must be just an 'evolution' 0. Currently, this trend has been evolving to the current terms of Smart TV, Connected TV and Hybrid TV (not confuse with HbbTV, which is a specific standard for implementing Hybrid TV systems), where the paradigm of television with enhanced contents coming from Internet changes to a real convergence between the television and the computer world. This change has been driven by the recent technology advance as well as by the new generated services on different platforms and its deployment is creeping in slowly but surely, at least according to the products and services presented recently in the market.

Future Challenges

The near future of iTV has still to be written. Currently, the main challenge seems to be the integration of all types of networks (Internet, Internet of Things, broadcasting, mobile networks etc), which enables the access to TV content anytime and anywhere by universal terminals [8]. The past false-starts have provided important information regarding the requirements and user expectations around what is supposed to be the success of iTV, but it has also demonstrated its challenges and weaknesses. Still, the current push of interactive services on any platform and the verified success of smartphones as connected platform suggest a silver lining for this next generation of iTV.

Consumers are demanding the access to interactive services anywhere, anytime and from any device but they still find access barriers related to platform compatibilities, network segmentations, and service limitations. Nowadays, there are two key scenarios that are presented as the killer application of this convergence: the home environment and ambient intelligence spaces.

In the home environment some recent market studies have demonstrated that users consume multiple multimedia contents at the same time and from different sources [9][10]. It is usual to write emails, chat, participate on television programs, play games and share in Internet what is happen while one is watching TV, especially with smartphones and tablets, but most of the used services still live in walled gardens that are maintained by the big IT companies. Although the web claims its role as the true cross-platform solution and many efforts have been made in that direction, the market battle for the management control of the integrated multimedia experience in the home environment is assured.

The home environment is a typical context of ambient intelligence environment. Ambient intelligence environments are understood as electronic environments that are sensitive and responsive to the presence of people, including television or audiovisual contents. The evolution of sensors, sensor networks and displays are enabling a cloud of new proposals for enhancing the everyday life activities of people using information and intelligence that is hidden in the network connecting the devices that are available in a particular environment, which also has been known as Internet of Things. As this heterogeneous scenario requires a high degree of communication between the involved devices, the cross-platform web solutions seem more convenient to provide the required ubiquity. Furthermore the business models behind ambient intelligence environments are still more blurred than, for instance, in the mobile technology. This has probably caused the lack of big walled gardens in this area.

From a technological point of view, this scenario of interactive applications is demanding more efforts in cross-platform solutions to ensure the suitable access by anyone, from anywhere and anytime. This dissertation addresses some of these challenges focusing on interactive TV and the development of cross-platform applications.

2.1.2. Platform, Cross-Platform, Platform-Agnostic and Other Trends

In computing, a platform is a combination of hardware and software used to achieve a computing objective – given resources for a given application at a requested performance. A platform can only embrace software architecture but it can also comprise hardware architecture, or the combination of both. Usually, it includes: computer architecture, operating system, middleware and software components to build applications, which are based on programming languages and user interfaces. Basically, a platform can be defined as the place where application software resides and runs. This is the reason why a platform must provide a suitable software framework for running applications, but this framework can differ dramatically from one platform to another even when platforms are designed to accomplish the same tasks. Application portability between different platforms is strongly dependent on the software framework, which is provided by each platform.

Cross-platform or multi-platform refers to the computer software capability for running on multiple computer platforms. Typical examples of cross-platform software are operating systems, which usually run on a heterogeneous set of hardware architectures (e.g. Linux, Microsoft Windows, Mac OS X, Android, iOS, etc). A cross-platform application can be specifically compiled for each target platform as well as designed for being deployed in multiple platforms with no further adaptations. When an application has been written for running in multiple platforms without being compiled is usually considered a platform-independent or platform-agnostic and it requires that the target platforms include platform-specific capabilities to support the application and provide the same functionality into any target platform. Those are generally the cases of web applications, which require a native web browser, and Java applications, which require the deployment of a virtual machine installed on the target platform. In these cases, as in most of recognized platform-independent applications and software environments, the underlying differences between the addressed platforms usually require also some platform-specific adaptations for ensuring a suitable support.

Platform-specific applications are those specifically built for running in a particular platform. This enables a direct use of the functions provided by the runtime environment of the platform, which usually implies an improvement of the performance when running the application and the use of the native look and feel functions for presenting the application in a similar way than the rest of the system applications. Platform-specific applications are also called native applications.

The term platform-agnostic is differenced sometimes from the term device-agnostic since it makes reference specifically to the software that operates across various types of devices, including desktop computers, laptops, tablets and smartphones.

2.1.3. Runtime environments

Typically, these application formats can be classified depending on the runtime environment into declarative or procedural application formats. On the one hand, Declarative Application Environments (DAEs) are based on a browser for the presentation of user interfaces written in a declarative language, which is typically based on XML and includes scripting support for interaction with network server-side applications and access to the Application Programming Interfaces (APIs) of the target client middleware. DAE applications are associated with a collection of documents written commonly in HTML, JavaScript, CSS, XHTML and SVG. This is the most widespread solution for deploying interactive applications on IP-based networks like Internet and proprietary IPTV networks. On the other hand, Procedural Application Environments (PAEs) are those able to locally execute an application written in a procedural programming language, which is any programming language in which the programmer specifies an explicit sequence of steps to follow in order to produce a result (i.e. an algorithm). Examples of procedural programming languages are C/C++, Java, Perl and Python among others. For TV, Java is still the core of the most widely used middleware for broadcast iTV environments by its multiplatform nature.

2.1.4. Progressive Enhancement

Progressive Enhancement is a strategy for web design that emphasizes accessibility, semantic HTML markup, and external style-sheet and scripting technologies. Progressive enhancement uses web technologies in a layered fashion that allows everyone to access the basic content and functionality of a web page, using any browser or Internet connection, while also providing an enhanced version of the page to those with more advanced browser software or better bandwidth.

The Progressive Enhancement strategy is an attempt to subvert the traditional web design strategy known as Graceful Degradation, wherein designers would create Web pages for the latest browsers that would also work well in older versions of browser software. Graceful Degradation was supposed to allow the page to degrade, or remain presentable even if certain technologies assumed by the design were not present, without being jarring to the user of such older software. In practice, Graceful Degradation has been supplanted by an attitude that the end user should Just Upgrade. This attitude is due to time and budget constraints, limited access to testing alternative browser software, as well as the widespread belief that browsers are free. Unfortunately, upgrading is often not possible due to IT department policies, older hardware, and other reasons. The Just Upgrade attitude also ignores deliberate user choices and the existence of a variety of browser platforms; many of which run on handhelds or in other contexts where available bandwidth is restricted, or where support for sound or color and limited screen size, for instance, are far different from the typical graphical desktop browser. There is also a large group of Internet users unaware or uncaring about the features of the browser.

In Progressive Enhancement the strategy is deliberately reversed: a basic markup document is created, geared towards the lowest common denominator of browser software functionality, and then the designer adds in functionality or enhancements to the presentation and behavior of the page, using modern technologies such as Cascading Style Sheets (CSS) or JavaScript (or other advanced technologies, such as Flash or Java applets or SVG, etc.) All such enhancements are externally linked, preventing data unusable by certain browsers from being unnecessarily downloaded.

2.1.5. User experience

User experience applied to computing systems refers to the perception and feelings that a user obtains when using a computing system [11]. Its measurement reveals valuable aspects of human-computer interaction and practical aspects such as utility, ease of use and efficiency of the system. So that, user experience is subjective and dynamic in nature, because it is based on people's thoughts and these thoughts can vary in time and be influenced by the context.

The user experience evaluation methods can be classified on emotion assessment, and on overall user experience assessment. While the first group tries to identify the emotional data observing psycho-physiological measurements or expressed emotions, the second group investigates how a person feels about a system as a whole, typically after using it for a while. Typical measures of overall user experience evaluation are:

- Utility: Does the user perceive the functions in the system as useful and fit for the purpose?
- Usability: Does the user feel that it is easy and efficient to get things done with the system?
- Aesthetics: Does the user see the system as visually attractive?
- Identification: Can I identify myself with the product? Do I look good when using it?
- Stimulation: Does the system give me inspiration? Or wow experiences?
- Value: Is the system important to me? What is its value for me?

2.2.State of the Art

This section describes current context of the technologies involved on the development of iTV applications, with a special focus on cross-platform solutions.

2.2.1. Current iTV Platforms and Middleware

The current trend for the convergence of TV and Internet is flooding the market with a number of platforms and middleware that enable the access to broadcast contents, online interactive media and its combination as over-the-top content. On the one hand, multiple third-party companies and organizations are providing iTV middleware for TV sets and STBs. Their strategy is based on offer an attractive and efficient iTV software system that could be installed in the largest possible number of TV platforms, but this craving for being universally supported sometimes clashes with the facilities provided by device manufacturers. On the other hand, vendor's specific iTV platforms ranging from Smart TV sets to game consoles are offering fully integrated iTV solutions that enable an easy connection with same family devices to build connected environments. However, their solutions are usually limited for connecting devices from the same manufacturer, which also can filter the some content due to strategic market policies.

Most relevant iTV middleware managed by standardization bodies

- Multimedia Home Platform (MHP) [12]: Open middleware system standard designed for iTV by the DVB project, which is an industry-led consortium for the design of open technical standards for the global delivery of digital television and data services. MHP enables the reception and execution of interactive Java-based applications as well as

HTML applications on a TV set. It has been mainly promoted from European authorities for enabling iTV services in a standardized way. Further details regarding this standard are given in below Section 2.4.

- MHEG-5 (or ISO/IEC 13522-5) [13]: Specification standardized by the Multimedia and Hypermedia Experts Group (MHEG) for the presentation of multimedia information and iTV services. It is cost-effective, license-free, efficient, public standard interactive TV middleware that is used both to send and receive interactive TV-based contents. It is currently an alternative technology to HbbTV among others that is mainly used in United Kingdom, New Zealand, Hong Kong, Ireland and Australia (Commonwealth Countries). MHEG-5 is an object-based declarative programming language which can be used to describe a presentation of text, images and video. An MHEG-5 application consists of a number of scenes where the user of the application can navigate. Each scene lists the items of text and graphics to be presented and can contain blocks of procedural code which are executed in response to one of a predefined set of events such as keys being pressed, timers firing or content being successfully loaded into memory. These blocks of code consist of elementary actions which can perform operations such as changing the text displayed by a text object, or starting a video clip playing.
- Ginga (ISDB-T Middleware) [14]: Middleware specification for the Japanese-Brazilian Digital TV System and an ITU-T Recommendation for IPTV Services and Cable Broadcast services. Ginga development was partially based on a set of standardized technologies but mainly on innovations developed by Brazilian researchers. Its current reference implementation was released under the GPL license and is divided into two main integrated subsystems. Ginga-NCL for declarative Nested Content Language (NCL) applications (mainly for IPTV services) and Ginga-J for Java applications (mainly for broadcast TV).
- Hybrid Broadcast Broadband TV (HbbTV) [15]: HbbTV an industry standard promoted by the HbbTV consortium for the harmonization of broadcast, IPTV and broadband delivery of television entertainment to the end consumer via connected TVs (i.e. Hybrid TVs or Smart TVs) and STBs. It offers an open platform as an alternative to proprietary technologies for operating over different broadcasting technologies, such as satellite, cable, or terrestrial networks as well broadband networks as Internet or dedicated IPTV networks. HbbTV is based on elements of existing standards and web technologies including the Open IPTV Forum, CEA, DVB, and W3C. The HbbTV specification is expected to become one of the main platforms for interactive TV services in Europe, where several countries have adopted it and operated HbbTV services. In early 2012, HbbTV services are in regular operation in France, Germany and Spain; announcements of adoption have been made in Austria, Czech Republic, Denmark, Netherlands, Poland, Switzerland, Turkey; and trials, in Australia, China, Japan, and the US.

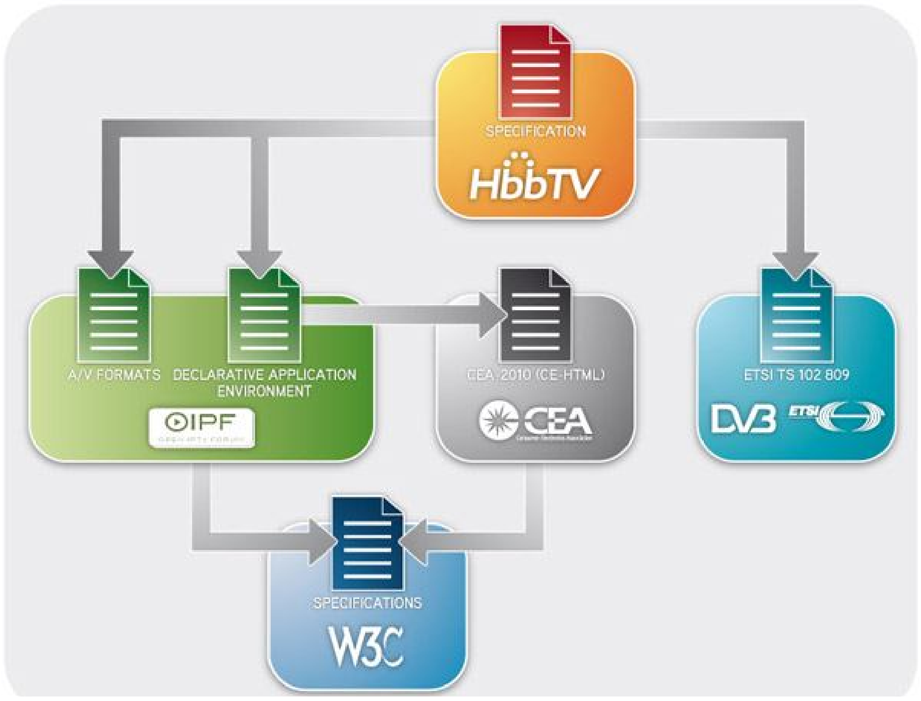


Figure 1. Relationship between HbbTV and other existing standards[2].

- Open IPTV Forum (OIPF) specifications [16]: The Open IPTV Forum certification program has not yet been finalized and so no Open IPTV Forum compliant set-top boxes or residential gateways are available today. However, they are the base for other iTV standards that include IPTV technologies such as HbbTV. OIPF is an industrial association for the proposal and promoting of IPTV technologies that provides specifications, profiles, tests and certifications in order to enable a global IPTV market with platforms, applications and services. Their specifications enable end-to-end IPTV systems with public standards that are used by a number of provider companies and accepted by related standardization organizations such as 3GPP, ETSI, ATIS and the Open Mobile Alliance. The Open IPTV Forum is a European initiative that currently includes members around the world and expects to increase its global representation further. Its specifications will have world-wide applicability.
- Tru2way [17]: Brand name for interactive digital cable services delivered over the cable video network that is used for market cable services, applications, and devices that support the tru2way cable architecture. Tru2way is the successor for the technology known as OpenCable, hardware and software specifications developed in the United States by CableLabs to define the next-generation cable TV device.

Most relevant iTV middleware managed by specific editors

[2] Source: http://www.nagra.com

- Boxee [18]: Cross-platform freeware media center marketed as the first ever "Social Media Center" for its combination of the iTV capabilities with the broadband social networking [19].
- Google TV [20]: Smart TV platform co-developed by Google, Intel, Sony and Logitech integrating an Android-based system for TV to create a software interface between TV contents and Internet services in which users can install third-party applications through Google's application market [21].
- Microsoft Mediaroom [22]: Latest update of the Microsoft's software for TV platforms aiming to get the lion's share of the Smart TV market. It enables the access to on-demand content, Internet services, live television programming and third-party applications developed through the Microsoft Mediaroom Application Development toolkit.
- TVBLOB [23]: Middleware used by the Tiscali's TV box for providing a TV-centric platform that marries the access to TV contents with Internet services and over-the-top TV experiences.
- Ubuntu TV [24]: Smart TV operating system based on Ubuntu and designed for the living-room TV that is capable to integrate all standard terrestrial broadcast system and also satellite and cable services as well as Internet services and applications designed specifically for the system. With a Linux-based operating system for TV sets and its own TV content services, Ubuntu TV can compete with other systems such as Google TV for the collaboration with TV manufacturers [25].
- XBMC Media Center [26]: Free and open-source middleware developed by the XBMC Foundation for use with televisions and remote controls. It is a customizable cross-platform solution that was originally designed as a media center application for the original Xbox game console, but since 2010 it is officially available as a native application for Linux, Mac OS X, iOS and Microsoft Windows operating systems.
- MeeGo for Smart TV [27]: MeeGo is a Linux-based open-source operating system primarily built for mobile devices that has been extended to a number of different hardware platforms such as notebooks, tablets, in-vehicle devices, Smart TVs, IPTV-boxes and other embedded systems. The Smart TV branch is currently based on a derivate fork of XBMC media center software.
- Yahoo! Connected TV [28]: This is the Smart TV solution developed by Yahoo!, which integrates the Yahoo! Widgets system for an enhanced television experience with internet connected applications.

Most relevant vendor's specific iTV platforms

- LG Smart TV: LG Electronics include the LG NetCast middleware to his TV sets allowing audience to receive information from the Internet while at the same time watching conventional TV programming.

- Panasonic Viera Cast: Smart TV platform created by Panasonic that makes it possible to stream multimedia content from the Internet directly into select Viera HD TVs and Blu-ray disc players.
- Philips Smart TV: The built-in platform for providing Smart TV solutions on Philips TV sets, based on the Open IPTV Forum [16] standards.
- Samsung Smart TV: The Smart TV platform developed by Samsung enables interconnectivity with Samsung mobile phones for extending the TV capabilities. Samsung also has deployed its own application market, known as Sumsung App Store, which provides interactive applications for the Samsung TV platform.
- PlayStation 3: While primarily designed as a game console by Sony, it features many typical Smart TV capabilities as content on-demand and internet access.
- Xbox 360: This game console developed by Microsoft also includes many typical Smart TV features but unlike other platforms it has no public software interfaces for building third-party applications.
- TiVo [29]: It is a Digital Video Recorder (DVR) developed and marketed by TiVo, Inc. and introduced in 1999. TiVo provides an on-screen guide of scheduled broadcast programming television programs, whose features include "Season Pass" schedules which record every new episode of a series, and "Wish List" searches which allow the user to find and record shows that match their interests by some fields such as: title, actor, director, category, or keyword. TiVo also provides a range of features when the TiVo DVR is connected to a home network, including film and television show downloads, advanced search, personal photo viewing, music offerings, and online scheduling. Since its launch in the United States, the TiVo has also been made available in New Zealand, Canada, Mexico, Australia, Taiwan, the United Kingdom and Spain.

Many other TV set manufacturers such as Sharp, Sony, Loewe and Toshiba have built also their own Smart TV solutions.

Most relevant set-top box vendors

- Motorola [30]: According to Infonetics Research [31], Motorola is the globally top set-top vendor in terms of revenue share, followed by EchoStar, Pace and Cisco. Inspired by their vision of Seamless Mobility, Motorola creates products, experiences and powerful networks for users to get and stay connected. Within IP Video Solutions this includes encoders, video servers and consumer premise equipment, such as the VIP line of IP set-top boxes and the KreaTV Application Platform.
- EchoStar [32]: EchoStar’s set-top box technology capabilities include everything from basic, standard features and functionality to advanced, value-add features. Product functionality is customizable to fit the operator’s immediate needs, and also claims to provide for added functionality in the future. EchoStar’s Set-top box technologies

include SD/HD, DVR, Digital Living Network Alliance (DLNA), remote controls, iTV apps, Digital Rights Management (DRM), WiFi and Dolby Digital among others.

- Pace [33]: This company develops STBs, advanced residential gateways, software and services for the pay-TV and broadband services industry. Pace's customer base includes cable, telco, satellite and IPTV operators in markets across the globe.
- Cisco [34]: Cisco provides TV STBs for cable networks around the world. They offer a number of devices for standard definition TV, high definition TV, DVR and DVB networks.
- Technicolor [35]: Formerly Thomson S.A. and Thomson Multimedia, is a French international provider of solutions for the creation, management, post-production, delivery and access of video, for the Communication, Media and Entertainment industries. Their range of STBs provides a wide array of services going from analog TV switch-off or HD TV transition to advanced functions such as full 3D experience, multi-room and multi-screen, on demand, wireless streaming or over-the-top services.

Most relevant media centers

- XBMC [26]: Apart from being a complete iTV middleware, XBMC is also provided as a media center application that can be installed in Linux, Mac OS X and Microsoft Windows systems.
- MythTV [36]: It is a free and open source home entertainment application for Linux and Mac OS X systems.
- Freevo [37]: Freevo is an open source media centre. It integrates DVR functionality along with music, video, gaming, home automation and more. It is written in Python and uses existing popular software such as Mplayer, Xine, Vlc and Skype. Freevo also provides access to popular services such as YouTube, Flickr, Apple trailers, IMDB, Hulu Desktop and more through integrated plugins.
- Moovida [38]: A cross-platform media center software with stylish design and customizable user interface for TV and PC. The paid version, called Moovida Pro and created by Fluendo, includes the Fluendo codecs to guarantee that mainstream formats are supported on an optimal and patent compliant way.
- Windows Media Center [39]: It is a digital video recorder and media player developed by Microsoft. It is an application that allows users to view and record live television, music and videos. The application is included in various versions of Windows XP Media Center Edition, Windows Vista Home Premium and Ultimate and all editions of Windows 7 except for Starter and Home Basic.
- Linux Media Center (Linux MCE) [40]: LinuxMCE is a free, open source add-on to Kubuntu which is mostly licensed under GPL. Third-Party developers can make and submit patches and new features.

2.2.2. Cross-Platform Content Description Formats for iTV

The number and variety of iTV platforms, middleware and application formats hinders the generation of cross-platform applications and usually implies the building of multiple branches of the same application to fit the requirements of the different platforms and devices. This is time consuming and increases development costs. A different approach can be followed by abstracting the content description language providing a platform-independent description of the interactive contents. This abstract language should unify the design of multimedia applications for all iTV platforms as well as providing facilities for translating this abstract description into different platform-specific formats. However, there is not a unified standard language with which create interactive applications that follow the principle of "build once, run everywhere".

The lack of a unified standard for the content description has driven this research to analyze the existing content description formats that enable the creation of platform-independent multimedia contents for iTV. The main criteria for the choice of this abstract language are focused on ensuring the maximum platform independency across formats and devices: (1) it must provide a high-level content description rather than specify the implementation details for addressing a specific platform; (2) it should be an open standard rather than a proprietary format; and (3) it should be easy to read and understand by non-expert developers. This analysis is based on previous experiences on the field [41], but language selection has been updated according to the current context:

- User Interface Markup Language (UIML) [42]: UIML is a XML-based markup language for user interfaces standardized by the Organization for the Advancement of Structured Information Standards (OASIS) for defining user interfaces on different devices. UIML tries to reduce the work needed to develop user interfaces by describing in declarative terms the interface elements and abstract it from its detailed implementation on a specific platform. This standard is not specific for TV systems and its transformation to fit some application requirements can imply a difficult work. Moreover, its particular syntax requires some expertise to build applications.
- Multimedia Content Description Interface (MPEG-7) [43]: MPEG-7 is the ISO/IEC standard for the description of interactive content in a wide range of applications. It uses the Description Definition Language, which is an extension of XML Schema, for describing the content structure. A core part of MPEG-7 are the Multimedia Description Schemes, which provide support for the description of media information, creation and production information, content structure, usage of content, semantics, navigation and access, content organization and user interaction. Metadata is stored in XML, and can be attached to time code in order to tag particular events, or synchronize lyrics to a song, for example. MPEG-7, as UIML, enables a high level of flexibility in the content design due to its platform

independency. However, this freedom requires an extra effort in structuring standardized contents and sometimes implies portability issues.

- Worldwide TV Mark-up Language (WTVML) [44]: WTVML is a XML-based format for developing and deploying iTV services, mainly developed by the British Sky Broadcasting Group and standardized through ETSI as TS 102 322. It is integrated in the WapTV system, which is an Interactive television technology platform comprising a microbrowser, a markup language, and a significant collection of associated software tools and services. The microbrowser and mark-up language are both based upon the Open Mobile Alliance WML 1.3 specification. WTVML is the ETSI recommendation to achieve platform interoperability by allowing the authoring of iTV applications that can be translated into any HTML-based format, while maintaining compatibility with existent middlewares such as MHP and OpenTV via native microbrowsers [45]. However, the use of a microbrowser may imply a performance penalty compared to native applications.
- Nested Content Language (NCL) [46]: NCL is another XML-based declarative authoring language for hypermedia documents. It provides several facilities for generating interactive applications with synchronization relationships among its components. Following other XML standards, NCL is specified by a modular approach. NCL modules can be added to standard web languages, such as XLink and SMIL. NCL was initially designed for the Web environment, but currently it is included as the declarative language of the Japanese-Brazilian ISDB-Tb (International Standard for Digital Broadcasting) terrestrial TV middleware (named Ginga). It is also the first standardized technology of the ITU-T multimedia application framework series of specifications for IPTV services.
- Digital Video Broadcasting – Portable Content Format (DVB-PCF) [47]: DVB-PCF is a data format designed by the DVB project for the description of iTV services. It is intended to support the business-to-business interchange of interactive content and to enable deployment on multiple target platforms with a minimum amount of re-authoring. DVB-PCF is a platform-independent description of "what" the viewer experience should be, rather than "how" it should be achieved. This description must be transformed into a platform-specific format by a "transcoder". This transformation step uses the available features of a particular platform to create the viewer experience described in DVB-PCF. It supports independent description of different aspects of the interactive service, i.e. content, presentation (layout and style), behavior and navigation. Furthermore, DVB-PCF does not require all aspects of a service to be described as one physical unit, such as a file. For example, service descriptions can be arbitrarily distributed across files located on the Internet, using references represented as Uniform Resource Identifiers (URIs).
- Scalable Vector Graphics (SVG) + Synchronized Multimedia Integration Language (SMIL) [48]: SVG is a family of specifications standardized and recommended by

World Wide Web Consortium (W3C) for describing two-dimensional vector graphics, both static and dynamic (i.e., interactive or animated). It is an XML-based language that can be used as a cross-platform production format since it is not attached to any specific platform. Its combination with SMIL enables time-based modifications to the elements of the scene, which makes SVG a suitable format for describing TV user interfaces. Both SVG and SMIL are further described below in Section 2.5 of this document as part of the HTML5 standard family.

- HyperText Markup Language (HTML/HTML5) [49]: This is the main markup language for displaying web pages and other information that can be displayed in a web browser. Although it is not attached to any specific platform, its flexibility and included technologies can hinder its transformation into other application formats, especially procedural. This standard is also described in detail in Section 2.5 of this document.
- Extensible HyperText Markup Language (XHTML) [50]: This is a family of XML markup languages that mirror or extend versions of the widely-used HTML, the language in which web pages are written. XHTML is an application of XML that is more restrictive than HTML and requires that XHTML document is well-formed, which enables the parsing through standard XML parsers. Although it facilitates the parsing compared to HTML, it is neither focused on TV contents – and some TV-specific features can be difficult to implement and translate into other kind of TV content formats.
- Extensible Markup Language (XML) [51]: This is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications. Hundreds of XML-based languages have been developed including RSS, Atom, SOAP, and XHTML. XML-based formats have become the default for many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (OpenDocument), and Apple's iWork. XML has also been employed as the base language for communication protocols, such as the Extensible Messaging and Presence Protocol (XMPP). Particularly, in terms of user interface design of iTV services, it can assist in the achievement of a generic user interface transformed into the different formats that each device and platform requires quickly, allowing rapid prototyping of the user interface design though the creation of a new customized schema.

2.2.3. The Web as a Platform for TV Services

While the key technology for bringing interactive TV contents to a wide range of connected homes is still under discussion, there is a bloody but profitable battle of platforms in the mobile context. According to [55], Symbian OS still maintains around the 20% of the mobile market around the world but it recently lost the leadership on behalf of the new advanced operating systems for smartphones such as Android and iOS, which cover around 23%. The arrival of smartphones and HTML5 has popularized mobile web applications, which are a new family of web applications that are especially addressed for mobile devices. HTML5, as explained in detail in the section 2.5, provides important innovations for the generation of offline applications that are increasingly closer to the native applications offered by the mobile platform providers. That is: enabling web applications with similar capabilities (performance, look & feel, etc) than native ones. Moreover, the discussion between native and web applications is not restricted to mobiles, and it is also valid for any smart platform such as the current iTV environments.

The web is currently seen as the key platform to deal with the proprietary silos built by the main companies such as Google, Apple and Microsoft. These companies are providing their own operating system, application formats, services, advertisement systems, and application stores. They manage and control the entire platform where users can access to interactive services from different devices, especially on mobiles, but the generated applications are usually incompatible from one platform to another. Samsung is seriously pushing for getting a privileged place in this race with the key difference that it is also an important TV manufacturer. This enables Samsung to provide comprehensive built-in iTV solutions, such as the presented in its smart TVs, which can be easily connected to its mobile devices for an improved multimedia experience. Samsung has also created its own application market (Samsung App Store) and smartphone operating system (Bada) in order to compete with the rest of the current proprietary silos.

In this context, the web can be considered as the third platform besides Apple and Google because it also provides its own services (web services), its own application format (based on the HTML family) and a big marketplace for offering applications such as the web itself, where developers can deploy their solutions without requiring the license of a silo owner. The main advantage of the Web over the other big platforms is that its applications (now, HTML5 applications) are by far the most portable and device-independent. No other technology has ever reached the level of ubiquity than HTML5 applications thanks to the recent advances on browser technologies in all multimedia environments.

Regarding iTV systems, the MHP experience has demonstrated that the deployment success of a specific technology like this depends heavily on the agreement between both national administrations and industry. The standard was pushed and promoted but the promise of having MHP interactive services on TV was vague and insufficient for most users. MHP is a complete platform for accessing to iTV services that defines all the hardware and software stack from the low-level resources to the high-level application profiles. Interactive applications in MHP were mainly written in DVB-J, which is a specific Java-based language for creating special applications that are broadcast with a service and are known as Xlets. Although it is based on Java, the particularities of MHP make its learning curve slower and developers require a special training to create interactive applications. Moreover, the performance achieved by MHP receivers when running applications does not satisfy the requirements of most users and developers. From users' point of view, the scenario of MHP services does not justify buying a new device (a STB). Although the MHP specification is freely available for developers, the required technology and device certification significantly increases its final price. From developer's point of view, MHP is a complex platform-specific technology that does not provide a reasonable performance enhancement compared to other cross-platform technologies such as the HTML family. These are some of the reasons for the progressive change from MHP to more suitable technologies such as HbbTV.

An alternative to the procedural formats for iTV, usually based on Java, is the set of languages or descriptions based on a declarative format such as the web-based standards. Some variants of HTML has been defined as suitable formats for different iTV systems such as DVB-HTML for the DVB networks (optional in MHP) and CE-HTML for the Open IPTV Forum, DLNA, HbbTV and Philips' Net TV. These formats push the innovation in the application development due to the fast learning curve that they present by its similarity with the well-known HTML description. Almost all iTV platforms support a variant of a declarative language that is more or less close to HTML. This enables a better connection between the iTV world and the Web.

Currently, with Smart TVs and application markets, iTV systems cannot be separated from an IP connection to access to interactive services via Internet or a private IP network, which is known as Hybrid TV. The convergence between TV and Internet is unstoppable and facilitates the application development (fast learning curve), the application access (via an IP connection), the return channel (implicit in IP networks) and the user understanding what services can be expected from these systems. Most users frequently consume interactive services in Internet and the idea that they can freely access to them on TV is far more attractive than just offering enhanced information, simple games or participating on simple polls and TV contests, which usually have a cost for the user. In a connected TV, users can access to all information available on internet, play any online game and comment what is happening via social networks. This can be seen as the same core services previously offered by iTV provider companies but more focused on the user needs and wants.

Taking into account that the revolution driven by application markets in mobile phones is arriving to the TV environment and that HTML5 is increasingly present as an alternative to platform-specific (and usually non-standard) solutions, it can be said that the Web is becoming also a significant platform for iTV services.

2.3.Reference Format for Content Authoring: DVB Portable Content Format

DVB's standard Portable Content Format (DVB-PCF) has been selected here as the reference format for managing cross-platform iTV applications. This section introduces the format, its architecture and its main features as well as the justification for its choice.

2.3.1. Introduction to the Content Format

DVB-PCF is a high-level and platform-independent declarative description of an iTV service that defines its basics aspects as they relate to the user experience, i.e. the content, presentation and behavior, and leaves to one side the implementation details for each TV platform that will access it.

This content format follows an integration model approach to address the portability problem of interactive applications for TV. It defines a common interface that enables the adaptation to other application formats through a transcoding step, which is platform-dependent by nature. This conversion generates the code of the iTV application that can be then deployed in the target application format (e.g. MHEG-5) and executed in the access platform (e.g. set-top box) using its programming interface.

In order to enable a service to be described efficiently, and in a way that best conveys the author's intention, the format uses a referencing mechanism. This allows the service description to be flexibly partitioned and gives the transcoder considerable freedom to decide how to package the service optimally to suit the target platform.

DVB-PCF is based on existing industry standard formats such as Extensible Markup Language (XML), Multipurpose Internet Mail Extensions (MIME), Unified Modeling language (UML) and Uniform Resource Identifiers (URIs) [52]. DVB-PCF can describe in a declarative way the main features of the existing iTV services, especially those related specifically to TV environments such as the audiovisual consumption, information-driven services, games without many reactive requirements and video synchronized applications.

Moreover, DVB-PCF also tries to unify the production of iTV services by defining a standard format that could be generated from any existing authoring tool. Therefore, as shown in Figure 2, DVB-PCF can be seen as a meeting point between the authoring tools of iTV services and the target platforms that will access to the generated service.

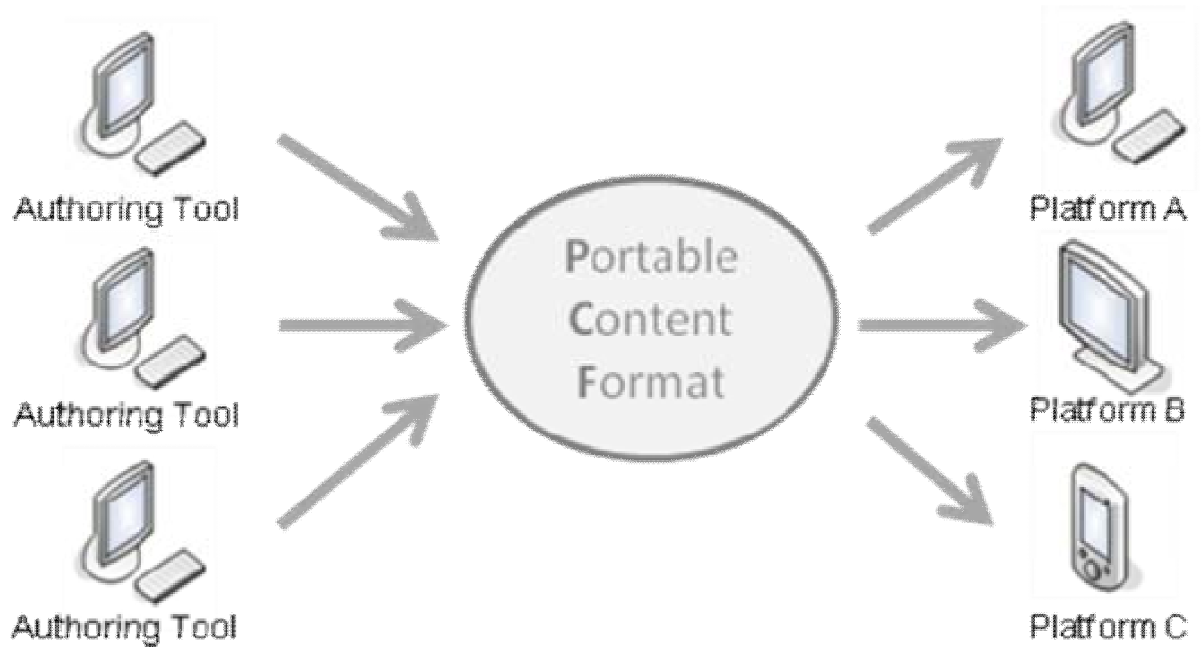
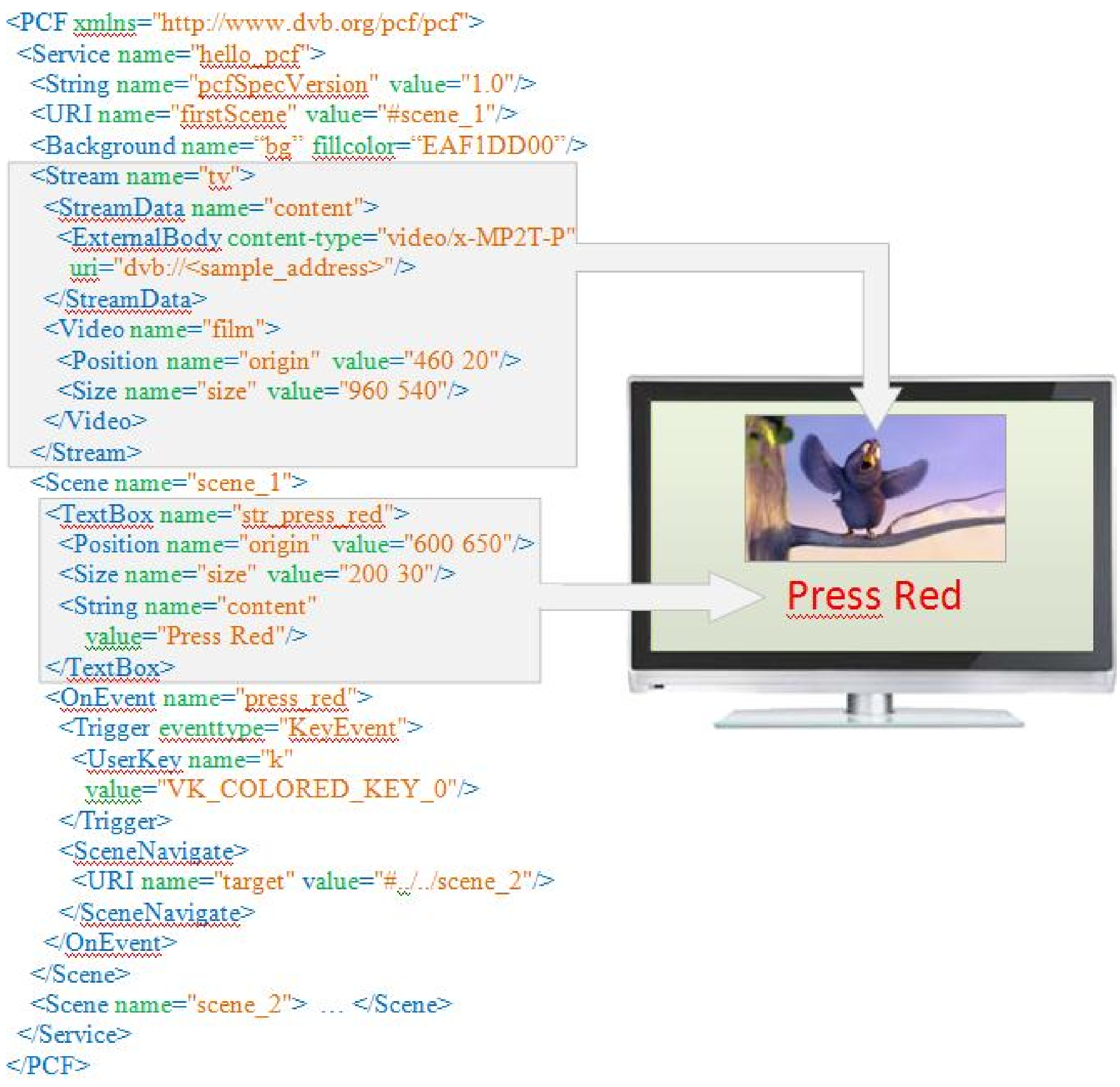


Figure 2. Scope of the Portable Content Format (DVB-PCF).

2.3.2. Component Architecture

Figure 3 shows a basic example of a DVB-PCF service description. In the example, the core aspects of the format are shown, i.e. content, components and behavior (user interaction and navigation between the scenes). The example also contains a stream component, which includes a video component, shared by all the scenes in the service. This service contains two scenes named scene_1, where some DVB-PCF components are defined, and scene_2 respectively. This first scene only contains a text component named str_press_red and its behavior, which defines the trigger (a key event) and the 'to do' action (switch to the second scene). The item named firstScene is used to declare the scene that will be displayed when the service loads.



```
<PCF xmlns="http://www.dvb.org/pcf/pcf">
 <Service name="hello_pcf">
  <String name="pcfSpecVersion"  value="1.0"/>
  <URI name="firstScene"  value="#scene_1"/>
  <Background name="bg"  fillcolor="EAF1DD00"/>
  <Stream name="tv">
   <StreamData name="content">
    <ExternalBody content-type="video/x-MP2T-P"
     uri="dvb://<sample_address>"/>
   </StreamData>
   <Video name="film">
    <Position name="origin"  value="460 20"/>
    <Size name="size"  value="960 540"/>
   </Video>
  </Stream>
  <Scene name="scene_1">
   <TextBox name="str_press_red">
    <Position name="origin"  value="600 650"/>
    <Size name="size"  value="200 30"/>
    <String name="content"
      value="Press Red"/>
   </TextBox>
   <OnEvent name="press_red">
    <Trigger eventtype="KeyEvent">
     <UserKey name="k"
      value="VK_COLORED_KEY_0"/>
    </Trigger>
    <SceneNavigate>
     <URI name="target" value="#../../scene_2"/>
    </SceneNavigate>
   </OnEvent>
  </Scene>
  <Scene name="scene_2">  … </Scene>
 </Service>
</PCF>
```

Figure 3. Code example with the basic DVB-PCF component structure.

In general terms, these are the core aspects for the description of any DVB-PCF application:

- Scenes: The DVB-PCF service item is sub-divided into scenes, which represent the different destinations that can be accessed within the service.
- Components: The building blocks of a DVB-PCF service description. They can be included directly in the service level to define shared objects in all the scenes or in a specific scene of the service.
- Visual components: Visual "widgets" used to describe the visual appearance of the service at a particular point, for example TextBox, Video, Menu and geometric shapes like Pixel, Line, Ellipse and Rectangle.
- Non-visual components: Used for the presentation of non-visual content or providing other aspects of service functionality, for example Audio and Stream.
- Layout: In order to place the visual components on the screen, the layout of each scene can be explicit and the exact positions of the components can be defined. It can also define a flow layout with specific flow rules to position the components according to the other component positions in the scene.
- Content: Managed and presented using DVB-PCF components.
- Behavior: The response to events generated at run-time.

2.3.3. Behavior and User Interaction Capabilities

In DVB-PCF, the behavior of the service and its components is event-driven and it is defined through XML event items and the DVB-PCF action language, which allows the specification of the actions to be executed in an event firing response. There are four types of events:

- System events: Events independent of the user’s interaction and include file/object updates in the transmission stream, media synchronization and in-band triggers.
- User events: Events generated by the user interaction with the remote control keys.
- Component events: A component provides information about a change in its internal state through this type of events. This mechanism allows notifications about the state of the event (e.g. selection and focus) and the event propagation.
- Error events: Specialization of component events that define the rules for propagating the component error conditions within a service.

2.3.4. Structuring a Service Description

The DVB-PCF standard provides a referencing mechanism that allows the distribution of the service description in different local or remote files to accommodate the modularity required for an efficient authoring and for business-to-business interchanges. This is possible due to the href property present in all DVB-PCF components, which allows any DVB-PCF item to be specified by reference rather than by its stated value. It also enables a clear separation of the service layout from its data to facilitate the work of each role involved in the service production and the business-to-business interchange. The file structure of a DVB-PCF service can take advantage of the DVB-PCF referencing model. Although for small services the DVB-

PCF document structure could be simpler, even in a unique file, a smart DVB-PCF document structure enables the distinction between layout documents, which can be updated by an authoring tool, and the data files, which can be generated automatically from the information in a database. Figure 4 summarizes graphically a simple proposal for this file distribution across DVB-PCF documents.

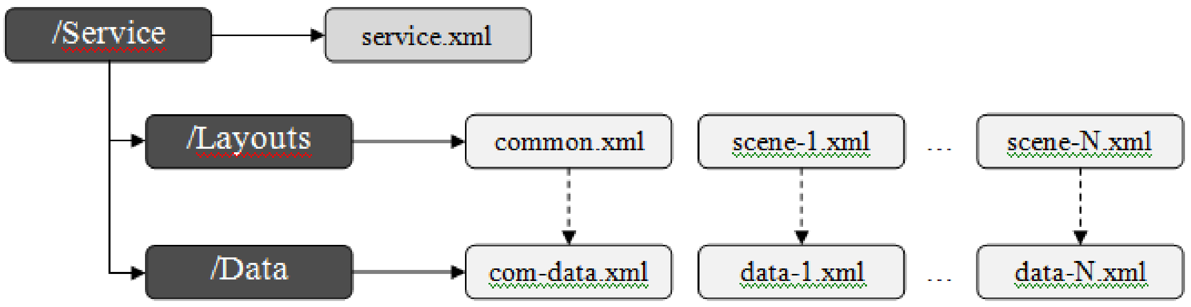


Figure 4. Proposed document structure for defining DVB-PCF services.

The service scene catalog is defined in the service.xml file and it contains references to the corresponding data and layouts for each scene and shared components. Each data file is related to a particular scene, i.e. those that are in the folder data, are referenced from their corresponding layout file in order to display this information in the desired component for a specified scene. This information can be generated dynamically by content producers and published as DVB-PCF files.

2.3.5. Managing Differences between Target Platforms

DVB-PCF also provides a profile mechanism to describe the minimum features of a platform necessary to achieve the requirements of an interactive service. The profiles define pragmatic sub-sets of DVB-PCF functions based on the ability of the platforms to support specific features and the ease of implementing a supporting DVB-PCF transcoder. A DVB-PCF profile shall be defined by reference to one or more profile packages, which contain sets of DVB-PCF features (e.g. return channel). These features shall be grouped into one or more levels that embody the boundaries in platform capabilities (e.g. the presence of a specific feature) and the significant steps in the complexity of DVB-PCF transcoder implementation.

Although custom profiles can be defined, DVB-PCF defines three main profiles:

- Basic: Encapsulates the minimum useful subset of DVB-PCF features for providing a high degree of portability with the minimum requirements for the target platform.
- Core: Includes what is believed to be the maximum subset of DVB-PCF features that can be expected to offer a high degree of portability.
- Full: Encapsulates all the DVB-PCF features.

Once the DVB-PCF profile is defined, it is associated with a service description in the Service element of the service description itself.

2.3.6. Media Synchronization

The media temporal synchronization mechanism provides the means to coordinate audio-visual stream and interactive contents. It is defined in DVB-PCF through StreamEvents, which control the generation of stream events from a specified elementary stream within the Stream component. These events have attributes to represent the event group, the time code for firing and an optional string payload field to define the parameters of the event. This feature is particularly interesting for iTV environments because of the high load of multimedia contents, which are sometimes related to each other. For example, an interactive application can show real time additional information concerning a live event (e.g. sports) or acquire and display subtitles synchronously in a language not supported by the provider. A further and more complex example of that technology is presented in chapter 4 based on a use case for learning environments.

2.3.7. The Return Channel

The DVB-PCF standard provides three main items that enable the return channel definition: (1) the ReturnPath component, (2) the Transaction component and (3) a TransferCollection component.

The ReturnPath component defines the properties of the return path itself and manages the data transaction process between the iTV service and the application server. It includes the connection address of the target server via a URI and the start-up behavior of the return path, which can be opened or closed depending on the channel connection status. The connection target value is an abstract URI to be resolved by a later transcoding step. It is defined as a Uniform Resource Name (URN) [53], which is a URI that identifies a resource without taking into account its specific location. The portability achieved with URN must be resolved in the transcoding process in order to get a unique name of the actual resource depending on the target platform (e.g. a URI for a broadband platform and a telephone number for a dial up capable platform). The Transaction component manages the status of the data transfer processes and generates events regarding the success or failure of the transaction. It embodies a pair of TransferCollection components, for the request and the response, that define the sequence of serialized data to be transferred over the return path. This mechanism enables the straightforward generation of a Rich Internet Application [54] from a DVB-PCF description. Two more DVB-PCF components complete the list of the return channel definition items: (1) the Indicate component, a cut-down version of the ReturnPath component (without transaction process) that only establishes a connection with the server for very simple applications (e.g. voting, counter, etc); and (2) the SecureReturnPath component, for secured data transactions.

2.3.8. Comparison with Other Languages: Justification of DVB-PCF

Other declarative application formats based on XML such as HTML, XHMTL, SVG and WTVML are platform-specific presentation formats that are interpreted by the browser engine. Their description of the user interface must detail all the implementation issues to ensure an accurate visualization. This increases the complexity of the transformations between different application formats, especially for the translation to procedural application formats like MHP and MHEG-5. DVB-PCF describes how the components of an iTV service must be presented, i.e. layout and behavior, without taking into account the implementation details of a given browser engine. UIML, which is somehow similar to DVB-PCF, provides a platform-independent solution for the presentation of multimedia content and, although more tools are available for this format, the accurate management of the time related issues (e.g. synchronization) of DVB-PCF and its specialization in regards to iTV services make it a better candidate as a cross-platform content description.

The major technological drawbacks of the above mentioned formats is that they are text-centric solutions that require the use of scripting in order to synchronize XML-based applications with the audio-visual content and to produce playlists. This is because those formats do not include time-related considerations. Other declarative languages like SMIL and NCL include time information and thus provide a more media-centric solution. However, DVB-PCF has been developed to facilitate the automatic translation into platform specific formats, leading to a significant reduction in cross-platform costs.

In contrast, the controlled delivery of multimedia content to a user’s personal device (e.g. Personal Video Recorder) has been addressed by the set of specifications covered by TV-Anytime. It seeks to exploit the evolution in convenient, high capacity storage of digital information to provide consumers with a highly personalized TV experience. TV-Anytime manages the metadata related to the description of the TV content, such as program title and synopsis. In TV-Anytime program metadata describes the audio-visual content in a formal way and allows the consumer to find, navigate and manage content from a variety of internal and external sources including, enhanced broadcast, interactive TV, Internet and local storage.

The management and personalization of TV contents from a user’s personal device is out of the scope of DVB-PCF, which is focused on defining and producing the interactive contents that are available to users rather than the management of their preferences in the local platform. In other words, DVB-PCF enables the generation of interactive content that can be searched and selected from a user platform with TV-Anytime.

2.4.A Procedural Client Format: MHP

The Multimedia Home Platform (MHP) has been chosen as the example of platform-specific procedural format for describing iTV applications that will be obtained from the transcoding process proposed in this dissertation. Although MHP is being replaced by HbbTV in many

European regions, the portability reached by this standard across different iTV networks justifies the choice. This section details the main features of MHP.

2.4.1. Introduction to the Multimedia Home Platform

MHP was the solution designed by the DVB Project to enable interoperable applications to be downloaded from DVB broadcast networks and executed on receivers from any manufacturer. It is defined as a common API for providing a cross-platform middleware for the interoperability between iTV applications from different providers and specific iTV hardware (i.e. TV sets and STBs).

The first stable version of MHP was finished in 2003 and was known as MHP 1.0.3. [14]. It enables the acquisition and running of special Java-based applications, called DVB-J or Xlets, by a MHP receiver. The applications can be transmitted by the broadcast TV provider through the same channel than audiovisual contents, enabling a standardized access to broadcast iTV applications from any provider. This power was taken into account by some countries to promote this standard among the broadcasters, leading to some countries to subsidize the MHP STB for a broader dissemination. This technology arrived when most countries where switching off the analog television, but the efforts to deploy the first large-scale iTV system were frustrated mainly by a lack of a common regulation by government authorities, the lack of clear market demand for interactive services and the lack of interactive contents that push for the paradigm change. Moreover, other aspects related to the production and deployment costs such as development tools, platform certification and application compatibility may also have influenced in the failure of this technology.

A parallel version of MHP was built by the DVB Project for adding full support to Internet applications, enabling web browsing, the use the IP network as a two-way return channel and the access to DVB-HTML applications, which is the DVB specification for declarative applications that is a subset of HTML using the standards XHTML 1.1, CSS 2.0, Document Object Model (DOM) 2.0 and ECMAScript (the basis for JavaScript). It was firstly published in 2001 with the name MHP 1.1, completed in 2005 as MHP 1.2 and definitively reviewed in 2010 as MHP 1.2.2 [58]. This version enabled also the implementation of MHP on non DVB networks such as IPTV networks and the Open Cable Application Platform (OCAP) [56], the solution adopted by the main US cable providers that is currently known as Tru2way. The interoperability between different networks was achieved through the Globally Executable MHP (GEM) [59] , which is a subset of MHP 1.2 that removes the features that were specifically designed for DVB networks for providing a network independent solution for iTV applications. GEM was standardized by DVB, and adopted by ETSI, the ITU, CableLabs, ARIB, ACAP, and the Blu-ray Disc Association for enabling the access to Java applications on broadcast receivers, IPTV terminals, hybrid solutions and Blu-ray disc players.

The GEM/MHP global architecture is summarized in Figure 5 and shows the network independency of GEM applications across different networks. Receivers usually implement a subset of this architecture in order to enable the access to a specific network or a combination of them, in which case the device is considered a hybrid receiver. For hybrid markets, the most common options are:

- MHP + GEM-IPTV for a combination of satellite, terrestrial or cable plus IPTV in a proprietary network.
- MHP + MHP-IPTV for a combination of satellite, terrestrial or cable plus IPTV in a fully standardized DVB network.
- Tru2way (OCAP) + GEM-IPTV for a combination of US cable plus IPTV in a proprietary network.

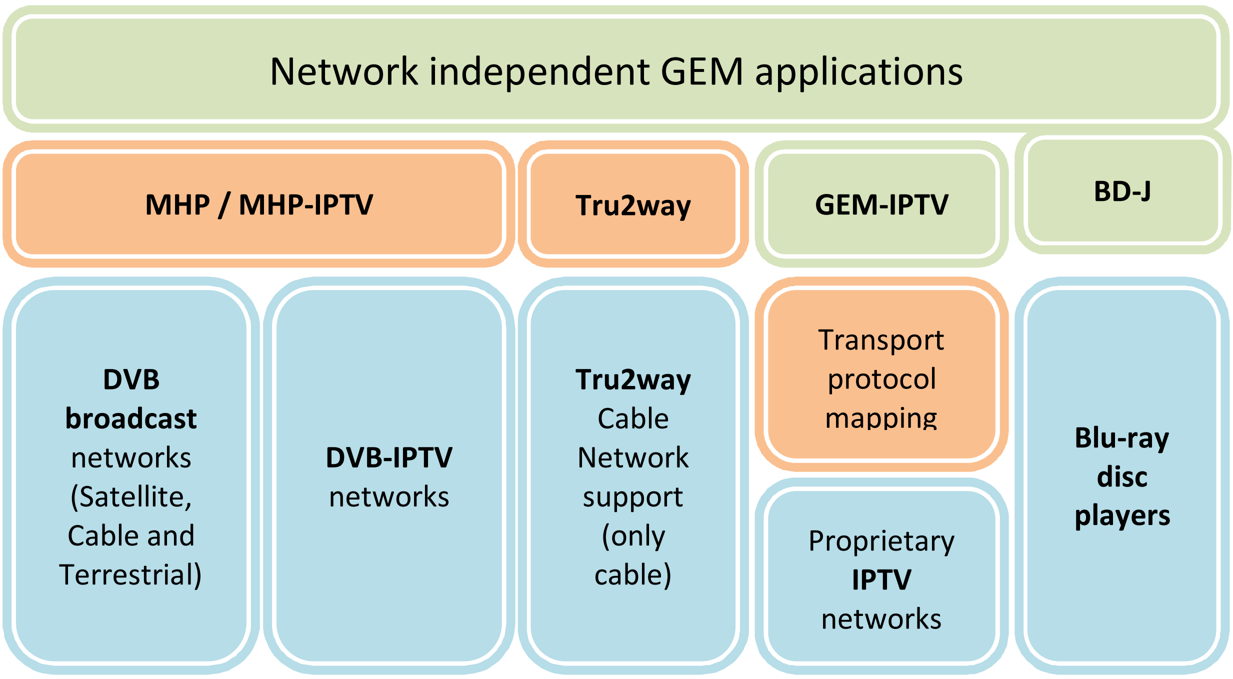

Network independent GEM applications
MHP / MHP-IPTV
Tru2way
GEM-IPTV
BD-J
DVB broadcast networks (Satellite, Cable and Terrestrial)
DVB-IPTV networks
Tru2way Cable Network support (only cable)
Transport protocol mapping
Proprietary IPTV networks
Blu-ray disc players

Figure 5. The GEM/MHP overall architecture.

As shown in Figure 5 and in addition to hybrid receivers, the architecture of GEM/MHP also provides support for GEM applications on proprietary or non-standardized IPTV networks through the GEM-IPTV API and a mapping of the transport protocol (e.g. RTSP, IGMP, UDP) between the proprietary network and the network-independent GEM application. Moreover, GEM/MHP provides support for interactive applications on Blu-ray discs via the standard specification Blu-ray Disc Java (BD-J), which gives support for Java ME Xlets on Blu-ray players.

2.4.2. Applications and Services

The standard MHP defines a complete platform for accessing to the most common iTV services, but typically these services can be directly obtained from a third-party content provider (such as broadcast programs, internet content etc) as well as through the applications offered by the iTV platform provider (pre-installed applications, supervised internet browsing etc). Table 1 outlines a generic classification for MHP services according to the level of interactivity provided by the service and based on the research work conducted by [60]. This classification can be also applied to most of the existing iTV platforms.

Platform Services	Content Services
Navigation Services	Autonomous Interactive Services
Platform-Specific Services	Program Associated Interactive Services
	Interactive Audiovisual Programs

Table 1. Generic classification of interactive services for MHP platforms.

Platform Services are those services offered by the platform provider and can be divided into Navigation Services and Platform Provider Services. On the one hand, Navigation Services enable the browsing and navigation across the different services available in the platform. This can of service is usually known as Electronic Program Guides (EPGs) and provide users of television, radio, and other media applications with continuously updated menus displaying broadcast programming or scheduling information for current and upcoming programming. On the other hand, Platform-Specific Services are those services that are preinstalled in the platform for enhancing the user experience of the system. This set of services can vary from one platform to another, but are usually related to messaging, chats or a supervised internet browsing. This browsing is restricted to the set of contents that the platform provider has defined previously, so that it is not a universal internet browsing but a walled garden in which users are limited to the not filtered content.

Content Services are the set of services that are supported by the MHP platform and accessed via the network available for the platform (broadcast, IPTV, internet, cable etc). They can be divided into Autonomous Interactive Services, Program Associated Interactive Services and Interactive Audiovisual Programs. This can of services are usually available for multiple iTV platforms, not only for MHP, but require a specific-platform application for accessing to them.

Autonomous Interactive Services are audiovisual products that are totally independent from the program being watched. They are the main set of interactive services that are available in any iTV platform due to the reduced development facilities and costs that they require in comparison with the other kinds of Content Services. Most of the main MHP applications fall into this group and the present research has been also focused here. The services of this group are mainly related enhanced information, participation, eGovernment, education, eCommerce, telebanking, games and bets among others.

Program Associated Interactive Services are iTV services that are linked and synchronized to a TV program in order to complement and enhance its user experience. These services are only available during the program timeline and can present information according to the program events. Typical examples for this group of services are the enhanced information in sports events, the participation in TV shows and ecommerce.

Finally, Interactive Audiovisual Programs defines TV programs that are specifically designed to be modified according to viewer actions such as video-on-demand catalogs, camera angle selection, multi-stream programs that enable content selection inside the program and lineal programs that can vary its sequence and are user voting dependent. By the nature of these services, they are out of the scope of the present research, which is focused on the development of interactive applications rather than interactive TV programs.

2.4.3. Platform Profiles

MHP defines 4 profiles for clustering the features that must be supported by a receiver to reach a certain level of interactivity. MHP 1.0 specifies only the first and the second profile. The third profile was introduced with MHP 1.1 and the fourth profile with MHP 1.2. These are defined as follows:

- <u>Enhanced Broadcast Profile (Profile 1)</u>: This is the most basic profile of MHP. It allows the deployment of local interactivity applications without a connected return channel. Applications may be downloaded from a broadcast stream (i.e. MPEG-2 Transport Stream), enabling services such as Electronic Program Guides (EPGs), information services and news tickers (i.e. small screen spaces on news TV programs dedicated to headlines or minor pieces of news).
- <u>Interactive Broadcast Profile (Profile 2)</u>: This profile enables a two-way communication with the iTV service provider through the return channel. It allows the participation of the audience in TV programs via applications and other services like VOD, pay-per-view, T-commerce (i.e. electronic commerce on TV) and televoting.
- <u>Internet Access Profile (Profile 3)</u>: This profile adds the support for Internet applications by defining the suitable APIs for connecting to the most common Internet services. This enables applications such as email, chat, online games, social networks, etc.
- <u>IPTV Profile (Profile 4)</u>: This profile, which is included in the specification MHP 1.2, adds support for proprietary broadband IP networks by integrating support for DVB-IPTV. It is a standard interface between broadcast networks, IPTV networks and Internet.

2.4.4. Deployment Status

Nowadays, the largest developments of MHP are in Italy (terrestrial), Korea (satellite), Belgium (cable) and Poland (satellite) while Germany, Spain, Finland, Greece, Austria, Colombia, Uruguay and Australia have smaller deployments or significant trials. However, some of them like Spain, Finland and Greece fail in the deployment of MHP STBs and most of the target homes do not have a MHP receiver, which have provoked the failure of the implementation of this technology.

The evolution of MHP to GEM has also enabled the support for MHP services on different platforms such as Tru2way for cable networks on the United States, Ginga for the Japanese-Brazilian iTV network and HbbTV, wich aims to replace MHP in many European regions.

2.5. A Declarative Client Format: HTML5 and New Cross-Platform Web Technologies

HTML5 is the declarative format for describing iTV applications chosen for testing the methodology proposed in this research concerning the development of cross-platform iTV applications. This section details the main features of this technology.

2.5.1. Introduction to HTML5 as a Set of Standards

The HyperText Markup Language version 5 (HTML5) is the latest revision of the HTML standard, which was created in 1990 and standardized as HTML4 in 1997. HTML5 does not belong to any company or organization and is not attached to a particular browser but it is pushed by the main players involved in the evolution of the Web such as Google, Microsoft, Apple, Mozilla, Facebook, IBM, HP, Adobe and many others. It is intended to be the natural evolution of the web by the agreement of the industry and unifies the application development for multiple platforms and devices. No other technology has achieved ever the ubiquity that HTML5 is able to reach.

Not only includes the HTML specification but also unifies for the first time HTML and XHTML. It actually comprises a collection of standards, technologies and APIs for the development of web applications, desktop applications, and mobile applications as well as applications for any kind of device that provides the suitable support.

HTML5 also includes the specifications for DOM and JavaScript, standardizing the functions the browser must provide for managing the behavior of the web application. Moreover, it also provides support for CCS3, SVG, SMIL and other web-related technologies.

HTML5 enhances significantly user experience, and the interactivity and connectivity of the traditional web applications by adding new important features that are oriented to bring the power of desktop applications to the web-based applications.

HTML5 is a technology for applications and a push for innovation. It enables web applications to reach similar functionality, speed, performance, and user experience than native applications. It is widely supported by multiple platforms and devices through a browser and, as the browser is frequently updated in most platforms, the support for new HTML5 features reaches most platforms at the same time.

HTML5 is a technology for users and a push for smart environments. Applications do not need to be specifically installed nor updated, avoiding duplicates and extra version managements. As the application is not attached to a particular platform, it can follow the user anywhere anytime for a more immersive multimedia experience.

2.5.2. Why HTML5?

HTML5 introduces many interesting features for creating web-based applications with similar functionality than native applications for any multimedia platform (desktop, mobile, TV, etc). But unlike platform-specific applications, the portability achieved by HTML5 across different platforms has not a direct competitor among the current technology for programming interactive applications. Next lines describe the innovative features of HTML5 for developing applications.

Multimedia and graphics

Before HTML5, sophisticated applications with fast animations, fast user response, video management or complex visual effects, such as games, had to be implemented either as native applications for specific platforms, or via browser plugins since web applications could not reach these features by themselves. HTML5 enables the development of games, complex animations and audiovisual content management through the integration of related standards such as CSS3 for presentation enhancements, 3D CSS and WebGL for 3D graphics and interfaces, a canvas component and SVG for vector graphics, and rich APIs for high performance management of audio and video contents. All these features require a commitment between browser and the platform in order to enable applications to run fast. Modern browsers implement hardware-accelerated rendering through the Graphics Processing Unit (GPU), which enables high-performance and real time graphics for games and other applications with hard time constraints. Moreover, JavaScript engines are currently sufficiently fast to provide the suitable support to these features.

Local storage and offline applications for higher performance

Web applications and the offline world finally converge with HTML5 to create applications that can run without connection or enhance its performance when connected by defining proper local cache policies that reduce the number of connections to the service provider and the number of page reloads. This also indirectly enables anytime, anywhere applications that can follow the user regardless of the connectivity mode (e.g. travelling by plane, going to not connected places and reduce the roaming costs). HTML5 adds the application cache and local storage interfaces for the storage of applications and data respectively, the session storage interface for sessions, IndexedDB for database structures, and the file system interface for accessing to the platform native file system.

Offline APIs improve the application performance by enabling a fast access to locally stored information and reducing the number of requests to the server. The client-server model that has been traditionally applied to web applications is based on the assumption that clients are low-profile devices that require that web servers take most of the load of building the pages requested by the user. Nowadays, the advances on processors and memories have generated high performance client devices, which far exceed this assumption even for mobile devices. Moreover, the increasing number of connected devices has led to an overhead in servers and networks (e.g. public WiFi networks supporting smart mobile devices).

In this context, higher application performance can be achieved with HTML5 by storing locally the application files and both static and dynamic application data. As an implementation example, this a generic HTML5 cache policy for any dynamic request:

1. User starts a dynamic request to the server (e.g. top ten highlighted products in a shop).
2. The request is received and presented in the client device.
3. The request data is stored locally as an object that includes the request string, the request result and the time of the request.
4. User starts a new dynamic request to the server.
5. The application searches locally if the new request string is stored.
6. If the request string is found locally, the application checks the request time and determines if the results are still valid according to the maximum allowed time, which depends on the estimated time that the results are considered consistent (e.g. the shop of the example updates the products every 24 hours and the common time that a user spend in the shop is 20 minutes per connection, so the request results can be consistent for at least 30 minutes. If a user starts the same request in less than 30 minutes, results are obtained from the local storage instead of connecting to the server).
7. If the request string is not found or the request time exceeds the maximum allowed, the request is addressed to the server.

The improvement in the traditional client-server paradigm achieved by HTML5 for web-based applications is based on reducing server request to the transfer of essential data between clients and servers.

Simpler development for wider deployments

HTML5 enables developers to build applications for a larger number of devices and reduces the fragmentation across browsers, platforms, and devices. The code portability achieved by HTML5 allows developers to build applications for multiple platforms without having to learn neither new programming languages nor specific Software Development Kits (SDKs), which is a typical time consuming task for targeting multi-platform applications. The case of mobile applications is especially significant. Mobile developers use to address their applications to the most important mobile platforms (i.e. Android, iOS and Windows Phone), but they forget the rest of smartphone users and the amount of feature phones that cannot access to the application markets but can connect to the Web via web browsers and download some Java applications. A market study concludes that feature phone sales will grow up 115.4 percent from 2010 through 2015 compared to only 15.8 percent growth for mid-range and high-end smartphones due to the expected growth in emerging markets around the world [61]. This implies a dramatic number of devices that many application developers are leaving by the wayside advisedly.

Security

HTML5 improves the system security when accessing applications through the browser because many of the HTML features that traditionally required additional plugins are now implemented in modern browsers natively, which reduces the risk of installing untrusted software. The standardization of the parsing algorithm eliminates browser discrepancies that hindered the application interoperability before HTML5 and many of the new APIs had been designed to be safer and ask for the suitable permissions to the user. For example, the geolocation API requests permission before obtaining user location information.

Modern browsers are also designed to be safer with the addition of a number of security features such as sandboxing, multi-process architecture, new Hypertext Transfer Protocol (HTTP) headers support, and new security policies.

The sandbox model, which is not new for HTML5, prevents applications to access directly to the local system resources or connecting to external web applications on different domains. These functions must be implemented via the browser, which supervises the security of the process and minimizes the risk of installing malware in the background. For instance, local file access can be performed through the File System API provided by the browser, but never directly from the application.

The multi-process architecture of the browser enables the tab management for presenting web applications that are enclosed into its own rendering engine, memory and system process. This isolation, which was an innovation provided by Google Chrome, not only prevents the malware infection between different processes but also the potential spread of the instability generated by one particular web application to the rest of application threads.

The development advances of the HTTP, the web data communication protocol, include new security features for preventing cross-site attacks and unwanted connections to untrusted networks, especially from iframe elements in HTML.

Modern browsers also include extra security policies for user protection by maintaining blacklists of suspicious domains or applications, defining content profiles for stating the resources that an application may use and filter Cross-Site Scripting among others.

These layers of security are not present in desktop applications, where the access to local resources is not limited to an application environment such as a particular tab in a modern browser.

Costs and maintenance

As HTML5 applications work on multiple platforms regardless of device-specific features, their maintenance is easier, the related costs are reduced, and the overall productivity is improved.

Web applications do not need to be neither installed nor updated; they are simply one click away for the user. Updates can be applied on multiple devices at the same time through an IP connection and the access to updated data in a remote server. Moreover, when a device needs to be changed, applications do not need to be reinstalled. From users' point of view, application versions become irrelevant and, from developers' point of view, devices become irrelevant (at least it should be).

Presentation and user interaction

The interaction with a web application, which was typically limited to mouse and keyboard events, has also been improved by the new APIs defined by HTML5. Drag and drop, geolocation, device orientation and touch events are now possible, and the input from external devices such as webcams, microphones and USB devices will be available in a close future.

The power of the Cascading Style Sheets specification version 3 (CSS3) enables amazing visual effects in web applications such as rounded corners, shadows, 3D effects and animations without the need of further scripts. This enables a better specialization of developers because they do not need to know both JavaScript and CSS.

Real-time communication

HTML5 provides an easy and standard way of communicating applications with real-time constraints.

WebSockets enable real-time communication not only between the client application and the server but also between different clients to share the screen, audio and video conferencing and play multiplayer games.

WebRTC enables native support for video and audio conferencing and live streaming.

2.5.3. Scalable Vector Graphics

The Scalable Vector Graphics (SVG) can be seen as a programming language for two-dimensional vector graphics with the particularity that is not a language but a XML description. The SVG specification is an open standard that has been under development by the World Wide Web Consortium (W3C) since 1999. It enables both static and dynamic graphics, including interactive and animated content. The key benefit of being a XML description is that contents can be searched, referenced and scripted, which makes SVG content accessible from the web (e.g., one can search SVG maps on internet using any of the main search engines). This feature makes SVG the perfect standard for providing vector graphics on the web and HTML5 has incorporated SVG as part of the HTML5 standard family, which means that now SVG can be embedded on HTML documents in addition to be supported directly by the browser as a SVG document.

SVG, as a support for vector graphics, has some advantages over bitmap graphics (such as JPEG, GIF and PNG):

- Generated files are usually much smaller than bitmaps, resulting in faster downloads.
- Graphics can be scaled to fit different display sizes without pixelation.
- Graphics are constructed within the browser, reducing the server load and response time.

- The end-user can interact with and change the graphics without need for complex and costly client-server communications.
- It provides native support for SMIL (Synchronized Media Integration Language), which adds the notion of timing to HTML elements for presenting complex time-based animations without requiring JavaScript or any scripting.
- It can interact with JavaScript and coexist with HTML documents either in separated files and embedded into HTML5 documents.

Moreover, as a XML document, SVG has some other benefits:

- Its contents are fully accessible from the web, while the bitmap description relies on the additional metadata that is provided by or extracted from the image itself.
- The code tends to fit into the standards for defining how it should be written and how client software should parse it.
- It is written in text and is generally human-readable (non expert developers can edit it).
- The way that JavaScript can manipulate both the SVG elements and the Document Object Model (DOM) is quite similar to how is used with HTML. Therefore, the SVG learning curve for web developers is pretty soft.

2.5.4. Synchronized Multimedia Integration Language

The Synchronized Multimedia Integration Language (SMIL) is the W3C recommended standard for describing multimedia presentations. It defines the suitable markup for timing, layout, animations, visual transformations and media embedding for presenting text, images, video, audio and links among other media elements. It is a separated language from SVG, but closely integrated with this standard for performing vector graphics animations.

SMIL is the way to create SVG animations through a declarative notation (related to declarative programming) rather than specifying the details of how to do something as in an imperative language (related to procedural programming). A declarative approach defines what the end result is supposed to be leaving detail implementation for the rendering engine. For instance, in SVG an element <circle> can be simply declared without specifying how a circle must be drawn on the screen. This is the same for HTML and PostScript.

A declarative animation for SVG based on SMIL is usually compared with a JavaScript animation, which is the alternative method for animating SVG elements on web-based applications. As an example, Table 2 presents a side-by-side comparison of a SVG animation implemented with SMIL and the corresponding JavaScript approach. In the example, an ellipse is animated to oscillate between two different sizes for rx over a period of 4 seconds. The first evidence provided by the comparison is that the SMIL code is noticeably shorter and easier to read, which means that there is less code to maintain, and its maintenance is simpler. It has to be noted that the JavaScript approach could accept some optimizations to reduce the number of lines, but the main propose of the example is to show how a declarative approach in SMIL that only states the resulting behavior can be more desirable in terms of scalability and maintenance than a procedural approach in JavaScript.

SVG + SMIL animation	SVG + JavaScript animation

`<svg`
`xmlns="http://www.w3.org/2000/svg">`
`  <ellipse cx="150" cy="75" rx="10"`
`    ry="40" fill="blue">`
`   <animate attributeName="rx"`
`dur="4s"`
`     values="10;110;10"`
`     repeatCount="indefinite"/>`
`  </ellipse>`
`</svg>`
	`<svg`
	`xmlns="http://www.w3.org/2000/svg"`
	` onload="startup(evt)">`
	`  <script>`
	`  <![CDATA[`
	`  var`
	`xmlns="http://www.w3.org/2000/svg"`
	`  var E;`
	`  function startup(evt){`
	`   E=document.getElementById("E");`
	`   move(10,110,10,1);`
	`  }`
	`  function move(thin,wide,curx,dir){`
	`   E.setAttributeNS(null,"rx",curx);`
	`   curx+=dir;`
	`   if (curx>wide||curx<thin ) {`
	`    s="move("`
	`window.settimeout(s,10);`
	`    dir=-dir;`
	`   }`
	`  }`
	`  </script>`
	`  <ellipse id="E" cx="150" cy="75"`
	`    rx="10" ry="40" fill="blue"/>`
	`</svg>`

Table 2. SVG animation for an oscillating ellipse in SMIL and JavaScript.

An interesting feature of the SMIL animation is the integration of the animation with the animated element. Looking at the example, one can realize that the <animation> element is nested directly within the <ellipse> element, which generates a clean and well-structured code. In contrast, the JavaScript code is based on procedures that are written separately from the animated element, which is referenced in the procedures using the element ID.

In contrast, it has to be noted that the flexibility that enables JavaScript as a procedural approach cannot be achieved by SMIL and that not all JavaScript animations can be performed with SMIL. However, both SMIL and JavaScript-based animations can coexist in SVG, hence developers have both sets of capabilities at their disposal to choose the one which fits better with the target set.

2.5.5. Analysis of the Current Browser Support

Although standardization process of HTML5 is still in progress, and it is expected to be so for the following years (new features are continuously discussed), the key features have been defined and published for its standard use. Currently, main browsers are providing support for many of these features and every new browser released adds new ones. However, each browser has its own implementation plan (and business strategy) for the feature support, and developers must be cautious in implementing these new features in cross-browser applications because the application portability may be compromised by lack of support of a particular feature. Fortunately, there are some important features such as the audio and video HTML elements that are currently supported by all the main browsers, allowing their widespread use within web applications.

Table 3 and Table 4 summarize the current support of the main browsers for the key features of HTML5 and SVG respectively. Browsers analyzed cover around 91% of the worldwide browser users and the comparison is done for the current (first column) and the forthcoming (second column) versions of each browser. The table colors indicate if a particular feature is supported (green), partially supported (orange) or not supported (red). A white box indicates that the support for a particular feature is unknown in forthcoming browser releases. Finally, the support for all selected features is averaged in last row with a simple pattern: 100% when supported; 50% when partially supported (although the real weigh can vary); and 0% when not supported or its support is unknown.

Although the comparison tables are not comprehensive of all HTML5 features, the results evidence that all browsers are increasingly implementing HTML5 features in each new update, which reveals a clear interest in this technology. Moreover, there is an agreement for the support of some key features across all browsers such as audio and video elements, contenteditable attribute, new semantic elements and canvas. However, the most remarkable point is that, while HTML5 is still improving and standardizing parts of its specification, the main actors involved in the web development are improving its support and pushing for the innovation of the web through HTML5.

Regarding SVG as part of the HTML5 standard family, its basic support is ensured in most used browsers, either inline within a HTML5 document or linked as a SVG image element. SVG effects, filters and fonts are differently supported and SMIL animations are supported in all major browsers with the exception of Microsoft’s Internet explorer, which has not been updated in this regard from the previous version and still requires a third party library or plugin to deploy SMIL animations.

HTML5 feature	IE		Firefox		Chrome		Safari		Opera		iOS Safari		Android	
	9.0	10.0	13.0	14.0	19.0	20.0	5.1	5.2	11.6	12.0	5.0		4.0	
Audio element	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
Canvas (basic support)	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
Color input type	🟥		🟥		🟥	🟩	🟥		🟩	🟩	🟥		🟥	
contenteditable attribute (basic support)	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
Datalist element	🟥	🟩	🟩	🟩	🟥	🟩	🟥		🟩	🟩	🟥		🟥	
dataset & data-* attributes	🟧	🟧	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
Date/time input types	🟥		🟥		🟥	🟥	🟥		🟩	🟩	🟩		🟥	
Details & Summary elements	🟥		🟥		🟩	🟩	🟥	🟩	🟥		🟥		🟩	
Drag & Drop	🟧	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟥	🟩	🟥		🟥	
Form validation	🟥	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟩	🟩	🟥		🟥	
HTML5 form features	🟥	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟩	🟩	🟩		🟥	
New semantic elements	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
Number input type	🟥	🟩	🟥	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧		🟧	
Offline web applications	🟥	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
Progress & Meter	🟥	🟧	🟧	🟧	🟩	🟩	🟥	🟩	🟩	🟩	🟥		🟥	
Range input type	🟥	🟩	🟥		🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟧	
Ruby annotation	🟩	🟩	🟥		🟩	🟩	🟩	🟩	🟥		🟩		🟩	
sandbox attribute for iframes	🟥	🟩	🟥		🟩	🟩	🟩	🟩	🟥		🟩		🟩	
Session history management	🟥	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟩	🟩	🟩		🟥	
Text API for canvas	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
Toolbar/context menu	🟥		🟩	🟩	🟥		🟥		🟥		🟥		🟥	
Video element	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
WebGL – 3D canvas graphics	🟥		🟧	🟧	🟩	🟩	🟧	🟧	🟥	🟧	🟥		🟥	
Averaged support (%)	34,8	71,7	61,3	65,9	80,4	89,1	65,2	73,9	73,9	80,4	63,0		54,5	

Table 3. Browser support for the key features of HTML5.

SVG feature in HTML5	IE		Firefox		Chrome		Safari		Opera		iOS Safari		Android	
Inline SVG in HTML5	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
SVG (basic support)	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
SVG effects for HTML5	🟧	🟧	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧		🟥	
SVG filters	🟥	🟩	🟩	🟩	🟩	🟩	🟥	🟩	🟩	🟩	🟥		🟥	
SVG fonts	🟥		🟥	🟥	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
SVG in CSS backgrounds	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧		🟩	
SVG in HTML5 img element	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
SVG SMIL animation	🟥		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	
Averaged support (%)	56,2	68,7	87,5	87,5	93,7	93,7	81,2	93,7	93,7	93,7	75,0		75,0	

Table 4. Browser support for the key features of SVG in HTML5.

2.6.Summary of the Chapter

The chapter has introduced the key theoretical concepts that are needed for a suitable understanding of this research.

Interactive TV (iTV), understood as the technologies for the user interaction on TV systems, has been evolving from more than 40 years but the related platforms and services have typically had difficulties for meeting the expected success. The diversity of platforms has generated a number of application formats and runtime environments that hinder application portability across different TV systems.

In order to address this problem, the consortium for the Digital Video Broadcasting (DVB) published the standard DVB-PCF as a portable content format for iTV applications that follows the "write once, adapt to everywhere" paradigm. Section 2.3 has introduced the format and the key details for the development of DVB-PCF descriptions.

The Multimedia Home Platform (MHP) and HTML5 been introduced in Sections 2.4 and 2.5 respectively as examples of different models of platform-specific formats for iTV applications.

CHAPTER 3

GENERATING CROSS-PLATFORM APPLICATIONS FOR INTERACTIVE TV

"Roads? Where we're going,
we don't need roads."

Dr. Emmett Brown ("Back to the future", 1985)

This chapter presents the proposals and contributions driven in this research, which has been focused on the cross-platform design of interactive applications for TV environments. The main proposal describes the methodology for building and maintaining cross-platform applications based on the standard DVB-PCF. The implementation of this methodology has generated a software framework and an integrated development environment. These tools validate the proposed methodology and enable further experiments in this area. Finally, the exploration of suitable techniques for implementing multimedia synchronization in web environments has generated a novel proposal for synchronizing video subtitles on the client side.

3.1. Methodology for Building Cross-Platform iTV Applications

This section presents the main contribution of this research, which is focused on the generation and maintenance of cross-platform iTV applications. The proposed methodology is based on:

- The abstraction of the programming language to reach a high level of platform independency with minimum translation costs in platform-specific adaptations.
- The implementation of a progressive enhancement strategy to adapt the generated applications to the variety of platform resources.

The application of this methodology enables the portability of iTV applications across multiple runtime environments and programming languages. Figure 6 outlines the addressed problem.

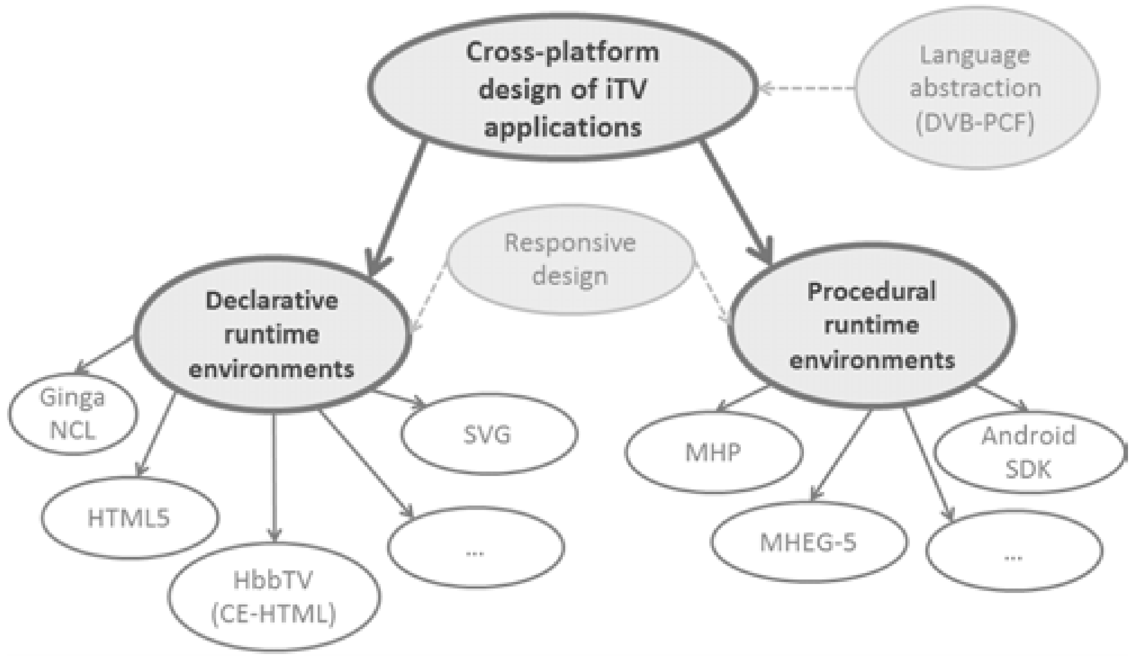

Cross-platform design of iTV applications
Language abstraction (DVB-PCF)
Responsive design
Declarative runtime environments
Procedural runtime environments
Ginga NCL
HTML5
HbbTV (CE-HTML)
SVG
...
MHP
MHEG-5
Android SDK
...

Figure 6. Map of iTV application formats and runtime environments.

It has to be noted that this cross-platform methodology is not a "write-once, run anywhere" strategy but a "write once, adapt to anywhere". Even so, the particularities of each individual runtime environment may imply further platform-specific adaptations that cannot be covered by the current implementation of present methodology. This work is focused on describing the overall transformation process rather than dealing with specific details of each platform.

3.1.1. Methodology Description

The diversity of runtime environments not only implies a change in the instruction sets, but also a different programming model. The translation from one specific iTV application format to another could be considered as a form of short term solution; however this approach is usually avoided due to the underlying data and programmatic model differences between the targeted formats [62]. This kind of direct mapping solution between interactive application formats can provide an explicit mapping from one format to another but reduces the flexibility of the system by having to implement a cloud of mapping functions to perform the translation from one format to another. In this case, the number of mapping functions increases exponentially with the number of formats and requires the implementation of conversions when considering different formats. A different approach is to integrate all interactive application formats into a new single format which satisfies the requirements of all relevant platforms. This means an abstraction of the programming language and provides an independent way to minimize the total number of mapping functions (particularly, in the case of three or more interactive content formats), which increases linearly with the number of formats. Figure 7 depicts the basic schemas for both mapping models.

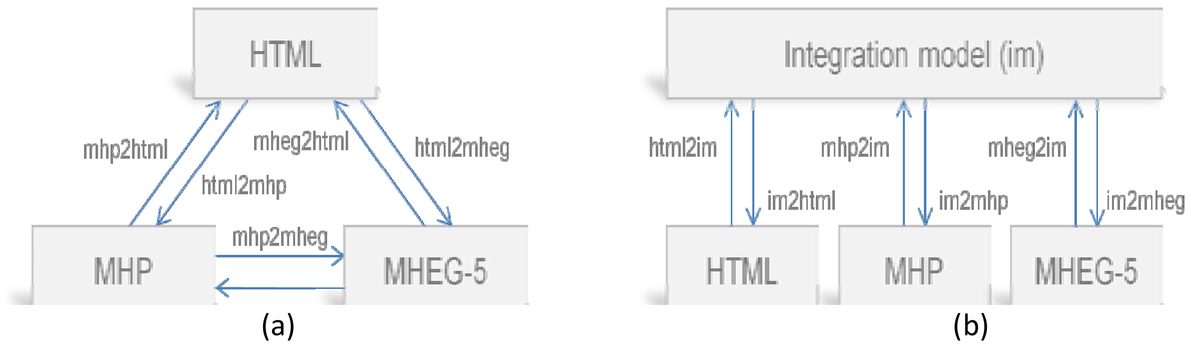


Figure 7. Mapping models for the translation between application formats. In (a), formats are directly translated across formats and, in (b), a single higher level format that provides better scalability and flexibility.

In order to reach a high level of platform independence, we selected the DVB-PCF standard as the reference integration model for a unified design of interactive applications in TV environments. It can be easily translated into any other interactive application format in a platform-specific conversion step that is also called “transcoding”. DVB-PCF facilitates that conversion avoiding the implementation complexities of each separate platform and tries to represent the desired user experience of the interactive application. In other words, this content format tends to focus on defining user perception and responses that result from the use of an interactive application rather than defining the implementation issues in detail.

The translation into the platform-specific format is generally done by the service provider, who usually knows the required configuration for the deployment. However, the DVB-PCF description of an iTV service can also be pre-translated by the application provider if it knows the details for the application deployment (target platform, target runtime environment, etc). In the case of the conversion into a web-based format this can be achieved through the generation of Rich Internet Applications and a simple XML configuration file with the deployment details.

The DVB-PCF standard recommends that the suitable scaling algorithms for each specific target platform must be implemented in the transcoding step. However, it does not define the mapping rules for the scenario where the resolution of the display differs from the reference screen or indeed when the interactivity model changes because of the different usability requirements for a specific device (e.g. the use of the remote control for a TV set, a mouse for a computer, and a touch screen for a smartphone). These adaptations must also be taken into account in the transcoding step in order to generate universal applications for any target device without modifying (or at least keeping modifications to a minimum) the usability and functionality of the designed interactive application.

Typically, the main actors involved in the production of iTV services can be classified into three categories: (1) audiovisual content providers, (2) interactive application providers and (3) the service provider. Audiovisual content providers are considered those that generate and provide the audiovisual material that is supposed to be the main thread of a TV program. This is, for instance, the role of a news service that provides different recorded (or even live) content to many TV channels. The interactive application providers design and develop interactive applications for deployment along with the related audiovisual content. Third-party development companies at the request of the TV service providers have traditionally carried out the implementation of these interactive applications. Currently, the rise of Internet-connected platforms and the emergence of online application stores (Google Play, Apple App Store, Samsung Apps, etc) enables an easy deployment of iTV applications by third-party developers. Finally, the service provider is the actor who takes all the content related to a TV program (i.e. the audiovisual content plus the related interactive applications), deploys it on a particular distribution network (i.e. terrestrial, satellite, Internet, etc) and makes it accessible to the users. This category includes the TV operators for terrestrial, cable and satellite as well as the IPTV providers and the Web sites that provide Web TV contents via the Internet. Sometimes the three roles involved in the production and deployment of iTV services are centralized in one company that controls the whole process, but usually there are different entities involved that must cooperate in order to effectively deploy an interactive service.

Although most iTV services are based on standardized middleware platforms, the content producer and the application provider usually need some knowledge of the platform to apply their contents properly. Producing iTV services in a platform-independent way through an integration model for the description of the interactive applications also implies high levels of cooperation between the parties involved in the service deployment. The adaptation of an interactive application, from the integration model to the specific runtime environment of the access platform, requires a platform-dependent step that can be assumed by the service provider as well as the interactive application provider. The service provider usually has enough information about the target platform and as such one of the key objectives is to design, build and deploy an interactive application that is not subject to reworking for different platforms. In addition, the service provider generally wants to maintain control of the viewer experience in the deployed application post-translation, so it can verify the requirements of their own customers. For example, the exact position on the screen of an advertisement can be a key consideration for an advertiser based on their own market studies and it must not be modified. Figure 8 shows a basic scheme of the adaptation options for an interactive application that are defined by means of an integration model with two possible scenarios: (1) the application provider gives the integration model of the interactive application to the service provider, who adapts this application code to the specific format required by the access platform to present the interactive content, i.e. the target application format; and (2) the application provider implements the adaptation from the integrated model and supplies it to the service provider as deployable iTV application.

First scenario would be desirable. It assumes that the integration model behaves like a real business-to-business interchange format and the service provider has the suitable tools to transform it into its runtime environment. However, this requires an industry agreement not reached to date.

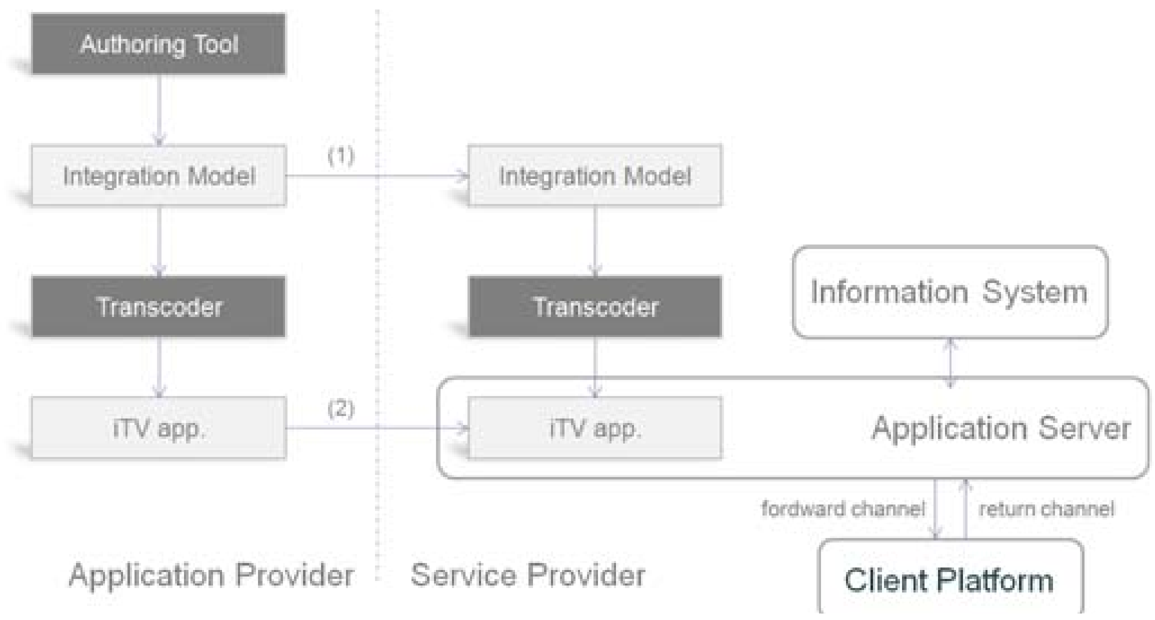


Figure 8. Scenarios for production and deployment of cross-platform iTV services.

The success of proprietary solutions may hinder the adoption of open and shared standards for the production of iTV services, even so, an application provider can benefit from the second scenario described above. This “transcoding” represents the direct digital-to-digital conversion of one code to another; and the “transcoder” is the module that is responsible for this. The configuration parameters necessary to deploy the interactive service must be sufficiently configurable to facilitate the application interchange and the adaptation to the target iTV platform.

Once a DVB-PCF description is created or acquired, it must be translated into the format supported by the target platform. This transformation is slightly different for declarative and procedural application formats due to their programmatic models, but the required steps to implement a transcoder can be basically the same for both.

Although PCF supports pixel-accurate positioning of the visible components as well as flowed layout according to an automated layout algorithm, our work has been centered on the pixel-accurate positioning and assumes that the service author knows the screen resolution of the target platform renderer.

In order to translate the DVB-PCF components, an analysis is required to identify the components which can be directly adapted to the platform-dependent middleware language. Those components that cannot be directly adapted will form complex objects, i.e. widgets, in the final application. Usually, those widgets can be generated using different programming strategies in the target language and those strategies depend on the available capabilities of the target environment. Hence: what is the best strategy to translate a DVB-PCF component to a specific platform format? The proposal of this research suggests to follow a Progressive Enhancement strategy.

The main idea of Progressive Enhancement is that an application must be designed using the lowest common denominator of target platform functionality, and then the designer adds enhancements to the presentation and behavior of the page according to the available resources of each platform. Although this strategy was stated for the design of web applications and the browser interoperability problem, the key idea can be extrapolated to multiple programming environments, especially to the user application design.

The Progressive Enhancement strategy is applied in the present methodology to generate the platform-specific widgets into which those DVB-PCF components (with no direct translation into the target platform format) are translated. That translation must implement the design principles of the Progressive Enhancement automatically by taking into account the most basic profile of a target platform and translating the DVB-PCF specification to meet these requirements. This optimizes the portability of the generated application across the multiple variants or profiles for the same target format. In addition to the basic profile, the translation must implement optional enhancements that are applied according to the detected features of the target platform. It means that the generated application is able to detect the significant features of the runtime environment where is placed and adapt to them. Interactive applications using a Progressive Enhancement strategy become smart applications for smart environments.

As an example, a DVB-PCF service can be translated into a web-based application taking into account a XHTML profile for low profile mobile phones, into a HTML5 profile for Smart TV platforms and into CE-HTML for HbbTV compliant platforms. If the DVB-PCF element to be translated is just a Rectangle, the transcoder can transform it into a simple HTML <div> element with the suitable CSS style for low profile HTML specifications. However, if the platform supports HTML5 and the Canvas element, the application can detect it and draw the same rectangle using Canvas for improving the application performance and possibly the user experience of the drawn scene.

This concept can be extrapolated to MHP applications. When designing an application that requires the return channel, for example, this can be designed for updating information from an Internet server when an IP connection is available, or to behave as a standalone application if no remote connections are available at all. The idea is to provide maximum functionality for each target platform without generating errors that can upset the user.

Once a transcoder is implemented for a set of target platforms that share the same application format, corresponding iTV applications written in DVB-PCF can be directly translated into the target format applying the implemented transcoding process.

3.1.2. System Architecture for Production

A transcoder tool is essential for translating a cross-platform application into a platform-specific format. But what is the relationship between the transcoder and the other key modules of an iTV production chain? What is the position of the authoring tool, the information system and the application repository? How can an application provider manage different layouts and apply it correctly according to the target platform?

This section presents a complete architecture for generating and maintaining cross-platform iTV services based on the integration model for abstracting the programming format. Figure 9 outlines this architecture at system and sub-system level, which consists of (1) Service Authoring, (2) Data Management, (3) Layout Management and (4) Content creation.

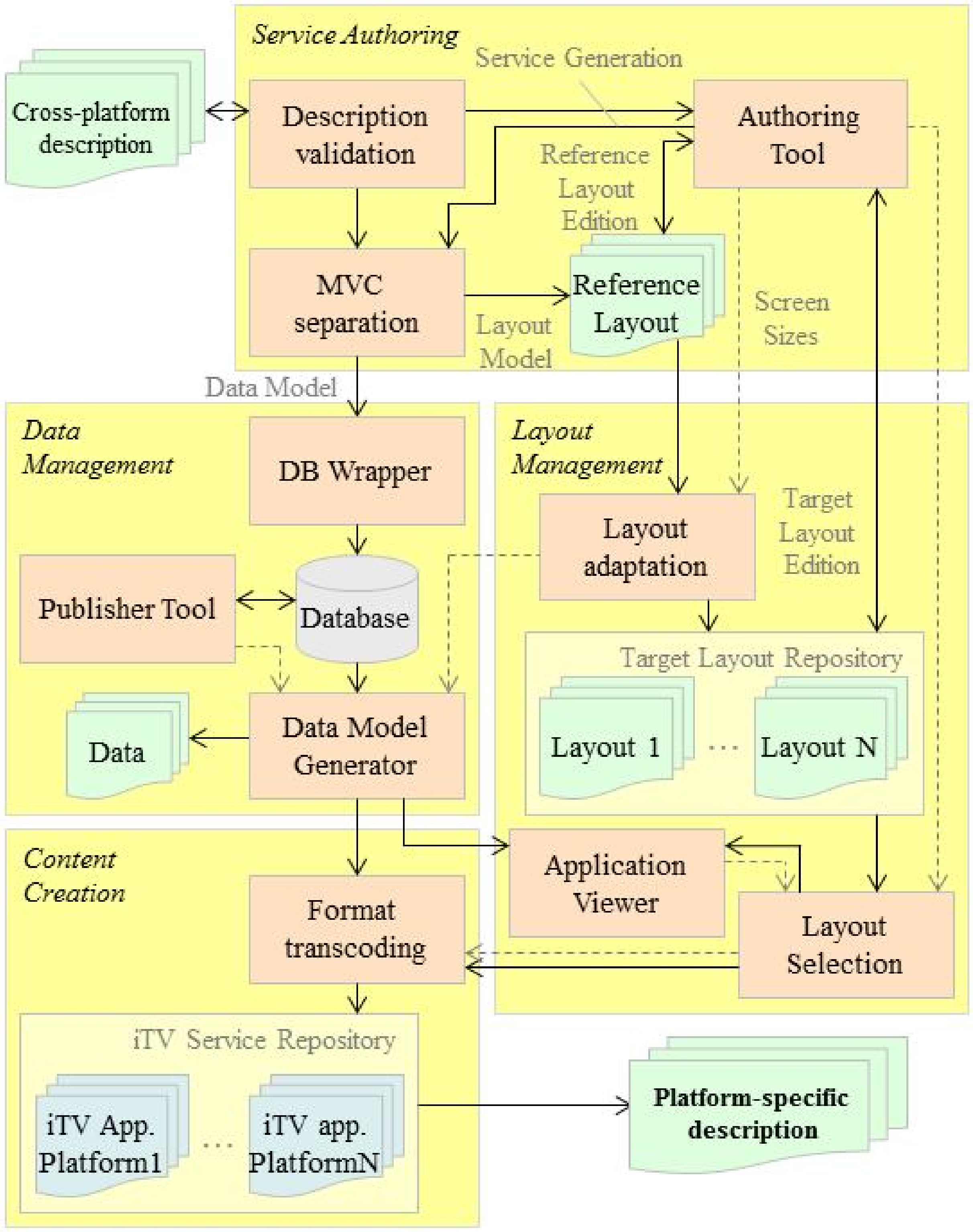


Figure 9. Overall System and Sub-System Architecture for cross-platform iTV production.

This architecture can be implemented entirely by a single company or organization (e.g. an application provider) but it can be also distributed between the many actors involved in the iTV production chain, as it fits within the two possible scenarios for transferring cross-platform applications. This architecture spans the process lifecycle; starting at the custom service authoring and the business-to-business interchange all the way to the service provided in a platform-specific format.

Service Authoring Subsystem

The Service Authoring sub-system is responsible for generating new services, acquiring external applications from the business-to-business interchange (which includes the parsing of an application description and its validation) and controlling the design and adaptation processes. It consists of the next components:

- Validation: Verifies the document schema and generates the object model structure.
- Authoring Tool: One of the key components of the system. Besides generating new service descriptions and also modifying them, it manages the scaling process allowing a redesign when necessary in order to achieve the best possible user experience in the target application format.
- Data Extraction: Extracts the editable data from the DVB-PCF description and provides the reference layout description, which can be updated by the Authoring Tool.

Data Management Subsystem

The Data Management sub-system manages the information system related to the application, which comprises all data that may be edited by content editors. It has been taken into consideration the following data that may be frequently edited: user interface texts, source of multimedia contents (images, audio, video, etc), as well as application metadata. The Data Management Subsystem depicts the basic interfaces between the proposed methodology and a simple information system. It consists of the following components:

- DB Wrapper: Receives a service data model and stores it in the database.
- Publisher Tool: Allows the maintenance of the data by content publishers.
- Data Model Generator: Extracts data from the information system and generates a cross-platform description of the required data from the one stored in the information system. Data and layout models can be managed across the architecture in a cross-platform format such as DVB-PCF (for better business-to-business interchange) and in its programmatic object representation if the interchange between modules is done in the same runtime system (which is more performant).
- Database: The information system.

Layout Management Subsystem

This sub-system is responsible for managing the application layout from its adaptation to the reference screen to its selection for transcoding. It consists of the next components:

- Layout Adaptation: Automatically adapts the reference layout to the different screen resolutions of the target layouts. This preliminary transformation is a suggested scaled version of the reference layout that must be verified by the Authoring Tool. This step can imply relevant changes in the application design in order to take the maximum advantage of a specific screen resolution, taking into account the recommendations for a specific target platform, such as the bandwidth optimization for mobile devices. The combination of the automatic scaling and the design control through the Authoring Tool with data independency gives full control over the application scaling reducing the time associated with design.
- Layout Repository: The generated target layouts are stored here.
- Layout Selector: Simple interface for the layout repository. It provides the suitable functions for facilitating the file system access and the layout selection.
- Application Viewer: Real-time display of the cross-platform version of the application for checking it before the transcoding step.

Content Creation Subsystem

The Content Creation sub-system transforms a DVB-PCF description into a specific iTV content language for a concrete iTV platform. It has two components:

- Format Transcoding: A key component of the proposed architecture. It translates a cross-platform application description based on the selected layout and the stored data into a platform-specific description of the application. This implies that the transcoder must implement a different set of transformations for each target content language but the service parsing and signaling can be shared by all the platforms. Moreover, several target platforms can support the same content language with little changes in implementation or instruction set (e.g. generic HTML-based services for different platforms).
- Service Repository: Stores the applications that are generated by the system.

3.1.3. Remarks

The methodology that has been presented in this section defines the major guidelines for the production of cross-platform iTV applications. It is based on the DVB-PCF standard as the integration model for abstracting the programming language in which the application is written. It also enables to overcome the differences between declarative and procedural programming. Moreover, the application of a progressive enhancement strategy for implementing the translation between the integration model and the target platform format ensures the support for the generated application across similar platforms with different profiles and resources. This methodology has been applied for translating DVB-PCF descriptions into web-based applications. Section 4.1 details this process.

The proposed architecture for building and maintaining cross-platform iTV applications integrates the presented methodology into a production system taking into account the typical elements of the architecture for producing applications. The key elements of the proposed architecture have been implemented in Java not only for testing purposes but also for serving as a base for building a software tool as a built-in solution for all the implemented features. Section 3.2 details the implementation for each element of the proposed software framework and the integrated development environment.

3.2.Framework for Creating and Managing Cross-Platform Applications

This section describes the implemented software framework and usage details for each software module. All modules can work separately as stand-alone applications or collaborate for implementing advanced architectures such as the proposed in this research.

3.2.1. Overall Framework Architecture

The developed framework is based on the proposed architecture for producing cross-platform iTV applications. The key modules of the presented architecture have been implemented in Java, licensed as free software under GNU General Public License (GPL)[3].

The proposed modules for the Service Authoring and the Layout Management subsystems are independent from the target format in which the DVB-PCF application description must be translated. The Data Management subsystem strongly depends on the specific information system related to the iTV application, so that its implementation and testing is out of the scope of the research. Finally, the Authoring Tool module requires a special attention. For research purposes, it has not been implemented as part of the framework but as an Integrated Development Environment (IDE) that includes the rest of the modules of this framework. This solution comprises in one tool the main features required for validating the proposals and experiments of this research.

Figure 10 outlines the modules that have been implemented in the framework and the relation with the rest of the proposed architecture. As shown in the figure, the basic workflow of the proposed methodology for building cross-platform iTV applications can be completed using the modules and tools that have been generated in the course of this research. Features and implementation details for each module are discussed below.

[3] Available at http://code.google.com/p/pcf-jtools [consulted on 08/07/2012].

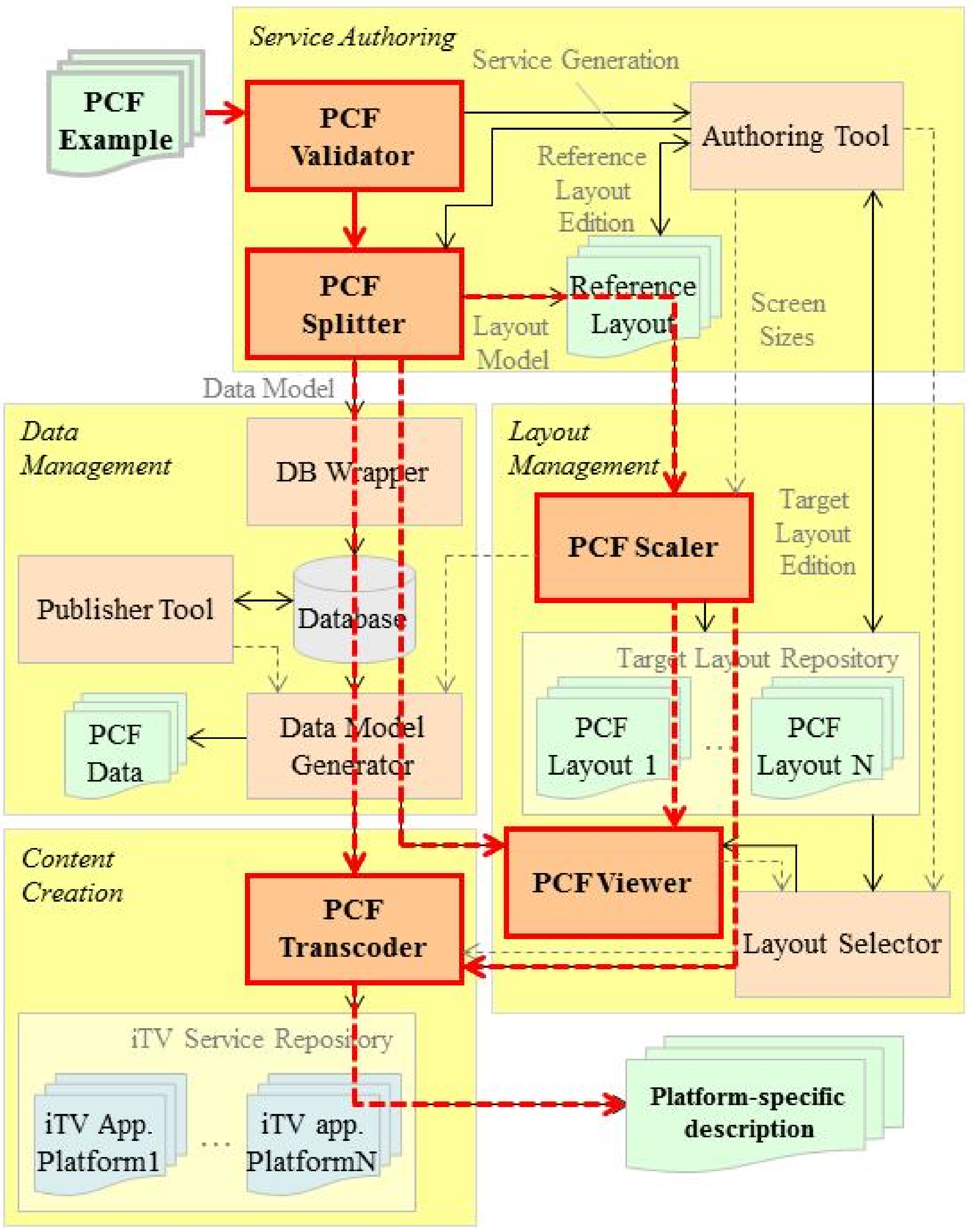


Figure 10. System and subsystem architecture of the proposed software framework.

3.2.2. Document Validation and Parsing: PCFValidator

The PCFValidator module implements the Validation component of the general architecture. For a DVB-PCF transcoder not integrated in an authoring framework (such that the validation of its output may not be ensured) the first step is to validate the correctness of the documents against the DVB-PCF XML Schema (TS 102 523 V1.1.1 Schema) [76] and parse them to have a memory object structure, which allows an efficient management of the application description. A DOM parser has been chosen for this step for validating the XML description prior to generating the object model. Moreover, a back and forth fast access is needed to read the DVB-PCF components across multiple documents, hence the DOM parser fits well in this case [66].

The result is a flag that indicates if the whole application description (including all referenced DVB-PCF documents) complies with the DVB-PCF Schema, which defines the rules for this kind of documents. Optionally, the generated object structure can also be retrieved through its Application Programming Interface, which facilitates its integration into the production system and improves the performance of the whole system by minimizing the XML management. The object structure used for the interchange of DVB-PCF descriptions across the entire system is the PCFModel, explained below.

Usage

- From command line:
  ```
  java -jar PCFValidator.jar pcf_file_path
  ```

- From Java code:

  1. Passing the file path of the XML document to be validated:
  ```
  com.cephis.pcfjtools.validator.PCFValidator validator =
    new com.cephis.pcfjtools.validator.PCFValidator();
  boolean isDocumentOk = validator.validate(pcf_file_path);
  ```

  2. Passing a file object:
  ```
  java.io.File pcf_file = new java.io.File(pcf_file_path);
  com.cephis.pcfjtools.validator.PCFValidator validator =
    new com.cephis.pcfjtools.validator.PCFValidator();
  boolean isDocumentOk = validator.validate(pcf_file);
  ```

3.2.3. The object Model: PCFModel

PCFModel is the Java class structure that has been created to describe an entire DVB-PCF description. It enables the interchange of application descriptions between the different components of the implemented software framework.

The component hierarchy of the object PCFModel follows the typical structure for an iTV application as defined in the standard DVB-PCF. Figure 11 presents an example of this component hierarchy for an application with two different scenes and some components that are shared by all scenes. All components not related to any scene are considered common components, i.e., components that must be always present in the user interface.

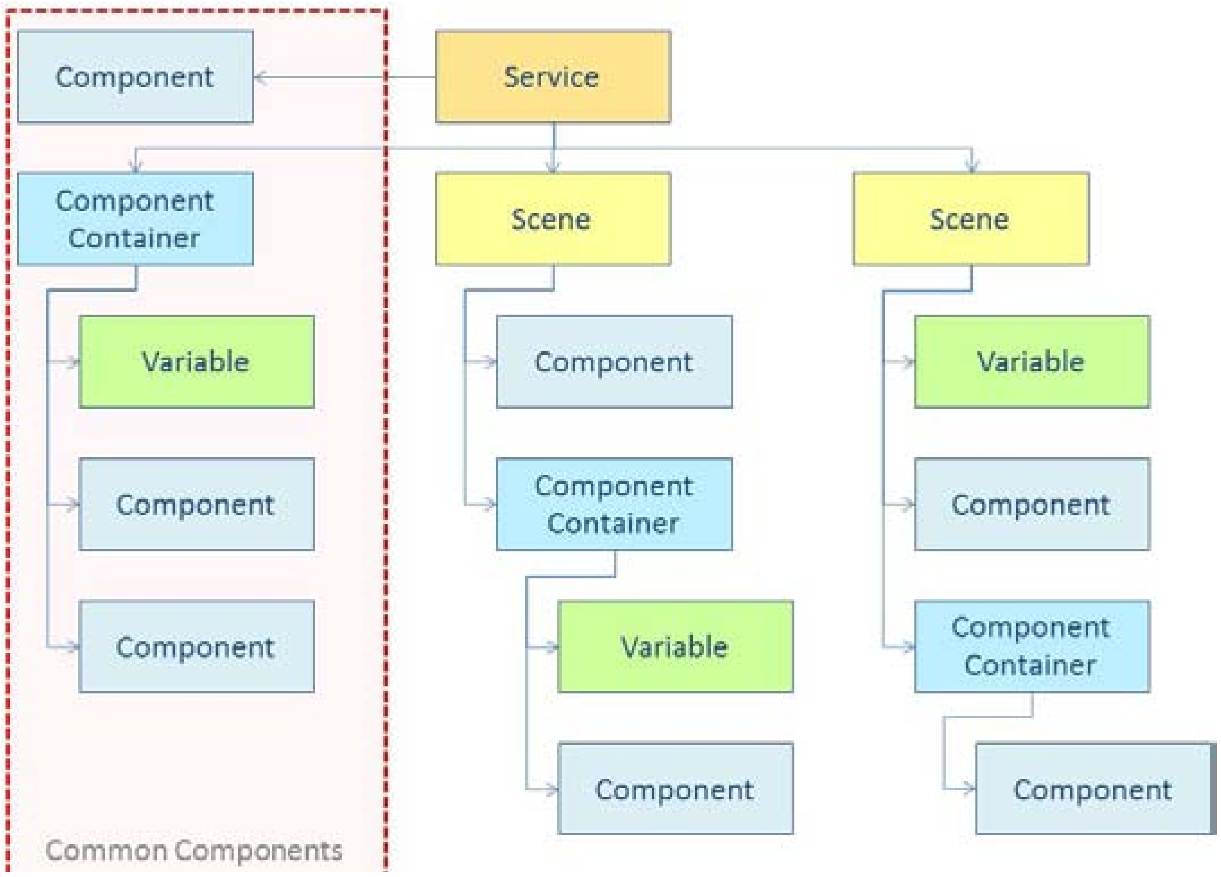


Figure 11. An example of the component hierarchy in DVB-PCF and PCFModel.

The relation between the above-mentioned components or items is shown in the UML class diagram of Figure 12.

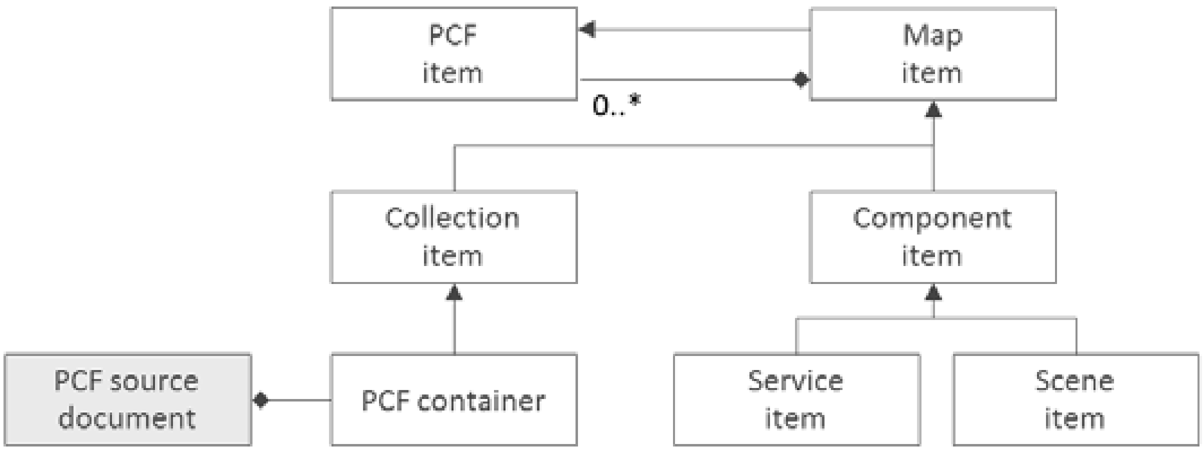


Figure 12. Relationship diagram between the structural items of DVB-PCF and PCF Model.

All DVB-PCF service descriptions shall be contained within a PCF container. A PCF container shall be the top-level item in a DVB-PCF source document. Every DVB-PCF source document shall have exactly one PCF container. All the structural items of a DVB-PCF service description, PCF container, component item, collection item, Scene item and service item, shall be kinds of Map item (which can contain unlimited number of DVB-PCF components). The low level containers of DVB-PCF service descriptions are collection items. This allows an author to group items into structures of their choosing in a way that is useful for their particular service description. Service and Scene items are the highest level of any structured description and can provide the entry points of a DVB-PCF service.

The PCFModel Java module contains the whole object structure for mapping a DVB-PCF description. Moreover, a simple API has been included for managing DVB-PCF files. This API enables reading any iTV application description across multiple referenced files, and the generated object mapping can be written as a new DVB-PCF document that unifies the entire description in a single document for a later management.

3.2.4. Adopting a Model-View-Controller Strategy: PCFSplitter

This module implements the Data Extraction component of the general architecture. It analyzes the whole application description and extracts the data model. All extracted data is replaced by a reference attribute in the structure of the application description and the result is a self-contained application with its description separated into the data model on one side and the behavior and layout on the other side. The generated description works as the input description and is independent from the rest of the system.

The PCF Splitter comprises two operating modes:

- Stand-alone application: The module works at the XML level as an independent application that can be executed without any other system modules. It reads a DVB-PCF description (one or multiple documents) from the local file system; analyzes its structure and generates two different DVB-PCF documents. One document for the data model and another one for the rest of the application description.
- Integrated component: The module works at memory level for its efficient integration into the whole production system. It provides the suitable API for reading the application description as a Java object and generates two different objects corresponding to the data model and the rest of the application description. This operating mode has obviously better performance than the work at the XML level and it is preferable for its integration into the system.

Usage

```
java -jar PCFSplitter.jar pcf_file_path
```

It generates two referenced PCF descriptions that will be stored in a new folder with the copy of the related resources.

3.2.5. Scaling the application for Multiple Views: PCFScaler

The PCFScaler module implements the Layout Adaptation component of the general architecture. This tool reads a DVB-PCF description and scales all visible components of the layout in order to fit within a reference screen size, which is defined according to the target platform requirements.

The tool calculates the ratio between the original screen size and the reference screen size in which the adapted application must fit. Then, this ratio is used to proportionally adjust texts, images, videos and the rest of visual components. Although many iTV runtime environments can scale automatically the application layouts, the proposed scaling step reduces considerably the bandwidth required to transmit the application to the target platform and improves the performance when running the application.

Usage

```
java -jar PCFScaler.jar pcf_file_path target_scale
```

It generates a new PCF description according to the target_scale value, which can be defined as:

- Proportional ratio: The value by which multiply the width and height of the resulting layout. Example: 2
- Fixed size: Two values that define the new width and height of the resulting layout. Example: 1280x720
- Aspect ratio: Two values that define the new aspect ratio of the resulting layout. Example: 16:9

Resulting documents are in a new folder with a resized copy of the images.

Examples

1. Half the original size:
```
java -jar PCFScaler.jar "c:\PCFProject\index.pcf" 0.5
```
2. Twice the original size:
```
java -jar PCFScaler.jar "c:\PCFProject\index.pcf" 2
```
3. Get a 1280x720 user interface:
```
java -jar PCFScaler.jar "c:\PCFProject\index.pcf" 1280x720
```
4. Adapt the user interface to 16:9 aspect ratio:
```
java -jar PCFScaler.jar "c:\PCFProject\index.pcf" 2 16:9
```

3.2.6. The Document Viewer and Debugger: PCFViewer

The PCFViewer module implements the Application Viewer component in the general architecture. After data management and the adaptation of the layout to the specific-platform requirements, the resulting DVB-PCF description of the application can be visually validated through this module. In this step, as the application is still in a cross-platform programming language that cannot be directly rendered into a specific platform, PCFViewer facilitates the application rendering for its visualization before its transcoding into the programming language of the target platform. This module has been built as a simple stand-alone application for visualizing any DVB-PCF description.

PCFViewer can read a DVB-PCF description from a single document with references to the rest of the documents that describe the entire DVB-PCF application. The result is the visualization of a DVB-PCF description with the layout scaled to the specified screen size and the application data integrated. This application data may include the static information (i.e., that not depending from a particular user request) and a sample set of the dynamic information provided by the information system for testing purposes. Figure 13 presents a screenshot of PCFViewer 1.0.

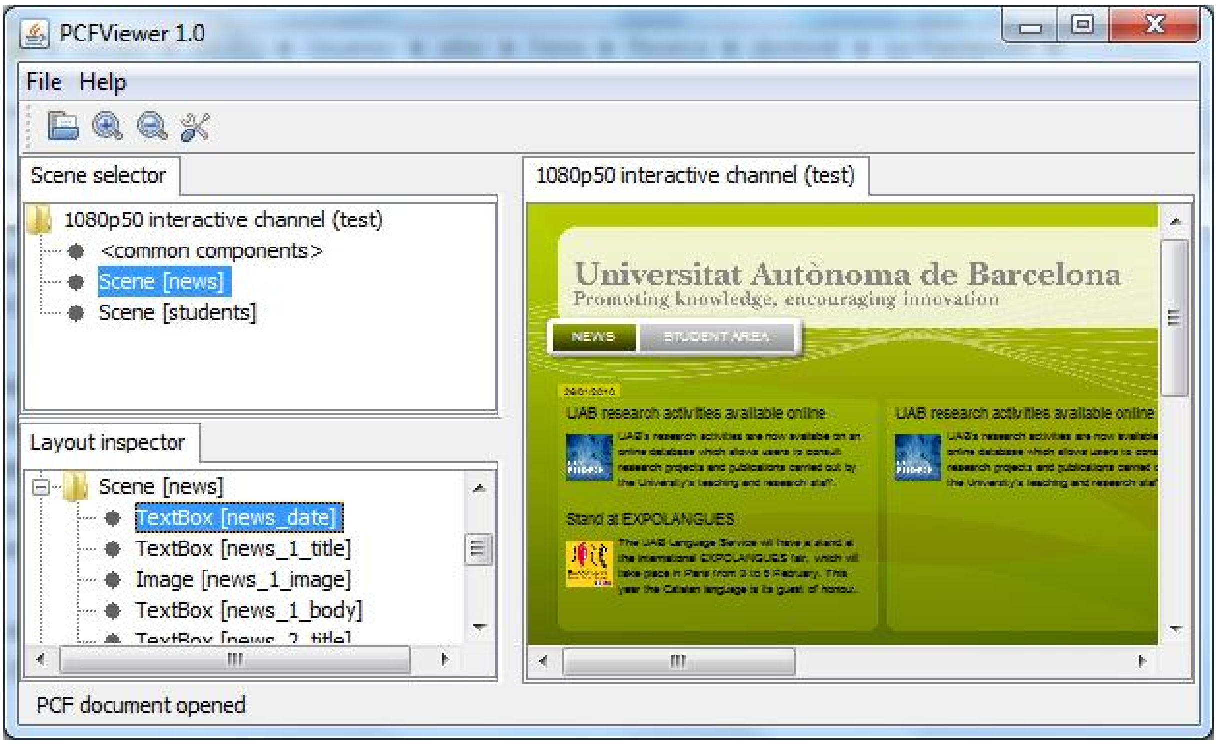


Figure 13. PCFViewer 1.0 presenting a sample iTV application described in DVB-PCF.

3.2.7. From Production to the Client Platform: PCFTranscoder

PCFTranscoder enables the translation from a DVB-PCF description into a platform-specific programming language. As other components of the framework, it can work at XML level for reading DVB-PCF descriptions from the file system, and also at memory level for a better integration within the whole system.

The translation process is divided into two steps, the first for the platform-independent transcoding and the second for the platform-specific transcoding.

The first step of the translation is applied for all transcodings and it is target programming language independent, either if it is a declarative or procedural language. The aim of this step is building a unified representation of the application across the different description documents by solving the references between themselves. This representation describes the entire application as a nested object structure that includes: the different elements of the application, its behavior and its metadata. The result is a memory representation of the application that enables a fast parsing and translation of each element into the target programming language, by all means faster than working at XML level.

The second step of the translation is specific to each particular programming language in which the DVB-PCF needs to be translated. This process starts with the analysis of the service description and generates the application container and metadata according to the target format. Then, the scenes of the user interface are written in the target language and the elements for each scene are converted according to the component clustering performed. The correlation between DVB-PCF components and its translation into the target programming language is stated in a configuration file, which enables a flexible way of changing the target format into the transcoder module. Finally, the resulting code can be written to a single file (useful for simple applications) or multiple files for a better separation between the model, the view and the controller of the application.

Usage

```
java -jar PCFTranscoder.jar [-o output_path] [-verbose]
     [-f output_format] pcf_file_path
```

It translates DVB-PCF services into web applications with two possible output formats: (1) plain HTML and (2) AJAX-based.

3.2.8. Integrated Authoring Tool for Building iTV Applications: VisualPCF

This tool enables the application and validation of all modules of the proposed software framework for the building and maintaining of cross-platform iTV applications. It integrates the previous modules and includes an authoring tool for generating iTV user interfaces.

The generated IDE, called VisualPCF, enables the generation of iTV user interfaces based on the standard DVB-PCF. The set of edition facilities provided by the tool is limited in comparison with professional tools and some features are still under development or in continuous improvement. However, the methodology proposed in this work can be fully tested and validated through the generation of simple but functional iTV services, as explained in the next chapter.

This authoring tool allows the visual design of simple iTV services with the typical presentation components for this kind of multimedia environment. This tool is not intended to be a commercial tool nor a complete solution for the designing of DVB-PCF layouts. However, it is quite well suited for testing and research since it enables the easy creation of DVB-PCF descriptions of an entire iTV service as well as the management of the DVB-PCF transcoding and scaling processes; the source code verification of the designed DVB-PCF description and its visualization at different scale levels.

These are the key features provided by the tool:

- Visualization of iTV applications written in DVB-PCF.
- Reading of DVB-PCF descriptions across multiple referenced documents.
- Validation of DVB-PCF documents based on the DVB-PCF Schema.

- Visual edition of DVB-PCF user interfaces. Drag and drop for adding elements and easy configuration of the element properties.
- Source code edition for DVB-PCF descriptions.
- Tree navigation of the document structure.
- Identification and management of the different scenes in the application.
- Export in DVB-PCF format. It can be performed into a single document as well as into multiple documents based on a Model-View-Controller strategy.

Figure 14 shows a use case diagram for VisualPCF and Figure 15 presents a sample screenshot of this tool.

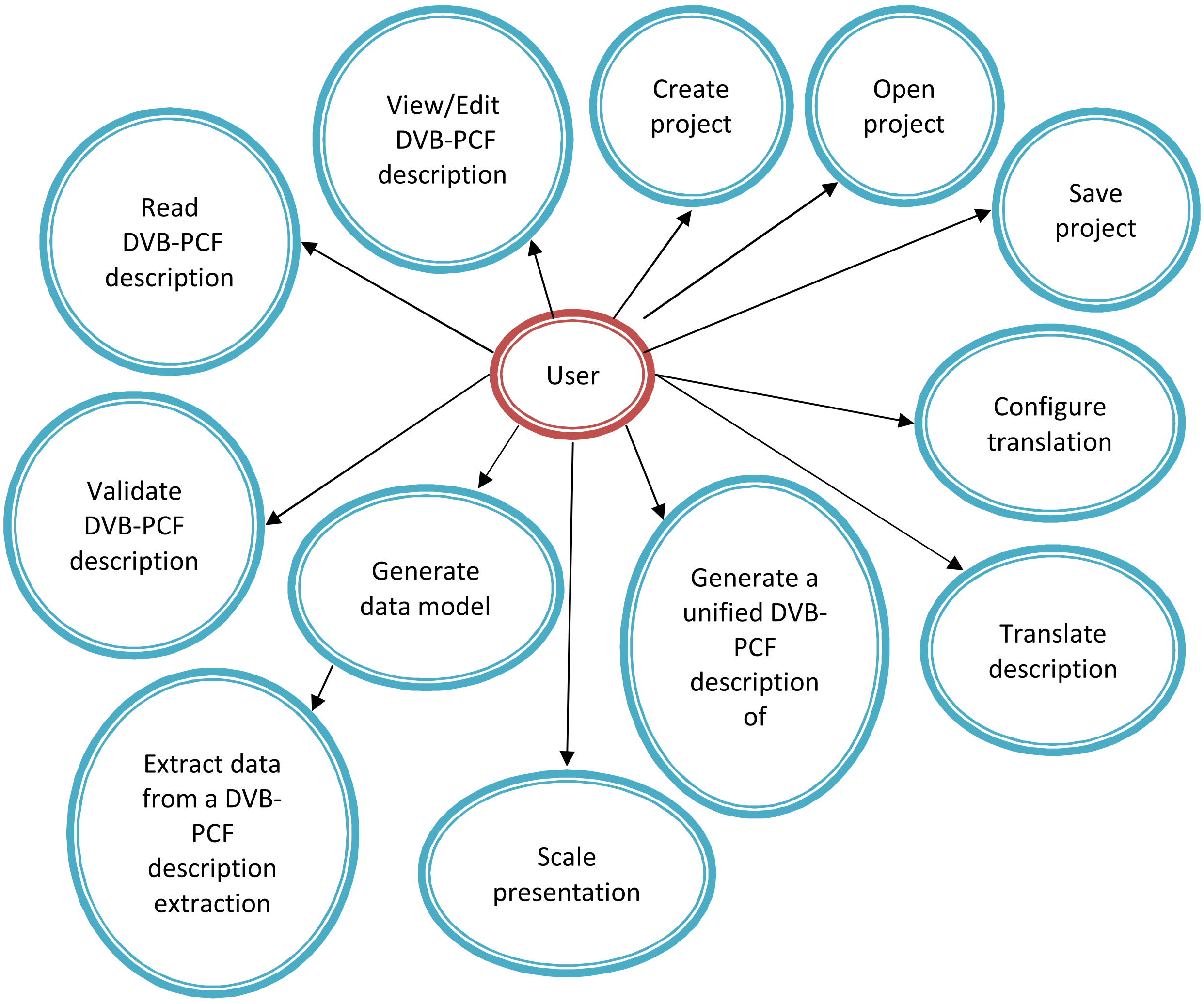

View/Edit DVB-PCF description
Create project
Open project
Read DVB-PCF description
Save project
User
Configure translation
Validate DVB-PCF description
Generate data model
Generate a unified DVB-PCF description of
Translate description
Extract data from a DVB-PCF description extraction
Scale presentation

Figure 14. Use case diagram for VisualPCF.

VisualPCF has facilitated the development of cross-platform iTV user interfaces with simple but functional behavior. Although the set of implemented features is limited to the scope of this research, the development of some interesting cross-platform applications has been started with this tool (both for testing and research projects).

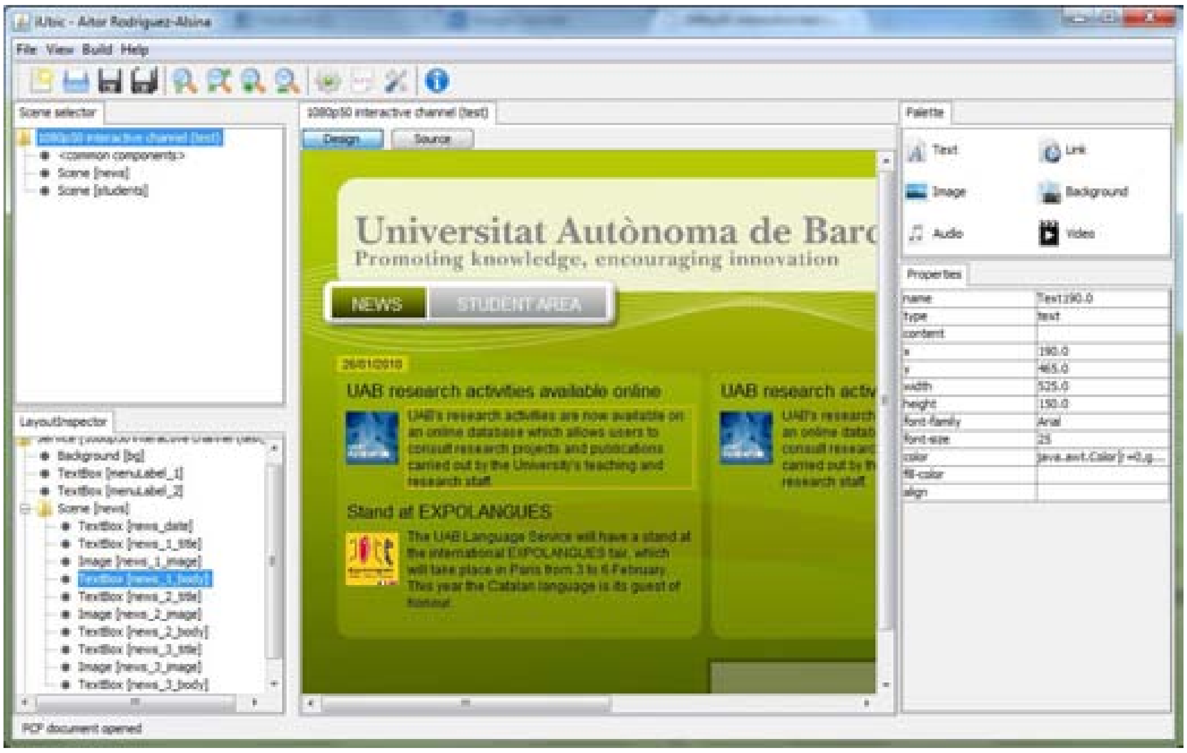


Figure 15. Screenshot of VisualPCF editing a sample iTV application.

3.2.9. DVB-PCF Examples

Some examples of simple DVB-PCF services have been included in the software framework for testing purposes. The source code has been taken from the DVB-PCF Specification 1.0 (TS 102 523 V1.1.1) [47], which provides a further description of them[4].

- hello_world.xml: Simplest possible DVB-PCF description.
- external_content.xml: Describes the same viewer experience captured by the "hello_world" example by referencing out from the DVB-PCF source document to an external resource (whose content is "Hello, world!").
- separated_documents.tar.gz: Example of how a service description can be partitioned into more than one DVB-PCF source documents.
- scene_template.tar.gz: The service description is partitioned into two documents, the service item on one side and the service layout on the other.

3.2.10. Remarks

The implementation of the presented framework has enabled the validation of the proposed methodology. This framework also enables further developments around the standard DVB-PCF thanks to its ability to work independently from the rest of the system. Section 4.2 details the experiments driven with these tools.

[4] These examples are also available for downloading at http://code.google.com/p/pcf-jtools/. [Consulted on 08/07/2012]

3.3.Approach for Synchronizing Multimedia Subtitles across Web Platforms

One of the key issues of the cross-platform iTV applications explored in this research is the content synchronization. This feature is especially important in the TV environment because it is usually driven by the audiovisual content with strict time constraints (live programs, TV programming, enhanced information related to a specific event such as sports, etc). In addition, the new trend for TV-related second screens (or multi-screen) for sharing contents across different devices raises new challenges regarding synchronization between different content elements on multiple platforms.

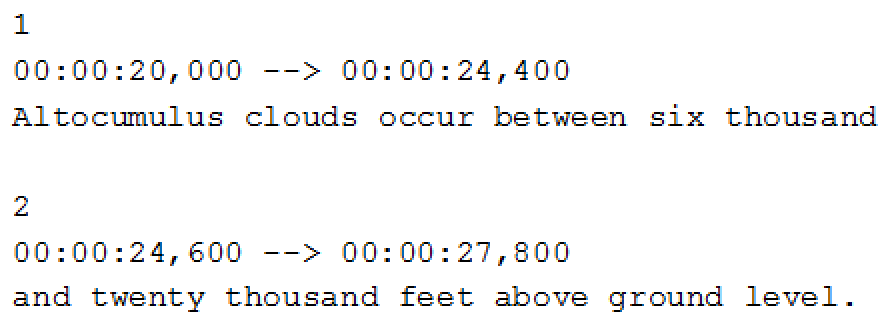
The content synchronization problem was addressed through a specific use case to explore the suitability of the methodology and tools proposed for generating iTV applications with time constraints. That was the motivation for building a video subtitling system that synchronizes video and subtitles on the client platform rather than incrust subtitles in the production side. This is the normal procedure on broadcast systems, which can transmit different audio streams for the same video content. But what happens with web-based applications?

When exploring the options for implementing the transcoding for time-based constraints we realized that there is currently a lack of a clear methodology for managing subtitles on web applications. HTML5 is in the way of stating the problem through the definition of a subtitle tag in the video component and the related browser behavior. Meanwhile, there are some JavaScript solutions that can solve the problem in a procedural way, but they require the use of third-party libraries that may hinder the portability of the implemented solution.

Taking advantage of the underlying Web ubiquity, this work proposes a video subtitling system that enables subtitle format customization and its adaptation to needs of the user/application. It is based on HTML5 and the new features that enable SVG and SMIL. Since the standardization of the HTML5 video component and its acceptance by the major Web browsers, there has been some implementations of the synchronization and presentation of subtitles attached to a <video> tag [67]. A different approach can be achieved by taking advantage of SMIL time properties for the multimedia content synchronization. Instead of managing the content timeline programmatically through JavaScript, a web programmer can make use of the event management and timeline running on the SMIL engine of a Web browser. This dramatically simplifies the development of a time-based application like subtitling by delegating the time management (i.e. the presentation time of each subtitle or caption) to the browser.

3.3.1. Background

Most subtitles distributed on Internet are described in text files that follow the SubRip (.SRT) format, considered "perhaps the most basic of all subtitle formats" [68]. Figure 16 shows an example of a .SRT file. Each subtitle entry consists of the subtitle number, the time the subtitle should appear on the screen, the subtitle itself, and a blank line to indicate the end.



```
1
00:00:20,000 --> 00:00:24,400
Altocumulus clouds occur between six thousand

2
00:00:24,600 --> 00:00:27,800
and twenty thousand feet above ground level.
```

Figure 16. Example of subtitles defined using the SubRip (.SRT) format.

Jan Gerber was one of the first developers that implemented a subtitle synchronization system for a HTML5 <video> tag, which resulted in the JavaScript library jquery.srt.js [69]. This library parses the HTML document looking for subtitle containers which have been previously defined with the "srt" class, and attempts to load the related subtitles. Subtitles can be written directly in .SRT format into the subtitle container or accessed remotely through the address defined as a custom parameter into the HTML container. This implementation works correctly but relies on custom HTML attributes.

Taking Gerbers's approach, Michael Dale proposed an evolution that demonstrates the benefits of including some HTML5 elements as child nodes of the HTML5 <video> container to associate a video component with time-aligned text [70]. This eliminates the custom attributes of the previous approach but still relies on the same JavaScript library as subtitle synchronization for time-related issues.

Philippe Le Hegaret adapted Gerber's JavaScript library to demonstrate the viability of using Gerber's proposal with a different format for the presentation of time-based text in his HTML5 DFXP Player prototype [71]. He used the Distribution Format eXchange Profile (DFXP) of the Timed Text Authoring Format (TT-AF) [72], which was created by the W3C for the conversion, exchanging and distributing timed text information amongst legacy distribution formats for subtitling and captioning.

Alex Danilo proposed a completely different approach to multimedia synchronization in web-based environments [73]. This work presents an example of multimedia and graphic synchronization using SMIL to present SVG subtitles that are displayed on top of an SVG video. SMIL facilities allow the direct definition of time marks on subtitle text elements and specify the begin time and duration for each subtitle. No JavaScript code is needed because the SMIL engine is responsible for maintaining the synchronization between video and subtitles.

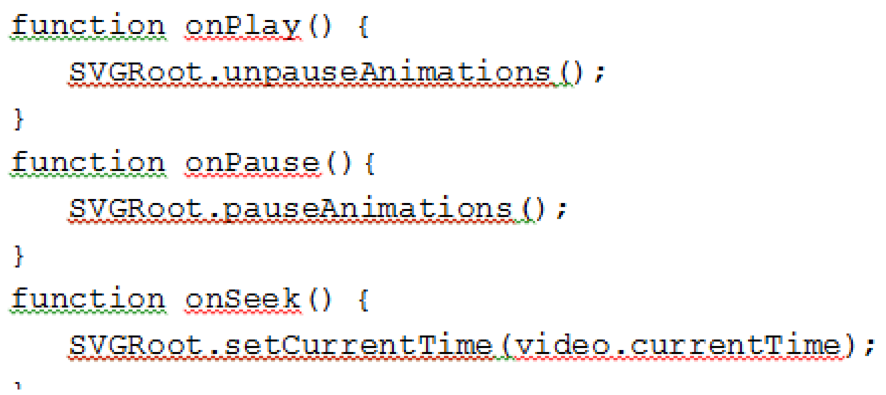
3.3.2. Approach Description

Inspired by Danilo's approach for synchronizing SVG subtitles through SMIL, the present work proposes a comprehensive solution for video subtitling based on HTML5. This solution enables the acquisition and parsing of remote subtitle files and allows dynamic modifications to the selected language and text format during video playback. It is robust against changes on the video timeline, i.e. the subtitles remain synchronized after functions such as pause, fast-forward, direct jump, etc. Subtitle text components are written in SVG in order to improve scalability and allow SMIL animations on them; JavaScript is used for parsing functions, client-server communications, the dynamic creation of subtitle text elements, and the event management of the <video> container. SMIL manages the time-related functions and decides the subtitle text visibility according to the “begin” and “end” attributes defined by the SMIL <animation> element. These time marks correspond to the times defined in the related .SRT subtitle file. Figure 17 shows a subtitle definition using an SMIL animation on HTML5 documents. Text opacity is initially set to 0 to hide the text, and the <animation> element specifies when this opacity attribute must be changed to set the text to visible. The time format is the same as in the .SRT subtitle files (i.e. hours:minutes:seconds,milliseconds), so no calculations or JavaScript are required in the translation. The SMIL engine of the Web browser manages the rest of the process in order to keep subtitles synchronized with the video content, so the application developer is relieved of this responsibility.

```
<text opacity=0>
  <animation attibuteName='opacity'
    begin='00:00:20,000'
    end='00:00:24,400'  values='1'/>
  At the left we can see...
</text>
```

Figure 17. Example of SVG text element with SMIL animation for presenting a subtitle.

One interesting novelty provided by HTML5 is the support for embedded SVG and SMIL code in the document as for any other HTML tag. These elements are also included as part of the Document Object Model (DOM), which means that they can be accessed and modified through JavaScript and CSS in the same way as any other HTML tag. By including SVG subtitles as part of the web document, any web application can present customized subtitles natively, at least in platforms with the suitable web browser. However, this basic implementation is insufficient to support user-driven changes in the video timeline (e.g. pause, seek in time) because the timeline managed by the SMIL engine must also be notified about these changes. This is important for keeping the subtitles synchronized with the video content. When the user modifies the video playback timeline, any web application can automatically apply the suitable modifications to the SMIL timeline through the JavaScript API provided by HTML5 for the SMIL time-management. For a basic playback management, three basic video properties are required: onplay, onpause, and onseeking. Figure 18 shows an example of the JavaScript functions required to control the SMIL timeline. When the user pauses, plays or seeks on the video, the HTML5 <video> container fires these JavaScript functions. The SVGRoot element of the example stands for the root element of the entire SVG document (i.e. <svg>) and the variable video references the <video> container itself.



```
function onPlay() {
    SVGRoot.unpauseAnimations();
}
function onPause(){
    SVGRoot.pauseAnimations();
}
function onSeek() {
    SVGRoot.setCurrentTime(video.currentTime);
}
```

Figure 18. JavaScript code for linking the HTML5 video element and the SMIL container.

This behavior is represented in Figure 19, which illustrates the basic synchronization schema between the video component and its subtitles. A couple of video pause operations are also included in the diagram.

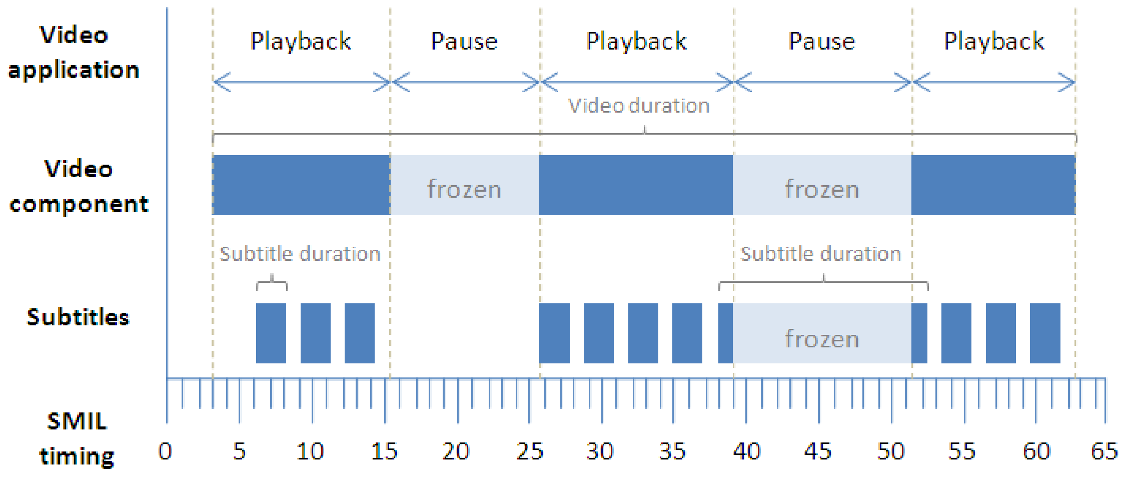


Figure 19. Time-based strip diagram with two video pause operations.

When the user pauses the video component through the provided video player application or the system fires an event to pause the video (e.g., when the user changes between system applications) the JavaScript functions spread the pause event to the SMIL timeline. This helps to keep the subtitle timeline synchronized with the video according to the subtitle presentation times. If a subtitle element was being presented during the pause operation, it is frozen until the video playback is resumed again. While the SMIL time line is paused all its time-dependent components remain in pause until the time is resumed again. Subtitles are synchronously presented depending exclusively on the SMIL timeline and its own presentation time. When the user generates jumps in the video timeline by seeking a concrete time position, the video application sends a signal to the SMIL timing container indicating that the time position has to be updated. After the update, each subtitle component is presented according to the new current position on the SMIL timeline. Figure 20 outlines this behavior when the jump is forward in time.

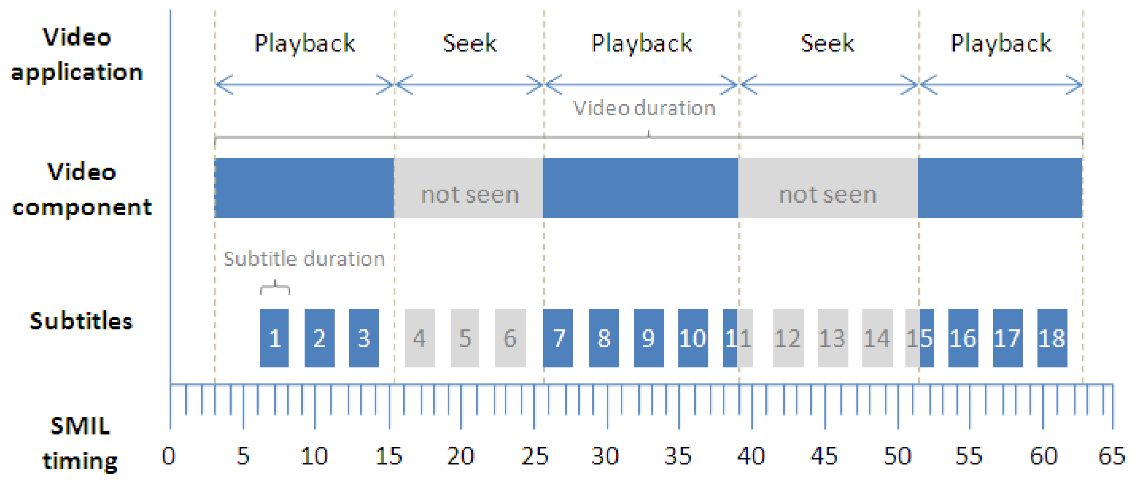


Figure 20. Time-based strip diagram with two video seek operations.

The change of the subtitle language operation, which is outlined in Figure 21, requires loading of a new subtitle file (according to the SRT specification). This file can be obtained and loaded dynamically from a remote location while the video is playing. The loading operation can add a non-predictable latency depending on the quality and speed of the connection required to get the new file. Since the user generates the language change event, the subtitles of the current language (language 1) can be still displayed until the system is able to present the new language subtitles (language 2) in order to maintain the continuity of subtitles. However, tests driven on the system indicate that this can be confusing for the user because the system does not seem to react (somehow) to the user action immediately. Instead, if the current language (language 1) disappears when the user starts the action he gets the feel that something happened and the system is working.

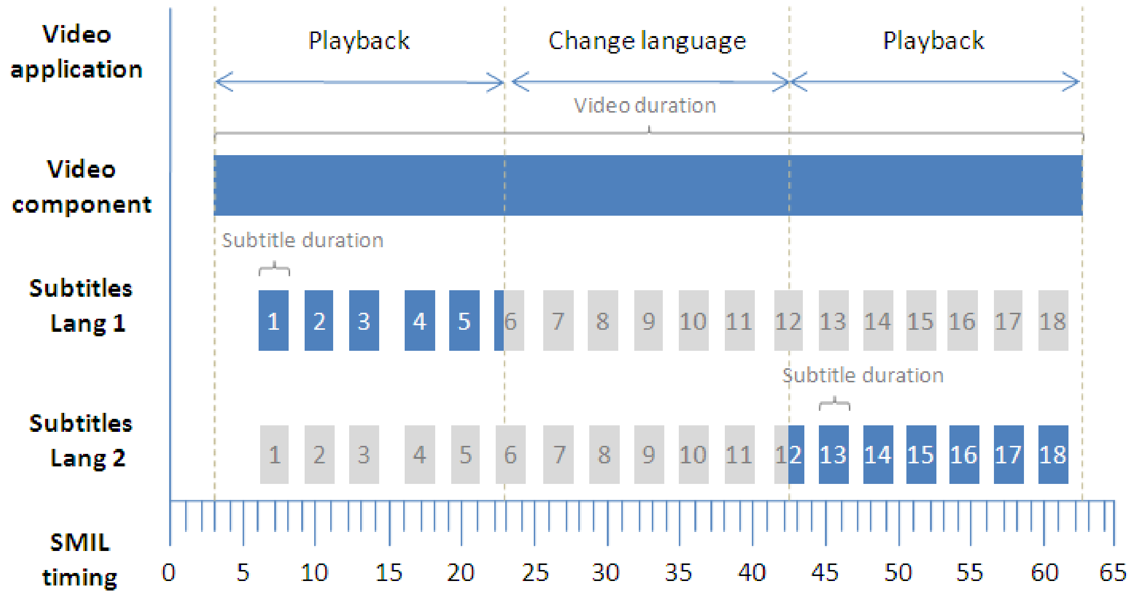


Figure 21. Time-based strip diagram with a subtitle language change operation.

For testing purposes we have simulated latencies up 15 seconds, which is the default HTTP connection timeout of Apache 2.0 web server. In practice, the subtitle file is usually obtained in much less time due to the file size (less than 100KB for a 2 hours film) and a proper connection quality.

Finally, the screen export operation required for multi-screen scenarios is the responsible for starting synchronization between the main screen and the secondary screens. As shown in Figure 22, when a secondary device is connected to the main system to obtain a complementary language for subtitles, the system exports its SMIL time container to the secondary device taking into account the starting time and the current time. This enables the secondary device to have the same time reference that the main system, keeping the synchronism between both systems as long as the main system does not modify its own timeline. When this occurs, the main system sends a signal to the secondary device to update its timeline according to the main system. In this case, it is assumed that both systems are in the same environment (e.g., the home environment), so the latency of this signal can be negligible or it is short enough to ensure a proper user experience. However, if a pause operation generates a jitter between both paused systems, the secondary device may display the content slightly advanced in time (e.g., the next subtitle is displayed in the secondary device while the main system does not display it yet). When occurs, this inconsistency is solved in the next playback or seek operation, when the main system sends again its current SMIL time to resynchronize both systems. This point is included within the user experience evaluation driven in this work, which is described in Section 4.4.2.

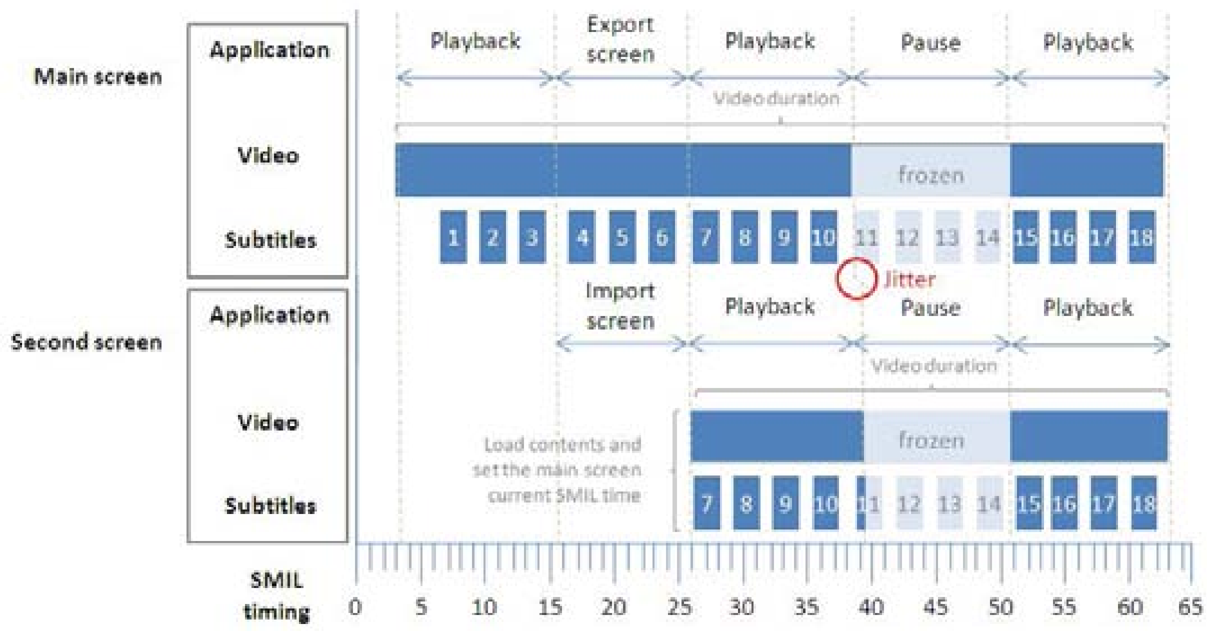


Figure 22. Time-based strip diagram with a screen export operation.

3.3.3. Analysis of the Current Platform Support

The platform support for the proposed solution depends on the availability of a web browser and its support for some HTML5 features such as the video and audio management, SVG and

SMIL animations. Although a variety of browser plugins and SVG players exist, we focus on the native support for SVG by the main web browsers in order to ensure maximum ubiquity. In this section we analyze the support for these features in three different application environments: desktop, mobile and TV.

Desktop environments

Nowadays, all major modern desktop web browsers provide support for HTML5 and SVG to the level required by the proposed solution, with the exception of Internet Explorer, which can present some problems with SMIL animations. Google's announcement on August 2010 that it had begun to index SVG content (standalone files as well as embedded in HTML5 documents) and cases of successful applications such as Wikipedia (which presents SVG images) all push for a wider acceptance of the format. Table 5 summarizes the support of the proposed subtitling solution in main desktop web browsers. The table presents the main features that are required for building the proposed subtitling solution and its availability in the main web browsers. As this analysis was previously performed in mid-2011 in [74], the table shows the evolution from those results (first column) to the current state of web browsers for supporting the solution (second column). The table also shows how the development teams of major web browsers are seriously involved in providing full support to HTML5 and its related technologies such as SVG and SMIL. This fact is more evident in the mobile side, where the industry seems to progress more rapidly in this direction.

Required feature	Desktop Web Browsers									
	Firefox		Chrome		Opera		IExplorer		Safari	
	4.0	*10.0*	*12.0*	*17.0*	*11.1*	*11.6*	*9.0*	*9.0*	*5.0*	*5.1*
HTML5 documents	X	X	X	X	X	X	X	X	X	X
HTML5 Video	X	X	X	X	X	X	X	X	X	X
SVG basic support	X	X	X	X	X	X	X	X	X	X
Inline SVG	X	X	X	X	-	X	X	X	-	X
SMIL animation	X	X	X	X	X	X	-	-	X	X

□ Evolution of the HTML5 feature support between different browser versions.

Table 5. Support of the proposed solution in main desktop web browsers.
For each browser, the first column indicates the support of the available browser version in our previous study (mid 2011) and the second column indicates the support of the current browser version (early 2012).

According to the table, the proposed subtitling solution is supported in all major desktop web browsers with the exception of Microsoft's Internet Explorer, which has not updated the previous version and it still requires a third party library or plugin to deploy SMIL animations. Moreover, as far as we know, the support of this feature in next version of the browser (10.0) was unknown at the time of writing this research.

Mobile environments

The mobile industry is evolving rapidly to HTML5, which facilitates the development of mobile web applications. Proof of this is the recent support of new HTML5 features by all major mobile web browsers. Less than one year before, we concluded in [74] that mobile web browsers were still some way back when compared to the HTML5 development in desktop browsers. Nowadays, the support that mobile web browsers can provide to HTML5 and especially to SVG and SMIL can be fully comparable to desktop environments. As shown Table 6, all the HTML5 features required for the implementation of the proposed subtitling solution are now supported by all major web browsers for mobile environments. This means not only that the solution is totally feasible for the mobile market but also that all kind of multimedia applications with content synchronization, scalable graphics and SMIL animations on these graphics are now accessible by most of the users with a smartphone or tablet. This global support should lead to a greater spread of this type of applications for mobile devices. As in the evaluation of the browser support for desktop environments, Table 6 presents the evaluation of the required HTML5 features in the analysis done at mid-2011 (first column) compared to the current state of mobile web browsers.

Required feature	Mobile Web Browsers							
	iOS Safari		Android Browser		Firefox		Opera Mobile	
	4.2	*5.0*	*2.3*	*4.0*	*4.0*	*10.0*	*11.0*	*12.0*
HTML5 documents	X	X	X	X	X	X	X	X
HTML5 Video	X	X	X	X	X	X	X	X
SVG basic support	X	X	-	X	X	X	X	X
Inline SVG	-	X	-	X	X	X	-	X
SMIL animation	-	X	-	X	X	X	-	X

☐ Evolution of the HTML5 feature support between different browser versions.

Table 6. Support of the proposed solution in main mobile web browsers.

For each browser, the first column indicates the support of the available browser version in our previous study (mid 2011) and the second column indicates the support of the current browser version (early 2012).

TV environments

The interactive TV situation is fuzzier than for mobiles because the technological details of what constitutes a Smart TV have not yet been clearly defined. Currently, three main approaches coexist as Smart TV platforms:

1. Device and Internet operators have presented their own solutions. In the case of Apple and Google (which have respectively created AppleTV and GoogleTV) the first provides an integrated Set-top Box (STB), whilst the second provides a framework deployable on prepared STBs and TVs. Both support the main features of HTML5, but they do not yet support SVG animations.
2. TV manufacturers build Internet-capable devices that combine the TV video flow with web applications and widgets. Their browser implementations provide little support for HTML5 features as they have very limited resources (even less than mobile devices). Nowadays, Smart TVs are the current trend for this kind of systems, which also include applications for adding new services and applications.
3. Some Internet video service providers (e.g. Netflix, Boxee) are interested in collaborating with TV and STB manufacturers to deploy their services on a variety of terminals.

In any case, the support for HTML5 and SGV animations on smart TV devices depends on the available device resources to deploy a web browser that supports the required features. Fortunately, the Open IPTV Forum recommends the use of SVG (SVG Tiny 1.2) for declarative environment applications on the client side [75] and some initiatives for the standardization of hybrid digital TV platforms (such as HbbTV) are following this model. Even so, the SVG specification could add some features in order to make SVG more suitable for TV (e.g. support for TV-related remote control keys, ability to seek in time and speed for fast forwarding/rewinding, etc).

3.3.4. Remarks

This section has presented a novel approach for video subtitling which enables a flexible management of subtitles in the client side. It allows multi-screen solutions that can be applied in the home environment and public spaces. Section 4.4 describes the experiments driven with this solution for evaluating the suitability and correctness of the proposal.

3.4.Summary of the Chapter

The chapter has presented the proposals and main contributions around the cross-platform design of interactive applications for TV environments.

The main proposal, presented in Section 3.1, describes the methodology for building and maintaining cross-platform applications based on the standard DVB-PCF. This portable content description enables an easy translation into platform-specific formats by its high-level definition of the user interface, which describes the user experience rather than the implementation details.

The implementation of this methodology has generated a software framework and an integrated development environment for enabling developers and content editors an easy authoring of cross-platform iTV applications written in DVB-PCF. These tools, presented in Section 3.2, enable the validation of the proposed methodology and further experiments in this area.

Finally, Section 3.3 presents a novel proposal for synchronizing video subtitles on web-based platforms. It is based on the Scalable Vector Graphics (SVG) standard and reduces considerably the code required by an application for managing time issues.

CHAPTER 4

EXPERIMENTS AND VALIDATION

"Your scientists were so preoccupied with whether or not they could, they didn't stop to think if they should."

Dr. Ian Malcolm ("Jurassic Park", 1993)

This chapter presents the experiments driven for validating the proposals of this research and discusses the provided results. The proposed methodology is validated through the implementation of two different transcoders for translating an iTV application written in DVB-PCF into MHP and HTML5. The proposed software framework has been used for creating a second cross-platform application based on an eHealth service for TV. Finally, the proposal for synchronizing subtitles on web-based platforms is analyzed through its implementation on three use cases and validated through the user experience evaluation of the most representative.

4.1. Experiments Driven for Validating the Proposed Methodology

This section describes the experiments driven for the validation of the proposed methodology. The objective was to demonstrate the translation viability of the cross-platform DVB-PCF description of an iTV application into two platform-specific programming languages with different programmatic models:

- HTML5, as a declarative format language, and
- MHP, as a procedural format language.

The description of the translation process for each platform-specific format has been divided into different parts related to the features of a typical iTV application:

- Component matching: State the relationships between DVB-PCF components and the components of the target programming language.
- User interaction and navigation: State the behavior of the application in response to the user generated events.
- Content provider interaction: Define the two-way communication between the application and the content provider.
- Content synchronization: Presents the particularities for synchronizing contents in each target format and the issues for being translated from DVB-PCF.
- Remarks: This final section summarizes the partial conclusions of each specific use case.

Finally, this section presents the overall results and conclusions obtained from these experiments.

4.1.1. Use Case description: Virtual Campus for iTV

Firstly, the presented methodology was successfully applied in the development of an iTV application for learning environments in the context of the HDTVNext European project (AVANZA I+D TSI-020400-2008-44). This project achieved in 2010 a complete chain of 1080p50 TV: from the content acquisition to its presentation in consumer devices. It means that the video coding format must be progressive (non-interlaced) at 1080 lines per frame and at 50 frames per second (the commercial term for Full HD, for example, refers typically to 1080p25). The key innovation of the project application was the development process, which enabled the design of an iTV application for multiple platforms through the DVB-PCF standard. In the abovementioned European project it was decided to use an IPTV platform to demonstrate the viability of the 1080p50 chain, hence we applied our cross-platform strategy for building a HTML5 application (the optimum format for the defined by the platform). This was the first transcoder we developed and, as far as we know, the first DVB-PCF to HTML5 transcoder published in a scientific publication [81]. In this work we explored DVB-PCF as cross-platform presentation format and its facilities to be translated into another declarative language such as HTML5. That process is described in Section 4.1.2.

After that research and subsequent experimental work, a DVB-PCF to MHP transcoder was developed to demonstrate the viability of generating procedural descriptions in addition to declarative ones. This experiment was useful to show the differences between both programmatic models when transcoding a DVB-PCF description. The process is described in Section 4.1.3.

4.1.2. Transcoding DVB-PCF for Declarative Runtime Environments: HTML5

Next lines describe the performed process for building a transcoder from DVB-PCF descriptions to HTML5 code.

Application Container

The container into which the application is translated corresponds to the basic structure of a HTML5 application. When a DVB-PCF application description is translated into HTML5, the transcoder generates this structure according to the configuration parameters of the DVB-PCF Service component. It includes the references to the behavior (JavaScript) and style (CSS) of the application. The remaining components of the application are accommodated within this structure, presented in Figure 23, and the related JavaScript files.

The presentation style of the application is defined in the CSS file stylesheet.css and the behavior, in the JavaScript file behavior.js, which can include other JavaScript files for a modular implementation. Both documents are filled with the style and behavior of each application component.

```
<!DOCTYPE html>
<html lang="en-US">
<head>
    <meta charset="utf-8" />
    <meta name="keywords" content="key, words" />
    <meta name="description" content="description" />
    <link rel="stylesheet" href="stylesheet.css" type="text/css" />
    <link rel="icon" href="favicon.ico" type="image/x-icon" />
    <script type="text/javascript" src="behavior.js"></script>
    <title>Application Name</title>
</head>
<body>
    <div id="appCommonComponents"> ... </div>
    <div id="appSceneContainer"> ... </div>
</body>
</html>
```

Figure 23. General container of a HTML5 transcoded application.

The <body> tag contains two sub-containers for including the rest of the application components. The first corresponds to the components that are common for all the scenes of the application such as the background and the navigation elements. The second includes the structure of the application scenes and its components. This separation enables an efficient management of the scene switching and facilitates the maintenance of the application for adding and removing scenes if necessary.

Component Matching

The implementation of the transcoder requires knowing how to transform each DVB-PCF component (source) into HTML5 (target). Table 7 summarizes the clustering with the main DVB-PCF components depending on its transformation complexity. A direct translation implies that there is a one-to-one matching between the source and target components. When this is not possible, the source component must be transformed into a set of target components with a common objective, which we called widgets.

The components with simple adaptation are transformed into an HTML tag and its CSS data in order to define the component style. As its definition and transformation are quite simple, it is quite likely that the implementation of this transformation will be regular. This behavior is not the same for components with a complex adaptation, which can be interpreted differently depending on the transcoder implementation. For example, a SpinControl component can be written and rendered in several ways for a web-based environment. In the following lines are described the DVB-PCF components that form complex widgets in a translation into HTML.

Main DVB-PCF components			
Visual Components		**Non-visual components**	
Simple (direct translation)	**Complex** (widget generation)	**Simple** (direct translation)	**Complex** (widget generation)
• Background • HintTextBox • Image • TextBox • InputComponents • Video	• Basic Shapes (Rectangle, Ellipse, etc) • Clock • ConnectStatusImage • ImageAnimated • ImageScalable • Menu • NumericNavigator • SpinControl • Subtitles • Ticker	• Audio • Cookie • CurrentTime • Random • Timer	• ReturnPath • Stream • StreamEvent

Table 7. DVB-PCF component clustering for generating HTML5 code.

The basic shapes components such as Rectangle, Line and Pixel can be translated within a Canvas element, which is part of HTML5 and is supported by the main web browsers.

The Clock, ConnectStatusImage, SpinControl and Ticker components are visual widgets implemented in HTML layers and managed by JavaScript functions.

The HTML5 Video tag allows the overlay of other HTML objects in a relatively straightforward way, so components such as Clock, Ticker, Subtitles and any other widget can be presented on the video without any plugin or additional code.

The ImageAnimated is a component used to display a simple sequence of images and ImageScalable displays a still image with control over its scaling and presentation. Both of them can be easily implemented with JavaScript functions to form a widget.

The Menu and NumericNavigation are navigation components explained below with the user interaction and navigation related issues.

The ReturnPath set of components provides control over the return path and its status when this communication path is not always available, as is the case with dial up or some satellite connections.

Finally, a Stream component controls the connection to a set of one or more elementary media streams, such as Video, Audio and Subtitles. Depending on its defined complexity, it can be translated directly into a Video tag component or, if advanced functions are required (e.g. subtitles), into a more complex widget. The Stream component also may contain StreamEvent components to provide media synchronization.

User Interaction and Navigation

In DVB-PCF, the user interaction is mainly defined through custom key events that activate a specific behavior, or by the component class default behavior – if the component includes predefined behavior. For example, the DVB-PCF Menu component reacts to the UP/DOWN user keys by moving its highlights over its menu items without the explicit service author description of this behavior. The custom events are divided in two parts: the trigger and the "to do" action. For user events, the trigger is a key-stroke event activated by the remote control that activates the behavior specified in the event node.

When translating to HTML5, this mechanism is implemented by defining the JavaScript onkeydown event and performing the key code matching between the defined DVB-PCF virtual keys (for example, the VK_COLORED_KEY_0 key code) and the specific platform key codes from the remote control. This matching is defined in a key matching configuration table, which is provided to the transcoder as part of the platform profile.

Mouse and touch events have also their corresponding Javascript functions to detect the user interaction (onclick, onmousemove, ondrag, etc). The transformation of these events does not require further configuration actions.

HTML5 does not define complex widgets such as the Menu component, so a direct translation is not possible. The most similar is the HTML5 <nav> tag, which defines a navigation section, though it does not include any predefined behavior associated. The Menu component allows the navigation and selection of one of a set of options presented in a visually coherent ordered list. The NumericNavigator component provides the navigation capability through the service using the numeric keys. In both cases, the component maintains an array of labels to specify the item order and, in the Menu component, it is also possible to define the looping behavior of the menu items through the menuLoop property. The translation of these components can be implemented in a similar way as the custom user key events by adding a JavaScript object to manage an identifier array of HTML nodes and by controlling its behavior through the specified component properties.

Finally, the scene navigation within the scene is mainly done by the translation of the HistoryNavigate component to the JavaScript History object and the SceneNavigate translation to the JavaScript Location object. While the former allows navigation through the document history stack the latter enables the transitions to any specified URL.

Content provider interaction

A particular case of the application behavior is the interaction with the content provider through a two-way return channel. It enables the acquisition of dynamic contents and the user participation on iTV services. The DVB-PCF return channel specification has been translated into an Asynchronous JavaScript and XML (AJAX) widget that encloses the typical behavior of the typical return channel functions.

The AJAX, as defined in [77], is a set of technologies used to build rich Web applications usually with improved performance, interactivity and user experience. It is based on:

- HTML/XHTML and CSS for the presentation;
- the Document Object Model (DOM) for representing and interacting with objects;
- XML and the eXtensible Stylesheet language Transformations (XSLT) for the interchange and manipulation of data between the server and client platform;
- the XMLHttpRequest object or equivalent for the asynchronous communications; and
- JavaScript for binding everything together [78].

AJAX enables the request and the display of specific contents without reloading the entire current page. It reduces the amount of required data for each transaction and enhances the user experience. Moreover, the reduction in bytes of HTTP responses is significant, over 56% average according to [79]. This saves transmission time between server and client, reduces the required bandwidth, and improves the response time.

For a translation from DVB-PCF into a Rich Internet Application, making an intensive use of AJAX, the DVB-PCF transcoder generates a unique HTML file for the application root. It contains:

- the appropriate AJAX requests to apply the suitable scene changes and manages the related dynamic data transfers with the server;
- the first scene of the service; and
- the set of components shared by all the scenes (e.g. header logo).

When a dynamic data transfer is needed, the DVB-PCF description includes a transaction component and its behavior. Application developers are free to specify the return path related items as shared components as well as scene specific components. The rest of the scenes are translated to the necessary JavaScript code that embodies the scene items in HTML and the functions to update the specific DOM elements.

When a change of scene is requested, the components of the previous scene are replaced with the new ones to avoid that the memory size of the client platform increases unpredictably. In order to complete the deployment of the generated service, the application server must:

- include the suitable request attention module to process the dynamic data request;
- access the system information (if needed); and
- return information related to the client STB.

This last part of the service implementation is highly dependent on the application server technology, but by applying AJAX techniques, it can be fully independent from the application layout, where the specification of the user experience of the iTV service resides.

Content synchronization

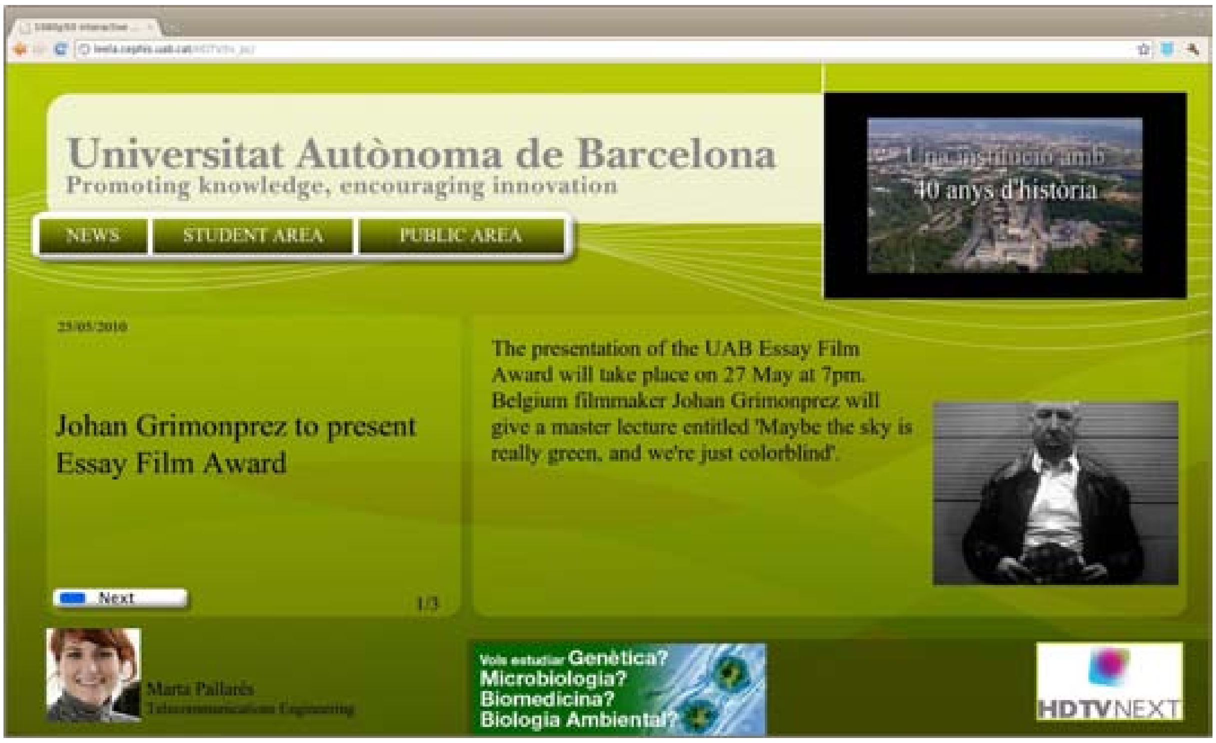
Content synchronization in HTML5 can be purely implemented in JavaScript or SMIL. While the former requires a custom implementation for the management of the timeline (i.e. firing events at a specific time, listen to component events, etc), the second provides a global timeline that is a time reference for every component of the user interface. SMIL has no natural competitors as markup language for describing time-based presentations, so that any other solution would rely on scripting. However, the flexibility of a scripting language such as JavaScript enables animations that are not feasible with SMIL.

For translating DVB-PCF descriptions into HTML5, the simplest solution was chosen and the DVB-PCF Stream component (which can include video, audio and StreamEvent components) was translated into a SMIL container with its own timeline.

The iTV application developed for this use case includes multimedia synchronization for presenting the news that are related to the TV program consumed by the user. When a specific program starts, the related StreamEvent calls a procedure that presents the related news on the screen.

Remarks

This first experiment demonstrated the viability to transform cross-platform DVB-PCF descriptions into web-based applications. Browser capabilities facilitate the implementation of features related to the content provider connection and Internet browsing. HTML5 browsers take also the responsibility of some application layers such as the video codec management and the application timeline (thanks to SMIL). Figure 24 presents a capture example of the HTML5 application obtained from the described transcoding process and shown in a desktop browser.


Universitat Autònoma de Barcelona
Promoting knowledge, encouraging innovation
NEWS
STUDENT AREA
PUBLIC AREA
Una institució amb 40 anys d'història
Next
1/3
Marta Pallarès
Telecommunications Engineering
Vols estudiar Genètica? Microbiologia? Biomedicina? Biologia Ambiental?
HDTVNEXT

25/05/2010

Johan Grimonprez to present Essay Film Award

The presentation of the UAB Essay Film Award will take place on 27 May at 7pm. Belgium filmmaker Johan Grimonprez will give a master lecture entitled 'Maybe the sky is really green, and we're just colorblind'.

Figure 24. Example of the HTML5 application obtained from the DVB-PCF transcoding.

4.1.3. Transcoding DVB-PCF for Procedural Runtime Environments: MHP

All transcoders implemented through the methodology presented in this work share the parsing of DVB-PCF descriptions for generating the memory model that describes the entire application (i.e., a PCFModel instance). Each specific transcoder transforms the memory model into the target application format. This structure enables an easy addition of new transcoders to the system while minimizing the code required for implementing each transcoder. The main objective for the implementation of this transcoder was demonstrating the viability of generating procedural-based iTV applications with the proposed methodology. Next lines describe the transformation process.

Application Container

The conventional model for Java applications does not work well in iTV environments. The basic Java application model assumes that only one application is executing in a given virtual machine, and that the application itself is in complete control of its lifecycle. This is less than ideal for a digital TV receiver, where several applications may be running at the same time, and where there is a need to enforce some separation between the applications.

Java provides an alternative to conventional applications that is a good starting point for what a digital receiver need: applets. Unlike a normal application, applets can be started and managed directly by other software components (such as the web browser), and several of them can be executed in the same web page at the same time.

The digital TV world is not the same as the Web, and so some changes had to be made in order to adapt the applet model to digital TV receivers. The result is known as Xlets. These are a concept similar to applets that have been introduced by Sun in the JavaTV specification and adopted as the Java application format for MHP and related iTV standards. Like applets, the Xlet interface allows an external source to start and stop an application, as well as controlling it when application is running.

In order to meet these requirements, the application structure of an MHP application (hereafter Xlet) has to implement the javax.tv.xlet.Xlet interface. Figure 25 presents the Xlet container where any transcoded application is fitted.

```
public class ApplicationName implements javax.tv.xlet.Xlet
{
    public ApplicationName() {}
    public void initXlet(javax.tv.xlet.XletContext context)
        throws javax.tv.xlet.XletStateChangeException {}
    public void startXlet()
        throws javax.tv.xlet.XletStateChangeException {}
    public void pauseXlet() {}
    public void destroyXlet(boolean unconditional)
       throws javax.tv.xlet.XletStateChangeException {}
}
```

Figure 25. General container of a MHP transcoded application.

Like applets, Xlets have to include public methods that allow the application manager to initialize, start, and stop it. In contrast, Xlets can also be paused and resumed. The reason for this is very simple – in a STB environment several applications may be running at the same time, hardware restrictions mean that only one of those applications may be visible at any time. In this scenario, non-visible applications must be paused in order to keep resources free for the visible.

Component Matching

MHP uses the standard Home Audio Video Interoperability (HAVi) [80] for describing the graphic user interface. In particular, a subset called Level 2 User Interface, which includes the lightweight elements of the standard java.awt package. Table 8 shows the DVB-PCF component clustering for generating a MHP application.

Main DVB-PCF components			
Visual Components		**Non-visual components**	
Simple (direct translation)	**Complex** (widget generation)	**Simple** (direct translation)	**Complex** (widget generation)
• Background • Image • TextBox • InputComponents • Video • **Basic Shapes (Rectangle, Ellipse, etc)** • **Menu**	• Clock • ConnectStatusImage • ImageAnimated • ImageScalable • NumericNavigator • SpinControl • Subtitles • Ticker • **HintTextBox**	• Audio • CurrentTime • Random • Timer	• ReturnPath • Stream • StreamEvent • **Cookie**

Table 8. DVB-PCF component clustering for generating MHP code.
The differences with the analysis performed for HTML5 are highlighted in bold.

The main difference with HTML5 is that MHP does not provide some elements related to the browser engine and Internet browsing, such as the hint box (used for labeling elements) and the cookie. In contrast, HAVi provides basic shapes and menu components as basic graphic elements.

User Interaction and Navigation

Most MHP platforms will receive user input from a remote control. While input from a keyboard (and possibly a mouse) may be available, it's not very likely that this will be used in most cases. Table 9 shows the key events defined by the MHP receiver which may be generated by a remote control, extracted from the MHP 1.0.3 Specification [57].

Constant name	Key	Key code
VK_UP	up arrow	not standardized
VK_DOWN	down arrow	not standardized
VK_LEFT	left arrow	not standardized
VK_RIGHT	right arrow	not standardized
VK_ENTER	enter (also known as select or OK)	not standardized
VK_0 to VK_9	number keys	48 - 57
VK_TELETEXT	teletext key	459
VK_COLORED_KEY_0	first coloured key	403
VK_COLORED_KEY_1	second coloured key	404
VK_COLORED_KEY_2	third coloured key	405
VK_COLORED_KEY_3	fourth coloured key	406

Table 9. Key events that may be generated by an MHP receiver.

Key codes for VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT and VK_ENTER are all defined by the Java platform. While the remote control may generate other key codes as well, these are the only ones that are actually guaranteed to be available on every receiver. It is not advisable to rely in any other.

Before an application can receive user events, it must choose a UserEventRepository that defines the group of events that the application wishes to receive. Once the application has defined the set of keys it wishes to receive events for, it can use the EventManager class to request access to these events. This class follows a singleton pattern design to avoid multiple instances of the same object, which can be obtained by calling the EventManager.getInstance() method.

One of the problems arising from allowing applications to receive events without having any visible user interface component is to understand how the receiver will behave at any given time. One example could be the case when the user is just watching TV and the digit buttons stop working because they are actually being intercepted by an application instead of being used to change the channel. That spoils the user experience. So, users need to recognize whether they are watching TV or interacting with an application.

The MHP specification recommends that applications without an obvious user interface should only use the colored keys or the teletext button, and that all other buttons should be left for the navigator. This allows developers to build applications that are activated with a single key stroke (e.g. pressing the red button shows more information about the product shown on the screen and opens an interactive advertisement) without interfering with the rest of the TV system, which is what people expect. According to MHP recommendations, applications wanting to use other keys must display a user interface that covers at least 3% of the total screen area in order to allow the user to easily understand the application context that is enabled in each moment.

Content provider interaction

The interactive broadcast and Internet access profiles of MHP include support for using a return channel in the MHP receiver to communicate with the outside world. Connection session is managed through the ConnectionRCInterface class and connection parameters are stated in ConnectionParameters.

When transcoding DVB-PCF descriptions, server connection parameters are sometimes unknown. An application provider can generate MHP applications for multiple vendors without knowing the details of the server connection for the return channel. In order to overcome this issue, the transcoder generates a configuration file for the MHP application that can be manually modified to adapt the MHP application to any service provider platform.

Content synchronization

If an application is associated with a specific TV show, it's often useful to be able to synchronize the behavior of application to the action on screen. In many multimedia authoring systems such as SMIL, this is based around the concept of a timeline where the changes in behavior take place at a specific time.

One problem with broadcast iTV services is that there is not a global time reference for applications, at least when talking about a broadcast MPEG-2 stream. Since we can join the stream at any point, we have no way of knowing how long it is since the stream started (or even what the concept of 'starting' means in a broadcast sense). This means that we can't use media time as a mechanism for synchronizing applications to their associated media. Instead, we use some extra capabilities defined by DSM-CC to help achieving the synchronization required.

DSM-CC stream events are markers embedded in a transport stream via MPEG-2 private sections, with each marker consisting of both an identifier and a time reference. The identifier allows each stream event to be distinguished from others and the time reference indicates at what point in the stream the event should be triggered.

As well as the stream events themselves, the transport stream will also contain an object carousel formed by a set of stream event objects. These identify each event by a textual name, and allow the mapping of this name to the numeric identifier contained in the event itself. This can be performed with Normal Play Time (NPT), specified in the DSM-CC specification. An MPEG-2 stream may include an NPT timecode for a global time reference for the stream. While this timecode is separated from DSM-CC stream events, and is carried in its own descriptor, DSM-CC stream events can use this timecode to help decide when to trigger events.

From a DSM-CC perspective, stream events are split into two parts: stream event objects and stream event descriptors. DSM-CC stream event objects are stored in an object carousel, and are just like any other DSM-CC objects. A stream event object provides a general description of a stream event, which includes an event ID and a human-readable name. This allows a receiver to know the set of the events that can be generated in that stream, and helps it to check if an application is registering itself as a listener for events that actually exist.

The stream event descriptor is the second part of the conrnerstone. This is embedded in the MPEG stream as a marker, and tells the receiver that an event has actually been generated. A stream event descriptor contains three main attributes: the ID of the event, an NPT value at which the event should be generated, and some application-specific data. The ID allows the receiver to work out which stream event object is associated with this descriptor. Since a broadcaster can't be sure exactly where a descriptor is inserted into an MPEG stream, each descriptor carries an NPT value indicating when the event should be triggered. This allows the receiver to know in advance that it should generate an event when a specific NPT value is reached, giving a little more predictability. It also adds a little more reliability into the system, since the broadcaster can send several stream event descriptors with the same values in order to make sure at least one of them is decoded properly. The MHP specification says that for most stream events, they should be signaled at least once every second for a minimum of five seconds before the time they should trigger.

A stream event descriptor can also tell the system that the event should be triggered immediately - these are known as “do it now” events. This mechanism allows stream events to be easily inserted in to live TV shows. For instance, a sports application may use stream events to detect when a goal has been scored in a soccer match, and which team has scored. Stream event descriptors containing “do it now” events are only broadcasted once per event and so, the receiver has to make sure that it receives them properly.

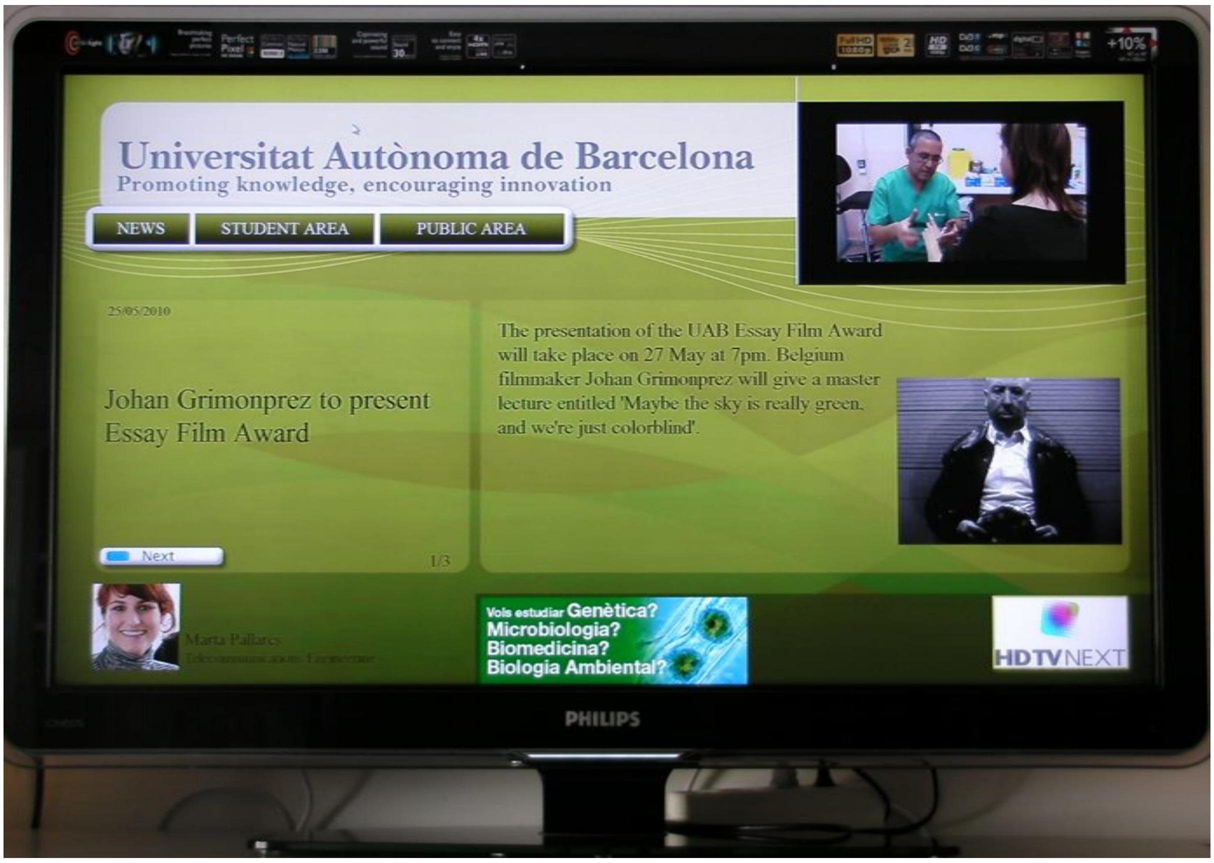
Whenever a stream event descriptor is received, the receiver takes the following steps:

1. It checks if any other event object with the same event ID is present in the default object carousel. If present, then the descriptor is ignored.
2. If the encoding of the descriptor shows that the event is a 'do it now' event, then the event is triggered immediately.
3. If the event is not a 'do it now' event, the receiver checks the NPT value at which the event should be triggered. If an event with the same event ID is already scheduled to be triggered at the same NPT value, or if the NPT value has already passed, then the event descriptor is ignored.
4. When the NPT value reaches the value specified for a scheduled event, the event is triggered.

One advantage offered by this separation of stream event objects and stream event descriptors is that events can be re-used. Several stream event descriptors can contain the same event ID, even if they are triggered at different times and contain different private data. This allows an application to use the event ID to define 'classes' of events.

When translating a DVB-PCF description into MHP, any StreamEvent component is converted to a DSMCCStreamEvent class in MHP and the application subscribes to that event when needed.

In the example for synchronizing news with the TV program consumed, the application creates two functions: an event listener for attending the changes on the TV program and an event consumption function for changing the news content. When a new TV program triggers the defined event, the receiver framework launches the event consumption function with the specific parameters for presenting the related news. Figure 26 presents a capture example of the MHP application obtained from the described transcoding process and shown in a HD TV at 1080 lines per frame.


Universitat Autònoma de Barcelona
Promoting knowledge, encouraging innovation
NEWS
STUDENT AREA
PUBLIC AREA
25/05/2010
Next
1/3
Vols estudiar Genètica? Microbiologia? Biomedicina? Biologia Ambiental?
HDTVNEXT
PHILIPS

Johan Grimonprez to present Essay Film Award

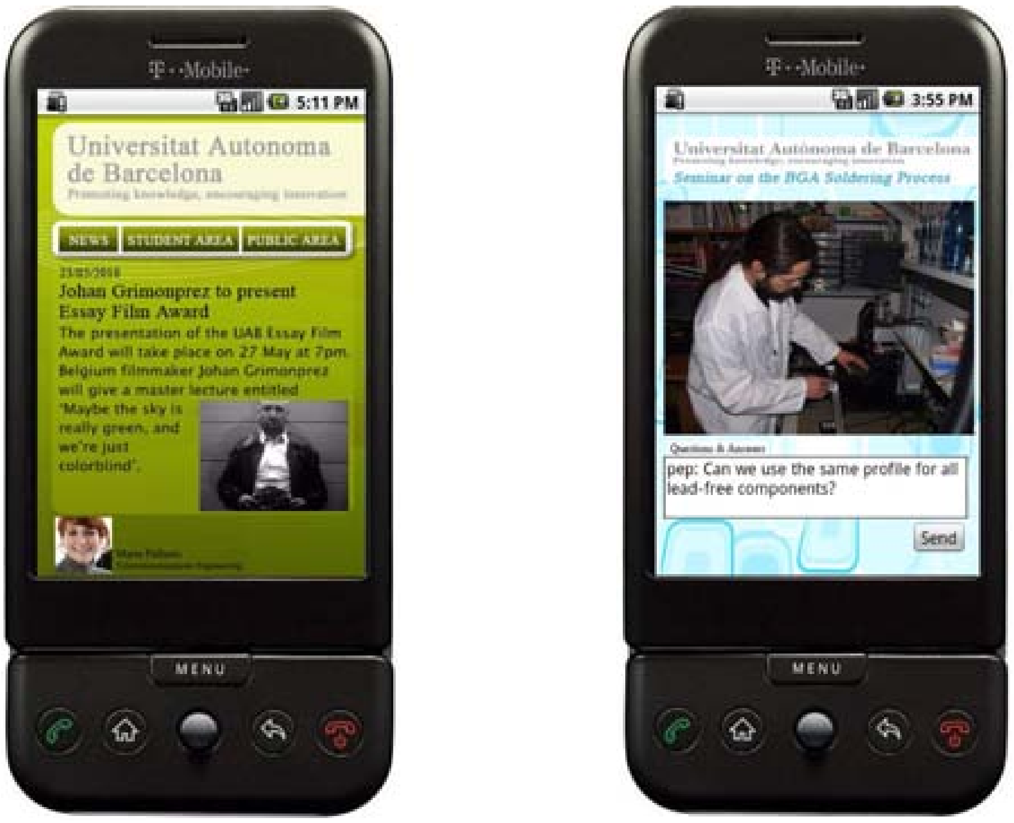
The presentation of the UAB Essay Film Award will take place on 27 May at 7pm. Belgium filmmaker Johan Grimonprez will give a master lecture entitled 'Maybe the sky is really green, and we're just colorblind'.

Figure 26. Example of the MHP application obtained from the DVB-PCF transcoding.

Remarks

This second experiment yields some findings: it showed the details from transcoding from DVB-PCF to MHP and the differences with the HTML5 process. The lack of a browser engine and the simplicity of the MHP framework for defining graphic user interfaces require more code effort for building a transcoder. Moreover, implementing synchronization in MHP is not as easy as in HTML5 and requires more expertise for developing synchronized applications. The MHP application architecture is more complex than HTML5 and its learning curve can be slower. In contrast, some graphic elements are easier to implement in MHP than in HTML5, such as basic shapes and some menu controls.

In addition, HTML5 is supported in a wider range of platforms than MHP, which facilitates the deployment of the generated web-based applications to different devices (such as mobile phones) with minimum changes for being adapted to each device. The implementation of progressive enhancement techniques in the application development reduces considerably the required effort for adapting the application to different devices but increases the application code by the device feature detection. This feature detection enable the application to provide its mobile version when a user request the access from a mobile device. Figure 27 shows two simple examples of mobile web applications obtained from the translation of DVB-PCF descriptions. The first capture corresponds to the case described in this section and the second is another example application that was developed to explore more content synchronization cases.



Universitat Autonoma de Barcelona
Promoting knowledge, encouraging innovation

NEWS | STUDENT AREA | PUBLIC AREA

23/05/2010
Johan Grimonprez to present Essay Film Award
The presentation of the UAB Essay Film Award will take place on 27 May at 7pm. Belgium filmmaker Johan Grimonprez will give a master lecture entitled 'Maybe the sky is really green, and we're just colorblind'.

Universitat Autònoma de Barcelona
Promoting knowledge, encouraging innovation
Seminar on the BGA Soldering Process

Questions & Answers
pep: Can we use the same profile for all lead-free components?
Send

(a) (b)

Figure 27. Mobile web applications obtained from DVB-PCF descriptions.

Regarding MHP, no further studies have been carried out with this format in this research due to the current deployment status of this specification.

4.1.4. Results & Partial Conclusions

DVB-PCF has enabled an easy way to apply the concept “write once, adapt to anywhere” on the development of iTV applications. The architecture and lightweight definition of this standard provide the required flexibility for designing cross-platform iTV applications and its transcoding into platform-specific formats such as HTML5 and MHP.

4.2.Validation of the Proposed Development Environment

The set of developed tools for applying the proposed methodology have been validated through the generation and transformation of an eHealth application for iTV. This application is based on the access to a personal health record management service, provided by MAG S.L. [82], in the context of a R&D project[5]. Most of user interface components and interface behavior of that application were designed using VisualPCF and translated into HTML5 with the built-in transcoder. The current section describes that process.

4.2.1. Use case description: Personal Health Record Management for iTV

This iTV application enables the access to a personal health management service (Info33) that aims at improving the user access to personal medical information from its own home. The service also enables the interaction with professional health personnel for medical consultations and report generation. The TV platform consists of a Linux-based set-top box that runs the XBMC media center and the built-in iTV application, which establishes a secure connection to the service for transmitting data in XML format and enables the user management through a TV-based graphic user interface. In addition, the platform enables user multimodal interaction to facilitate the accessibility to anybody. Figure 28 shows the architecture of this system.

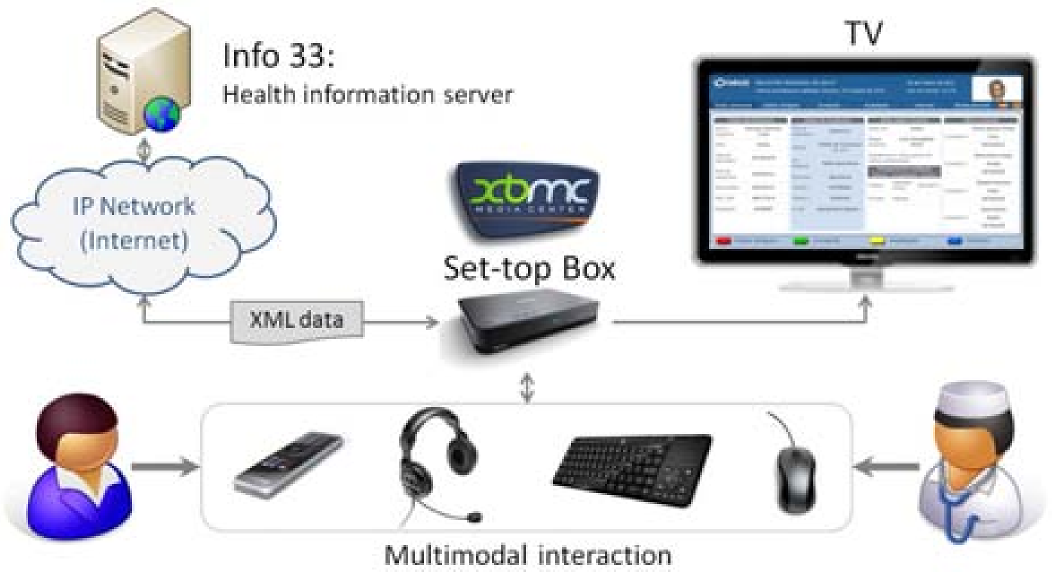


Figure 28. Architecture of the eHealth system developed for TV.

[5] “La eSalud al servicio del ciudadano”. AVANZA TSI-020302-2009-85. Spain, 2011.

Key technical features of that iTV application:

- Layout for high-definition screens (HD 720 and HD 1080).
- Information management divided into tabbed scenes.
- Multimodal interaction: Remote, voice, keyboard and mouse.
- Play local and remote video sources.
- Get XML data from a remote web service.

The iTV application was built as a cross-platform solution in order to facilitate future deployments on different iTV systems. However, due to limitations of the implemented tool, the critical parts of the application such as the speech recognition, the Web service communication and the XML data parsing were developed later.

4.2.2. Development of a cross-platform iTV application: VisualPCF

The following section describes the basic steps for building an iTV application using VisualPCF, based on the presented use case.

Background

When creating a new project with VisualPCF, the environment creates a new DVB-PCF application and a new black panel for starting the edition of common components of the user interface. No scenes are created yet.

One of the first elements that can be added to the application is the background image that is shared by all the scenes. In the “common components” scene (which is not a real scene), the Background item is selected in the palette and a dialog enables the selection of a local image file. When loaded, the image is copied locally in the project folder and presented in the editor. **¡Error! No se encuentra el origen de la referencia.** outlines this step.

Static Layout

The layout is mainly structured through layout containers, which may have components and other containers. A Container component is the equivalent to the <div> container in HTML interfaces. At the time of writing, the layout is only based on a fixed positioning for defining positions and sizes of elements. Hence, these parameters have to be configured for each visual element of the application. Any user interface component such as text, image and menu can be included as common components for defining the overall look and feel of the iTV application. The properties for each element can be modified by visually selecting the element and editing the Properties panel. **¡Error! No se encuentra el origen de la referencia.** outlines this step.

Navigation

Navigation can be defined by adding a link to any existent visual component that points to a different scene of the iTV application or to a remote source such as the content provider. Special navigation components are the menu items. VisualPCF provides two classes of menu items: Menu and TabbedMenu. Menu defines a vertical list of navigation items that change the presented scene. Similarly, the TabbedMenu presents a horizontal group of navigation items that are rendered as tabs in the application for presenting different scenes or remote content. Most iTV user interfaces contains a unique menu component to facilitate user navigation with the remote controller.

The configuration of the navigation component includes the definition of the target scenes. The scenes can be defined as application scenes (content is defined within the application) or remote content. When referencing a new application scene VisualPCF generates the code structure for the new scene and the navigation links between scenes. Remote content is defined through a URN identifier to enable the connection to a remote server which has to provide the content for that particular scene in a suitable format. **¡Error! No se encuentra el origen de la referencia.** presents an example of the definition of a tabbed menu component and its scene configuration. The result is shown with some added style properties for a better presentation.

Dynamic information

VisualPCF enables the definition of dynamic content in different ways:

- Creating a new scene for showing remote content (already explained): This content needs to be formatted in the server side. The application directly injects the received code without parsing or validating it. This is useful for presenting preformatted code in the iTV application such as web pages or predefined widgets.
- Creating a Container component with a reference to the remote content: Similarly to the previous case but for a specific component instead of a scene.
- Referencing components: Most VisualPCF visual components allow the configuration of a reference parameter. When stated, this parameter informs the application that the content for that component has to be acquired from a remote location.
- Defining external data sources: VisualPCF enables the definition of an external data source for each scene, including common components. This means that developers can configure the location where the application will obtain the specific data for each scene when the user access to it. This way usually reduces the number of external connections for acquiring dynamic information, which can be later referenced by the user interface components. The application expects an XML data file for a better cross-platform definition.

For the present use case, a specific external data source was defined for each scene. The common components include the time values for the last update of the application and the last connection to the system of that specific user.

Video

Video elements can be included to the user interface in two modes:

- Local/remote video file: Video included as a single file that usually requires the loading of the entire file before its playback. It is useful for video elements that do not change often.

- Video stream: Video defined as an elementary stream and included within a transport stream, which also may include related audio streams and subtitles. This component is usually stated for acquiring broadcasted signals or Internet streams.

The present use case included a video file presenting an avatar who welcomes the user and helps with the navigation across the iTV application. **¡Error! No se encuentra el origen de la referencia.** presents the configuration panel of VisualPCF for inserting video content and the result on the editor.

Transcoding

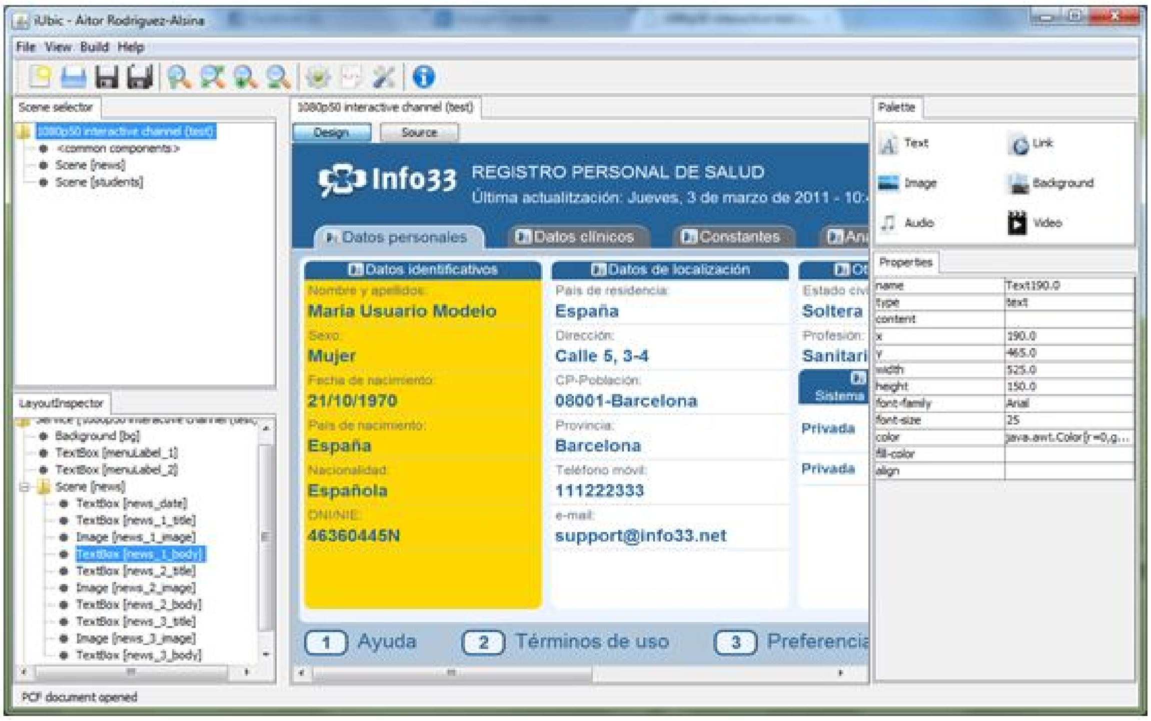
The transcoding parameters of the VisualPCF project can be configured at any time before the project compilation. The environment also enables the addition of new target profiles for translating the DVB-PCF description into different versions of the iTV application according to platform-specific requirements. **¡Error! No se encuentra el origen de la referencia.** presents a screenshot of the transcoding configuration panel in VisualPCF.

Different target profiles can be configured for the same target application format but for different reference screen sizes. In the example, the user interface is optimized for HD 720 (1280x720) and HD 1080 (1920x1080) displays. The scaling process is performed by the PCFScaler module, previously described in Section 3.2.5.

Checking the generated application

Once the DVB-PCF description is transcoded into a target application format, the generated code is stored in the distribution folder of the VisualPCF project. The application can be then distributed or provided with no dependencies with VisualPCF or the DVB-PCF description. Before that, the generated application can be visually validated within a suitable test environment. In case of web-based applications, this validation can be usually performed in a desktop browser taking into account that the server connection parameters have to be properly stated in the application configuration XML file for dynamic data transfers.

Figure 29 shows a screenshot of the designed application in VisualPCF and Figure 30 shows the transcoded application within a desktop browser. Due to the limitations of the current version of VisualPCF, some manual changes were required on the generated application for deploying it in a production system.



- 1080p50 interactive channel (test)
  - <common components>
  - Scene [news]
  - Scene [students]

- Background [bg]
- TextBox [menuLabel_1]
- TextBox [menuLabel_2]
- Scene [news]
  - TextBox [news_date]
  - TextBox [news_1_title]
  - Image [news_1_image]
  - TextBox [news_1_body]
  - TextBox [news_2_title]
  - Image [news_2_image]
  - TextBox [news_2_body]
  - TextBox [news_3_title]
  - Image [news_3_image]
  - TextBox [news_3_body]

name	Text190.0
type	text
content	
x	190.0
y	465.0
width	525.0
height	150.0
font-family	Arial
font-size	25
color	java.awt.Color[r=0,g...
fill-color	
align	

PCF document opened

Info33 REGISTRO PERSONAL DE SALUD
Última actualització: Jueves, 3 de marzo de 2011 - 10:

Datos personales | Datos clínicos | Constantes | Ana

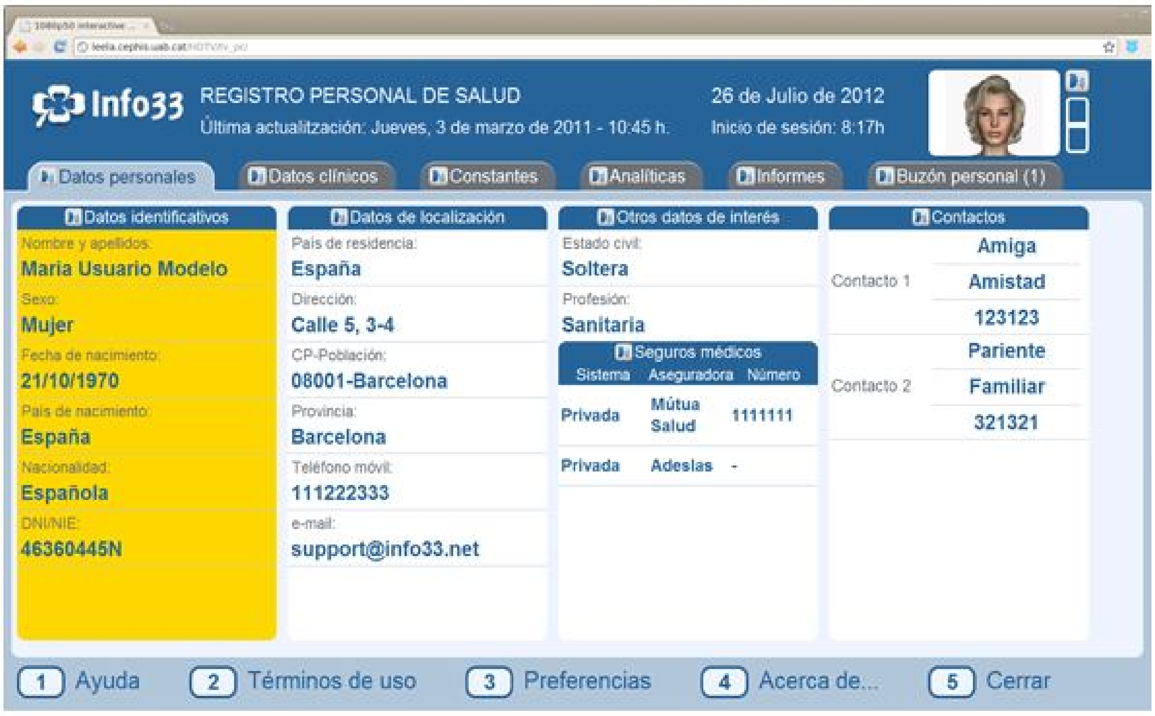
Datos identificativos: Nombre y apellidos: Maria Usuario Modelo; Sexo: Mujer; Fecha de nacimiento: 21/10/1970; País de nacimiento: España; Nacionalidad: Española; DNI/NIE: 46360445N

Datos de localización: País de residencia: España; Dirección: Calle 5, 3-4; CP-Población: 08001-Barcelona; Provincia: Barcelona; Teléfono móvil: 111222333; e-mail: support@info33.net

Estado civil: Soltera; Profesión: Sanitari; Sistema; Privada; Privada

1 Ayuda 2 Términos de uso 3 Preferencia

Figure 29. Result of an iTV application design with VisualPCF.



Info33 REGISTRO PERSONAL DE SALUD
Última actualización: Jueves, 3 de marzo de 2011 - 10:45 h.
26 de Julio de 2012
Inicio de sesión: 8:17h

Datos personales | Datos clínicos | Constantes | Analíticas | Informes | Buzón personal (1)

Datos identificativos
Nombre y apellidos: Maria Usuario Modelo
Sexo: Mujer
Fecha de nacimiento: 21/10/1970
País de nacimiento: España
Nacionalidad: Española
DNI/NIE: 46360445N

Datos de localización
País de residencia: España
Dirección: Calle 5, 3-4
CP-Población: 08001-Barcelona
Provincia: Barcelona
Teléfono móvil: 111222333
e-mail: support@info33.net

Otros datos de interés
Estado civil: Soltera
Profesión: Sanitaria

Seguros médicos

Sistema	Aseguradora	Número
Privada	Mútua Salud	1111111
Privada	Adeslas	-

Contactos

Contacto 1	Amiga Amistad 123123
Contacto 2	Pariente Familiar 321321

1 Ayuda 2 Términos de uso 3 Preferencias 4 Acerca de... 5 Cerrar

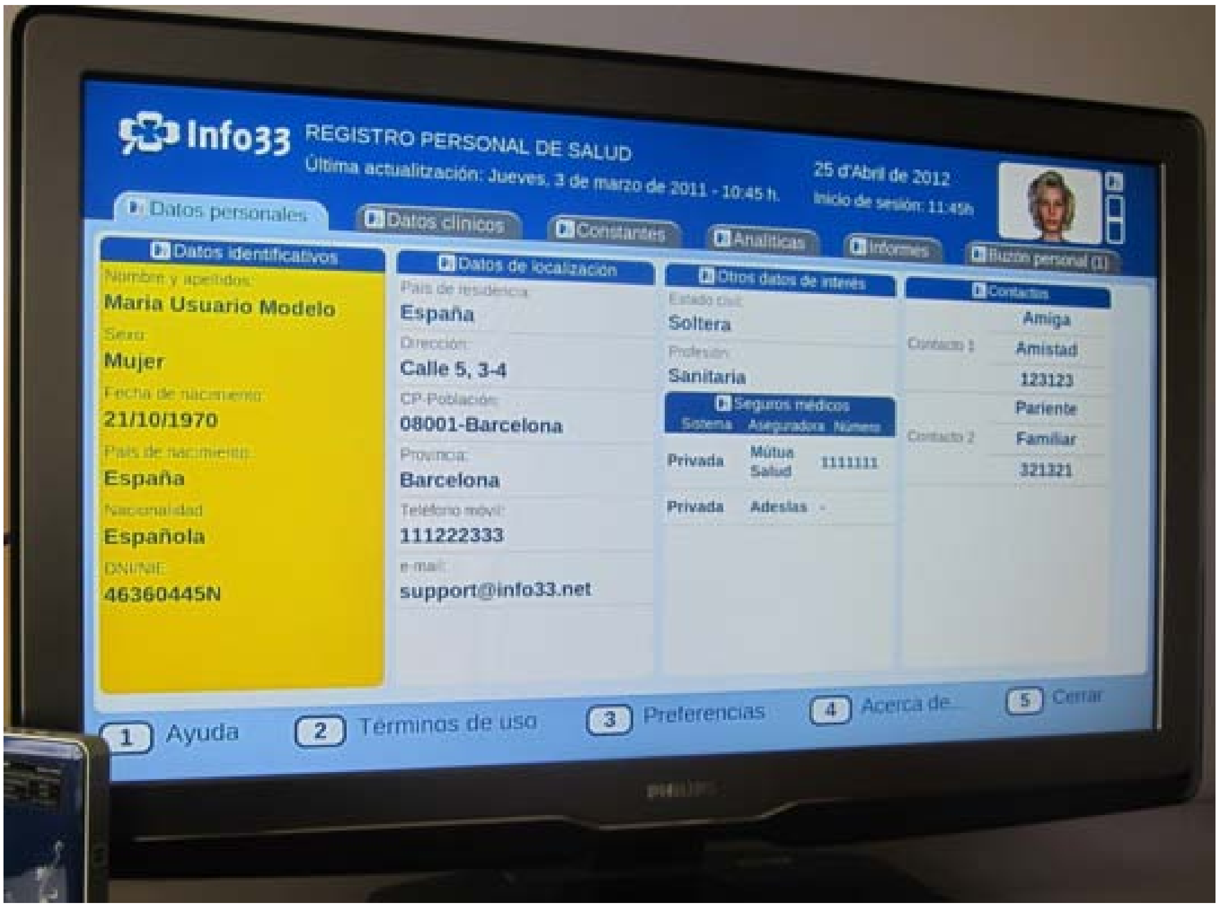
Figure 30. Browser validation of the application generated with VisualPCF.

In case of MHP applications the validation step requires a MHP testing platform or emulation software for PC such as XleTView [83].

4.2.3. Results & Partial Conclusions

VisualPCF has enabled the development of cross-platform applications for iTV from authoring to code generation for a specific platform. The user interface editor has demonstrated its flexibility for designing basic iTV interfaces, but it would be desirable to add more complex iTV widgets (such as clock, ticker or even an application launcher like the one used in MHP environments, among others). The creation of an authoring tool was necessary to improve the development process in DVB-PCF and bring this standard to non-expert developers and content editors. However, this approach could be improved by using an existing authoring tool for generic user interfaces and transform the generated code into a DVB-PCF description that can be integrated in the proposed system for generating cross-platform iTV applications. This point is one of the short-term future challenges for the present research.

The application use case discussed in this section provided the implementation details for generating web-based applications in a cross-platform way. The same application has been generated for two different screen sizes by simply calling the scaling module described in section 3.2.5. The resulting application for HD1080 screens is shown in Figure 31.



Última actualización: Jueves, 3 de marzo de 2011 - 10:45 h.

25 d'Abril de 2012
Inicio de sesión: 11:45h

Datos identificativos
Nombre y apellidos: Maria Usuario Modelo
Sexo: Mujer
Fecha de nacimiento: 21/10/1970
País de nacimiento: España
Nacionalidad: Española
DNI/NIE: 46360445N

Datos de localización
País de residencia: España
Dirección: Calle 5, 3-4
CP-Población: 08001-Barcelona
Provincia: Barcelona
Teléfono móvil: 111222333
e-mail: support@info33.net

Otros datos de interés
Estado civil: Soltera
Profesión: Sanitaria

Seguros médicos

Sistema	Aseguradora	Número
Privada	Mútua Salud	1111111
Privada	Adeslas	-

Contactos

Contacto 1	Amiga
	Amistad
	123123
Contacto 2	Pariente
	Familiar
	321321

Figure 31. The eHealth application presented on TV.

4.3. Validation of the Approach for Synchronizing Subtitles Across Web Platforms

The proposed approach for video subtitling in Ambient Intelligence environments was validated through its implementation into three different use cases that demonstrated its accuracy:

- The SVG Subtitle Access Bar: Extends the functionality of a HTML5 <video> container for web applications onto desktop browsers.
- Positioning subtitles on Interactive TV: Offers a solution for video subtitling in web-based interactive TV environments;
- Presenting subtitles on a Second Screen: Makes use of mobile devices to present personalized subtitles on a TV's second screen.

4.3.1. The SVG Subtitle Access Bar

The SVG Subtitle Access Bar (SVG-SAB) is a web widget that extends the functionality of any HTML5 <video> component to enable advanced subtitle functions. It facilitates the access to video content on Web applications by the adaptation of subtitles to user needs and preferences. This widget applies the proposed solution to synchronizing subtitles and video content on the client side using SMIL animations. This keeps contents synchronized even if the user dynamically changes the language, the subtitle text format and the position of the text on the screen. Although it can be implemented on any web application, the main goal of SVG Subtitle Access Bar is the experimentation with video subtitles for research purposes. It enables the study of the optimal format of subtitles according to the context requirements and user needs. All features can be changed dynamically through a web interface, which means that the widget can be controlled by a third-party application that manages the user and environment variables. This management may enable the dynamic adaptation of the subtitle text color depending on the contrast with the video content and the environment global lighting, the automatic language change according to the browser defined language or the split of both sources (video and subtitles) for presenting one or both of them separately on a second screen. As both sources are merged in the client side, unlike the traditional subtitled video contents, they can work separately but synchronously in the same application environment. This enables, besides the customization of the subtitle format, the positioning of subtitles beyond the video window even beyond the main screen.

Figure 32 shows a snapshot of the SVG-BAR attached to a video component playing a subtitled video. Subtitle languages can be selected and switched dynamically from remote .SRT files. Each subtitle is added as a DOM node with its corresponding presentation time, and SMIL manages these times in order to show each piece of text on the screen. To maintain the content synchronization developers need not take into account neither the current playback time nor the subtitle sequence order.

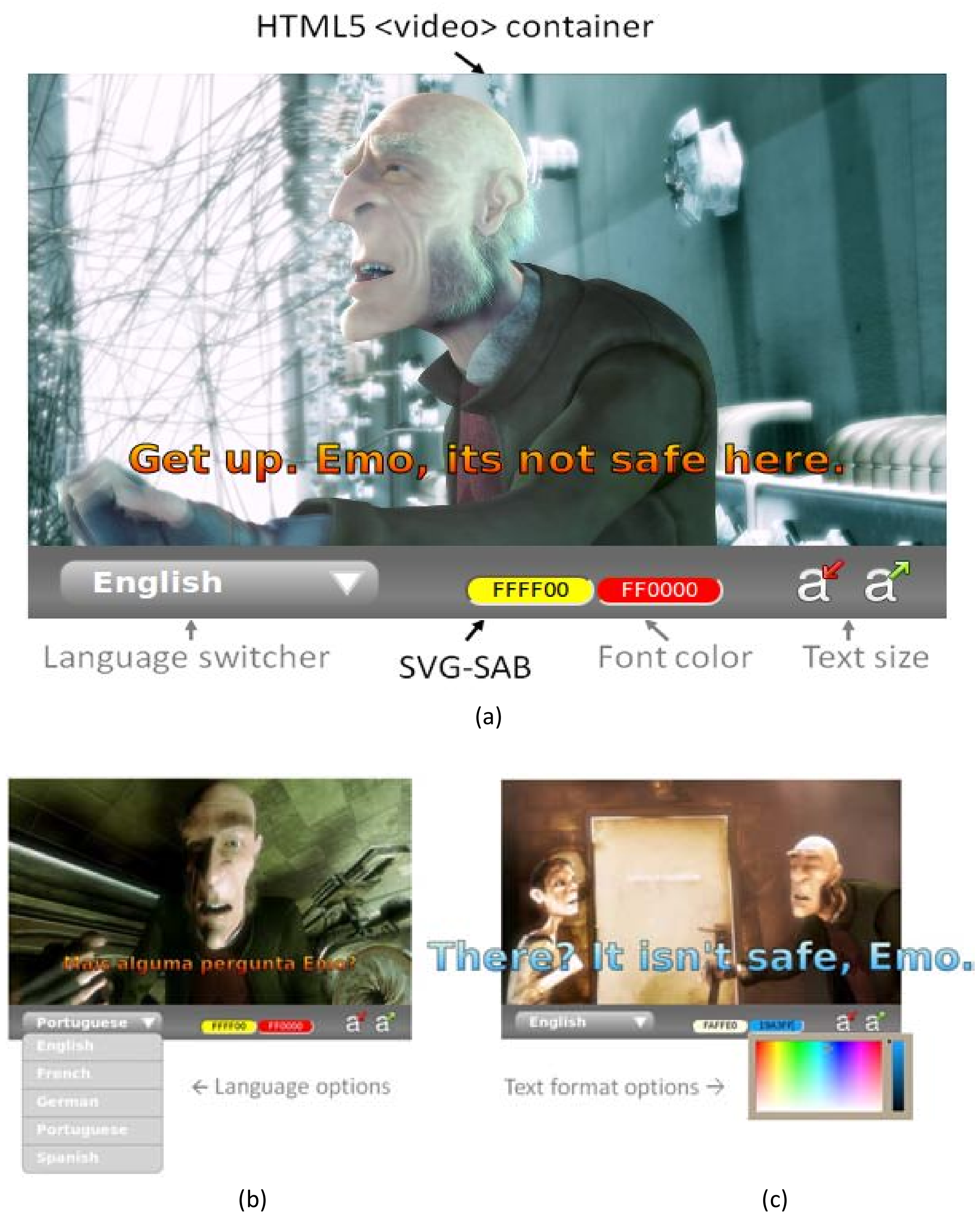


Figure 32. The SVG Subtitle Access Bar presented on a desktop browser.
(a) Presents the overall features of the video component widget. (b) Example of the dynamic language switching function. Different languages can be shown during the video playback without synchronization failures. (c) Example of the dynamic subtitle customization. The text color and gradient can be changed through a built-in color palette and the text can be resized manually. Size can be extended beyond the video window if necessary.

When a language switch is applied, the widget removes the previous subtitles and adds new ones without synchronization delay because the SMIL timeline is the responsible of presenting each subtitle at the correct time. The SVG-SAB widget also enables modifications on the text color and text size of the subtitle in order to enable its adaptation to the requirements of the video, the application, the user and the context, e.g. text color can be adapted to achieve a better contrast depending on the content and context lighting. Adaptation of the subtitle can also be necessary to overcome user difficulties in accessing content. The use and functionality of these subtitles is multiple: for language teaching and language therapy [84], for people with visual impairments or color blindness, for the elderly and people who speak other languages, or for those with literacy or comprehension difficulties [85][86].

The content used for the experimentation in this research as well as the film shown in the screenshots presented in this paper is “Elephants Dream” (2006) [87], an animation open movie film initiated by Blender Foundation to promote the free software application for animation made by the same foundation.

4.3.2. Positioning Subtitles on Interactive TV

On TV, subtitle customization enables better access to video content, especially within the environment of interactive TV. Subtitles can be placed and sized properly to ensure maximum readability, while the video content remains playing in an interactive user interface. Subtitle position can also be modified in order to place it wherever is most comfortable for the user [88]. This feature has been applied to construct an interactive application for TV that presents video subtitles that are always readable even if the video window is reduced in order to display the rest of the interactive contents. Due to the time management of the SMIL browser engine, content is always synchronized even if it is visually separated on the screen.

A proper subtitle positioning can also be generated automatically by the video component when it is resized to be accommodated to the safe area on the screen by an interactive application. As the enhanced video component is the responsible for managing and presenting the subtitles, the interactive application does not need know the details of presenting subtitles beyond accommodating them to ensure a correct visualization while the user is browsing the application.

Figure 33 presents an example of interactive application for TV that implements this smart widget for presenting video subtitles on connected TVs. The video is being played on TV and the subtitles are presented in the language selected by the user. When the user accesses to the interactive application the video component is resized and it is accommodated within the interactive interface, then the widget detects automatically that it has been resized and presents the subtitles accordingly. This is a very simple function but outlines part of the capabilities and flexibility achieved by the proposed solution for presenting subtitles on smart environments.

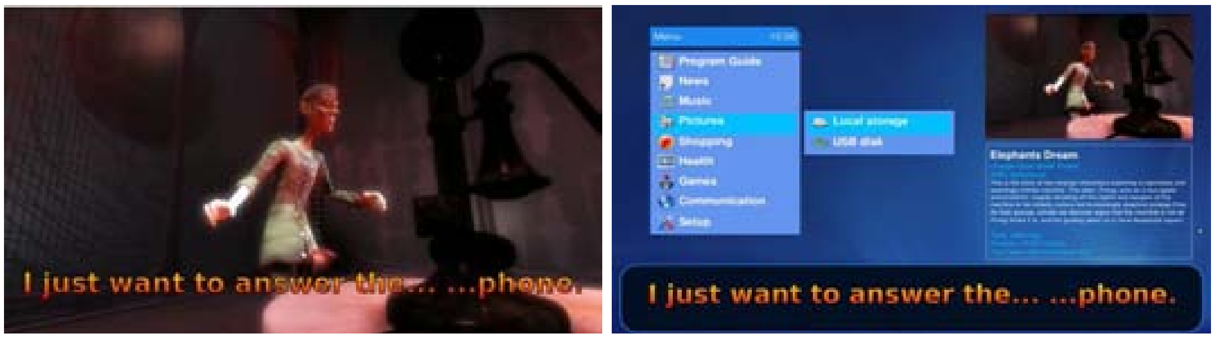


Figure 33. Positioning of subtitles within an interactive TV environment.
In (a), the user is watching a subtitled video content on the TV. He then accesses the interactive element and the video playback is reduced to a corner of the screen as shown in (b). Here, subtitles are presented in normal size and positioned in the same place as before.

These environments surrounded by sensors and smart applications for detecting the user needs, are only feasible if its components collaborate through common APIs that facilitate its integration in the whole system. This is the case of web components and applications, which are based on common standards, unlike native applications written for specific platforms.

4.3.3. Presenting Subtitles on a Second Screen

Within the home environment, a mobile device can be used as a second screen of the TV platform onto which to extend its content and management options [89]. Mobile devices can become an advanced remote controller, as well as the support for presenting the same contents as on the TV set or some complementary contents. The access to video content can benefit from this feature as it allows the personalized access to video subtitles by each user through their own smart mobile phone. This can facilitate the access to content for people with special requirements (e.g. the elderly, children and disabled people) and thus improves the family balance. Out of the home environment, this solution can be applied on public spaces as theaters, museums and conferences for providing subtitles or complementary information to all users and it is language independent. This complementary information can be permanently synchronized with the main contents and enhances the accessibility not only of a specific content but of the whole smart environment. On public smart spaces, a multi-screen solution like this can facilitate the access to the culture to people with special requirements. In theaters, for instance, a film or a play can be followed by people who do not understand the spoken language accessing to the related subtitles through its own mobile smart device regardless of the device operating system. A mention has to be done regarding the production of these subtitles for live events or plays where some part of improvisation is expected. Although the subtitle production in those cases is usually not an easy task, and though we take this into account it is out of the scope of this research. Our proposal claims that while subtitles are produced they can be sent to multiple (and different) devices and they can be presented without synchronization problems with the main content.

We have built a web-based mobile application to present video subtitles that are synchronized with TV contents, extending the TV screen to a mobile device. In a home environment, this mobile application connects to the media center, which serves the TV interactive application with the synchronized video subtitles (shown in the previous section), through the wireless local area network. The user receives the subtitles of the current video content on his mobile device and can change the language, size and color of subtitle texts independently from the TV stream or any other connected device. This mobile web application applies the same SMIL subtitle synchronization principle proposed in this paper. The main difference with regards to implementation from the previously highlighted cases is the synchronization requirements between the media center and the user device acting as a second screen. In our implementation, the user device application receives the current playback time from the media center, which acts as server. Media Center and device SMIL timelines are then synchronized and the mobile browser engine can present each subtitle at the correct time. This enables the extension of multimedia contents to all connected devices that are available in the user environment, and it can be integrated on Ambient Intelligence environments through its communication with the sensors and applications that make possible the automatic recognition of the user needs.

Figure 34 shows an example for accessing from an Android 2.2 platform to the TV contents for receiving subtitles as complementary information.

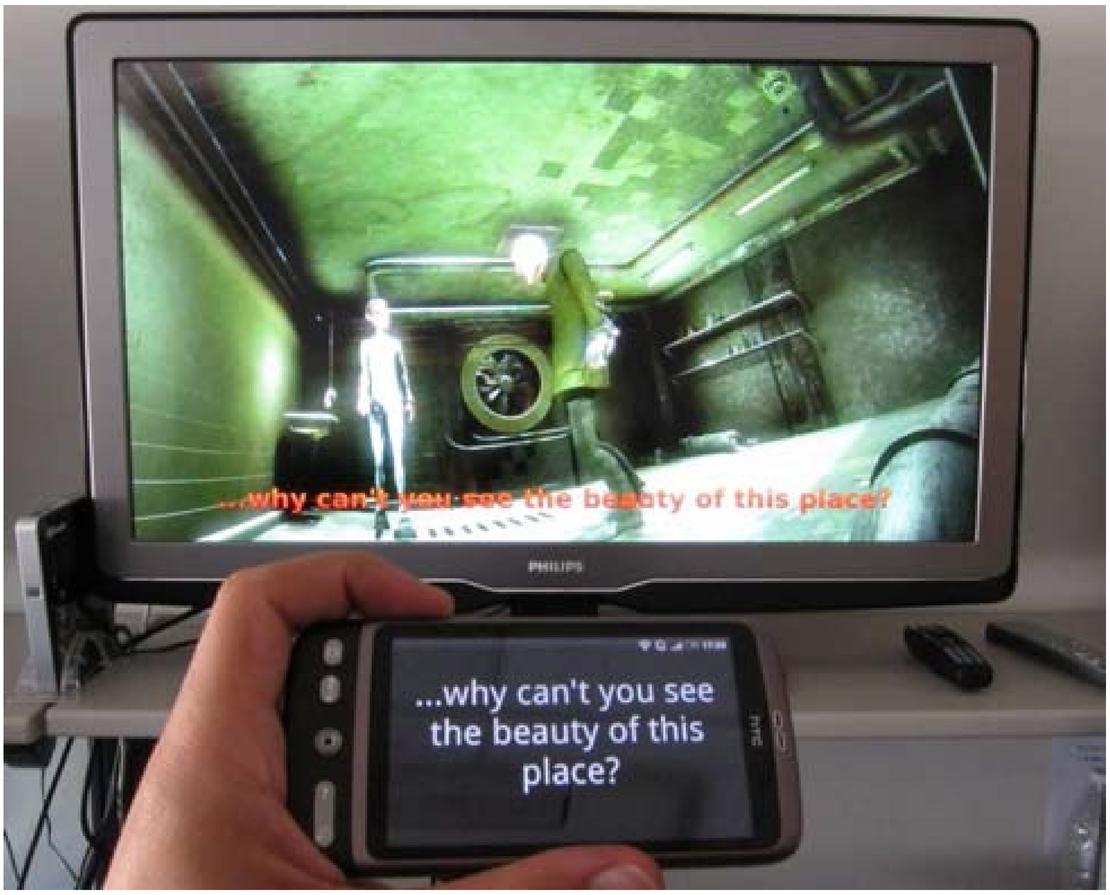

...why can't you see the beauty of this place?
PHILIPS
...why can't you see the beauty of this place?
htc

Figure 34. Example of second screen application for subtitles on an Android 2.2 platform.

If the environment can detect the user position, the system can send subtitles (or the whole movie) to those connected who leave the room, who can receive contents in a personal device. Detecting ambient noise, the system can send subtitles to users when it is difficult to hear the audio.

4.3.4. User Experience Evaluation

The user experience achieved by the proposed solution for subtitle synchronization across multiple screens has been evaluated in order to validate its acceptability by end-users. In the evaluation, users watching TV contents had to get complementary contents in smartphones. In this case, the proposed task was to watch a short movie while reading subtitles in their own smartphones to understand the storyline of the presented fragment. The main objective of this experiment was the validation of three aspects:

- Platform-independency: The device agnosticism of the proposed solution, letting the subjects to bring their own smartphones for the experiment. Although this fact could provide unpredictable results due to the loose of control over the variables that can affect the user platform (e.g., device manufacturer, operating system, selected web browser and any other installed function that can affect to the running test), the experiment with user's smartphones provide a real test of what is expected of a service for public spaces. A secondary reason for taking this decision was the familiarity users may have with smartphones, avoiding the possible usability issues that users without a previous training on this kind of platforms can experiment;
- Viability: The viability of smartphones as secondary screens based on the proposed solution and its acceptability for using it at home or theaters;
- Synchronization: the synchronization quality between the main content (in this case, on TV) and the secondary devices accessing to complementary information (in this case, smartphones accessing to subtitles).

Evaluation Methodology

In order to drive this subjective evaluation, some participants were recruited as volunteers to act as subject of the experiment. A total of 15 subjects were selected, 7 female and 8 male, aged between 24 and 53 years old. A basic questionnaire was previously presented to participants in order to characterize the sample of subjects and get some information about their habits regarding audiovisual contents, their visual acuity (a basic test just to check that they can understand the visual contents at the viewing distance) and their previous knowledge degree using smartphones. Before this preliminary questionnaire, subjects were asked to bring their own smartphone to the experiment session.

Almost all participants (86.6%) answered that they consume audiovisual contents (i.e, any kind of video content) every day either on TV, mobile device or desktop computer. They all use daily the smartphone to access one or more Internet services (e.g., email, news, social networks, browsing the web and others) and most of them (73.3%) sometimes do it when they are watching TV at home. Some of them (46.6%) consumed subtitled audiovisual content regularly, but the percentage rise (80.0%) when asked if they would consume more subtitled films if they were easily available. The smartphone they use were based on Android (60%), iPhone (33.3%) and Windows Phone (6.6%).

For the experiment, subjects were asked to watch an animation open movie film of 15 minutes called “Sintel” (2010) [90], which was initiated also by Blender Foundation. The movie was presented with sound disabled to force the reading of subtitles on smartphones.

Subjects tasks consisted in: (1) using the smartphone to connect to the local network and access to the application for subtitles; (2) watching the movie freely and read the subtitles on the smartphone to understand the storyline; (3) when first subtitle appears, subjects must change the default language (Russian) to their mother tongue; (4) answer a questionnaire regarding their satisfaction and the service quality.

The questions presented were based on a 5 grades Likert scale [90] to obtain answers from “Strongly disagree” to “Strongly agree”.

Evaluation Results

The results of the user experience evaluation, shown in Figure 35, are divided in 3 questions for quality and 3 more for acceptability and user satisfaction. The chart shows that the mobile web application used by subjects to get subtitles was simple enough to accomplish the required tasks. The connection and access to the application was easy and there were not remarkable issues with the application usage. Some comments were related to the language change menu, which can be improved for a better visibility. Regarding subtitle synchronization, some subjects detected synchronization mismatches in their smartphones when changing language because their mobile browser required too much time for importing the new language file (this operation requires intensive DOM management). This function can be optimized to run properly on devices with low resources without these synchronization issues. However, the language change operation was quick and easy for most of the subjects (and smartphones) of the study.

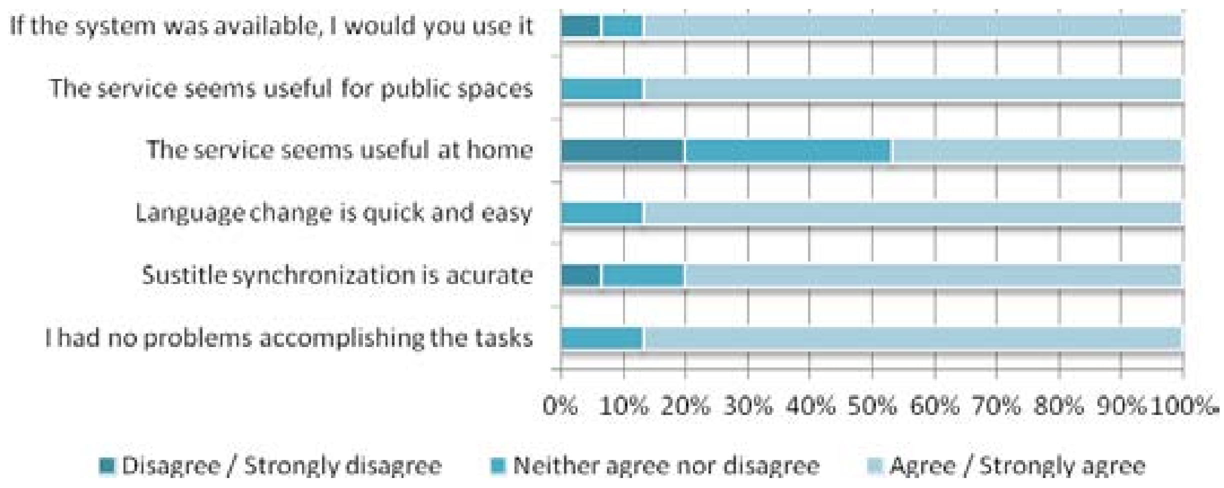


Figure 35. Results of the user experience evaluation for second screen systems.

The acceptance of a mobile device for accessing subtitles at home has been differently rated. On the one hand, although it was seen a useful function under certain situations, some subjects did not feel the need of using this service at home under standard conditions. The use of more devices to accessing complementary content must be required by the context because it can be seen as unnecessary and obtrusive when not need. This is the case of subtitles, which are usually presented on TV instead of a mobile device. On the other hand, the acceptance of the service was different when applied on public spaces like theaters, museums and the street. It seems that there are more situations in public spaces where this service is perceived as useful or needed. Finally, subjects mostly agreed that they would use this subtitling service if it was available, especially in public places as the previous question showed.

4.3.5. Results & Partial Conclusions

This approach has demonstrated to be practical and adapted properly across different devices. The conducted evaluation provides information about the context where this service is perceived as useful and may focus the attention of application and service developers. Beyond the subtitling service, the acceptability of smartphones as multi-screen devices has not been questioned.

4.4.Summary of the Chapter

This chapter has presented the experiments driven for validating the three main proposals of this research: (1) a methodology for generating cross-platform applications for iTV; (2) a software framework for applying and validating the proposed methology; and (3) a solution for synchronizing subtitles using HTML5.

The proposed methodology has been validated through the implementation of two different transcoders for the standard DVB-PCF. Firstly, the process for translating cross-platforms iTV applications (written in DVB-PCF) into declarative applications (written in HTML5) has showed the basic steps for implementing a transcoder. Secondly, the transcoding into a procedural language such as MHP has evidenced the differences between procedural applications based on a platform-specific runtime environment and declarative applications that are based on a browser.

The proposed software framework has been used for creating a second cross-platform application based on an eHealth service for TV. The use of VisualPCF as authoring tool speeds the development of cross-platform user interfaces. Section 4.2 describes the development of the presented use case using this tool, which includes the rest of the implemented modules for managing DVB-PCF descriptions.

Finally, the proposal for synchronizing subtitles on web-based platforms has been analyzed through its implementation on three use cases related to iTV applications. The most representative case, focused on second screens, has been also validated through a user experience evaluation. Results have demonstrated that SMIL time features enable an efficient management of subtitles across different HTML5 platforms.

CHAPTER 5

CONCLUSIONS

"I am turned into a sort of machine for observing facts and grinding out conclusions."

Charles Darwin

This last chapter summarizes the results achieved in this research and provides some lines for further improvements.

5.1.General Conclusions

This dissertation discusses the current context for the development of cross-platform applications on interactive TV (iTV) environments and proposes a suitable methodology that unifies the application design for multiple platforms and devices. This methodology has resulted in the definition of a production architecture that enables the deployment of cross-platform applications in multiple iTV networks and facilitates the business to business interchange by using a standard content format: DVB-PCF. This has provided the main contribution of the present dissertation.

DVB-PCF has demonstrated to be a flexible an efficient format for describing interactive contents in a portable way. This format is a high-level description of the expected user experience of an interactive application that avoids the implementation details for facilitating its translation into different platform-specific languages.

This work presents the key factors for the development of two different DVB-PCF transcoders based on the proposed methodology. The first transcoder, focused on HTML5 and declarative application environments, allows discussing the basic steps for the implementation of a generic DVB-PCF transcoder and the specific issues for the case of HTML5 applications. Platform-specific formats based on browsers as runtime environment such as HTML5 can take advantage of the browser capabilities and its continuous evolution. The set of HTML5 features that are currently supported by the main browsers enables the development of web-based applications with similar capabilities than other platform-specific formats while providing a more portable solution. Moreover, HTML5 takes advantage of the browser built-in capabilities for working with Internet applications, which facilitates the DVB-PCF transcoding of interactive applications for hybrid TV environments (i.e. those that combine broadcasted and Internet contents). The second transcoder, focuses on MHP and procedural application environments, evidences the differences between procedural applications based on a platform-specific runtime environment (such as those written in MHP) and declarative applications that are based on a browser (such as those written in HTML5). The complexity of the application programming interfaces provided by MHP for implementing hybrid TV solutions may hinder some way the development of iTV applications and the transcoding from DVB-PCF when performed by non expert MHP developers. One of the reasons for this is that, while browsers are being continuously updated with new features and programming facilities (due to the progressive evolution of HTML5), MHP defines a monolithic platform with static profiles that are not always aligned with the features and facilities expected by developers. A second reason is that MHP was designed to run on low-profile consumer devices (STBs) unlike HTML5 that aims to provide a multiplatform solution for developing applications for all kind of devices. Luckily for most application developers, MHP is being discarded in many iTV networks as the solution for deploying interactive services. However, the industry and related organizations should learn from the mistakes of the MHP experience to provide technological solutions with a good compromise between performance and interoperability.

The second contribution of this dissertation is a software framework for applying the proposed methodology for building cross-platform iTV applications. The software modules and the integrated development environment developed allow developers and content editors to dispose of an easy authoring of cross-platform applications written in DVB-PCF and its translation into platform-specific ones. This framework has been used for creating a cross-platform application based on an eHealth service for TV. The use of VisualPCF as authoring tool has demonstrated the viability for visually create “write once, adapt to everywhere” applications for iTV. The tool was developed for research purposes and it just provides the basic resources for building simple iTV user interfaces and enabling further experiments in this area. However, with some additional development efforts, it could be easily improved to become an efficient authoring tool for real production systems.

Finally, this dissertation provides a specific contribution for synchronizing video subtitles across different HTML5 platforms. This novel approach is based on SMIL and reduces considerably the code required by an application for managing time issues. This feature has been used for the synchronization between video and subtitles in three different use cases on the iTV context. This proposal has been validated through a user experience evaluation. Results demonstrate that SMIL time features enable an efficient management of subtitles across different HTML5 platforms without losing the synchronization between the presentation components.

In conclusion, the development of pure cross-platform consumer applications is a huge challenge that has to deal with the industry of multimedia technologies. The undeniable success of application markets such as Google Play and AppStore has provided an attractive business model for application developers and multimedia service providers. Moreover, this deployment model has largely set the direction of the application development for many consumer devices (PC, mobile phones, tablets, TV, game consoles, etc). Although a significant part of this development is performed through platform-specific technologies, HTML5 is being quickly adopted for more and more multimedia systems, especially on the Web. This (partially) new standard democratizes the application development for web-based environments in a variety of devices. However, the coexistence of different application formats requires solutions in the direction of the present dissertation in order to facilitate the cross-platform development and application portability.

5.2.Future Research

This dissertation addresses the problem of generating cross-platform applications for interactive TV, but nowadays, most multimedia environments have this same problem. One of the main lines for continuing this research is the application of the ideas presented here to other application areas different from interactive TV. DVB-PCF was built for describing iTV applications but the concept of iTV application can be easily understood as interactive software with intensive use of multimedia content and synchronization requirements. With the forthcoming irruption of multi-screen and 3D technologies on the consumer market, this kind of applications will become more common even for services not directly related with traditional TV. The natural evolution of multimedia environments comes also with the war around platforms for consumer devices (mobile, TV, game consoles, etc). The platform diversity that is still expected in this context for the next years argues for the improvement of the existing cross-platform techniques and tools.

Regarding the integrated development environment (VisualPCF), further improvements could take two directions:

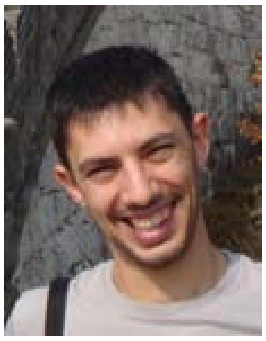
1. Continue improving the set of features provided by the application. More widgets, visual edition resources, code completion and a huge set of new capabilities can be added to the application for enhancing the developer experience and content editor facilities.
2. Import user interfaces instead of creating a new authoring tool for DVB-PCF descriptions. There are a number of authoring tools that can export application descriptions in a declarative programming language. The idea is to enable VisualPCF to import user interfaces from a different format and adapt it to the proposed DVB-PCF workflow.

In general, the transformation between declarative formats can be accelerated with eXtensible Stylesheet Language Transformations (XSLT), which is a XML-based language for performing code transformations between XML documents. This can be applied on different steps of the proposed transformation methodology when source and target formats are based on XML.

The work around subtitle synchronization on web-based platforms has been validated for a small group of users in a testing environment. The main idea behind the work with subtitling was to provide suitable solutions for enhancing the accessibility on public spaces, where the expected number of users can be significantly bigger. The presented solution can be applied on theaters, museums and conferences for providing personalized subtitles for large number of attendants. However, prior to the deployment on those real environments, it is required to drive further user experiences with a larger number of users that connect simultaneously to the subtitling service.

APPENDICES

APPENDIX A. AUTHOR'S BIOGRAFY



Aitor Rodriguez-Alsina is software engineer since 2006. He is currently working for Center of Ambient Intelligence and Accessibility of Catalonia (CAIAC). He participates in conferences and university courses to teach on different subjects such as Data Bases, new trends on Web development and Multimedia Accessibility. He also participates in the review process of several international conferences and journals as a reviewer, work which enable him to be in touch with the recent scientific production.

APPENDIX B. AUTHOR'S KEY PUBLICATIONS

Journals

[1] 2012 – "Subtitle Synchronization Across Multiple Screens and Devices". A.Rodriguez-Alsina, G. Talavera, P. Orero and J.Carrabina. In Sensors Open Access Journal. ISSN 1424-8220. Special Issue "Selected ppapers from UCAmI 2011 – the 5th International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'11)".

Book Chapters

[1] 2011 – "Analysis of the TV Interactive Content Convergence and Cross-Platform Adaptation". A. Rodriguez-Alsina, J. Carrabina. In the book "TV Content Analysis: Techniques and Applications". Publisher: CRC Press, Taylor Francis LLC. ISBN: 978-1-4398-5560-7.

Conference Papers

[1] 2011 – "Analysis and Design of a Subtitling System for Ambient Intelligence environments". A.Rodriguez-Alsina, G. Talavera, P. Orero and J.Carrabina. In 5th International Symposium on Ubiquitous Computing & Ambient Intelligence (UCAmI'11). 5-9 Dec. Riviera Maya, Mexico.

[2] 2011 – "Enhancing Accessibility through Speech Technologies on AAL Telemedicine Services for iTV". H. Delgado; A. Rodriguez-Alsina; A. Gurguí;, E. Martí ; J. Serrano ; J. Carrabina. In Proceedings of the 2nd International Joint Conference on AmI 2011, Amsterdam, The Netherlands, November 16--18, 2011.

[3] 2010 – "Unified Content Design for Ubiquitous Learning: The Soldering Seminar Use Case", A. Rodriguez-Alsina, E. Cespedes-Borras, Roger Puig-Fargas, M. Moreno-Berengue, J. Carrabina. Proceedings of the 4th IEEE International Conference on E-Learning in Industrial Electronics (ICELIE) 7-9 Nov. Phoenix, AZ, USA.

[4] 2010 – "Address Generation Unit for Multimedia Applications on Application Specific Instruction set Processors", M. Moreno-Berengue, G. Talavera, A. Rodriguez-Alsina, J. Carrabina. Proceedings of the 36th Anual Conference on IEEE Industrial Electronics Society. 7-9 Nov. Phoenix, AZ, USA.

[5] 2010 – "PCF Dynamic Data Transfers for Web-Based iTV Services", A. Rodríguez-Alsina, M. Moreno-Berengue, E. Cespedes, J. Carrabina. 2010 NEM Summit. Towards Future Media Internet. 13-15 Oct. Barcelona, Espanya.

[6] 2010 – "Interactive TV application design convergence across formats and devices", A. Rodriguez-Alsina, M. Moreno-Berengue, J. Carrabina. Proceedings of the ACM EuroITV. 9-11 Jun. Tampere, Finland.

[7] 2009 – "Platform-independent Web-Based iTV Services Through DVB-PCF Transcoding", A. Rodriguez, M. Moreno, E. Cespedes-Borras, J. Carrabina. Proceedings of the IEEE IC-BNMT 2009. 18-20 Oct. Beijing, China.

[8] 2009 – "Adding Tags to Courses to Improve Evaluation - A Multiplatform LCMS Approach that Allows Multidimensional Analysis", Eduard Cespedes-Borras, Aitor Rodriguez, Jordi Carrabina, Javier Serrano. CSEDU 2009: 92-97.

[9] 2008 – "Caracterització de plataformes HW-SW per a interactivitat en Televisió Digital amb MHP", A. Rodriguez, J.Carrabina, E. Teruel (UAB). IEnter Conference, Jornades sobre entreteniment audiovisual interactiu. 28-29 Feb. Caixaforum, Barcelona, Spain.

REFERENCES

REFERENCES

[1] Jens F. Jensen, Interactive Television - A Brief Media History, Proceedings of the 6th European conference on Changing Television Environments, July 03-04, 2008, Salzburg, Austria.

[2] Carey, J.: Winky Dink to Stargazer: Five Decades of Interactive Television. In: I-TV 1996, Edinburgh.

[3] Lee Becker, "A Decade of Research On Interactive Television" in William Dutton et al (eds.) Wired Cities: Shaping The Future of Communications, pp 102-123. Boston: G.K. Hall & Co., 1987.

[4] Van Tassel, J.M.: Advanced Television Systems. Focal Press, Boston (1996).

[5] Swedlow, T.: Interactive Enhanced Television: A Historical and Critical Perspective (2000), http://www.itvt.com/etvwhitepaper.html.

[6] Krantz, M.: Marriage of Convenience, in Time, November 10 (1997).

[7] Van Dusseldorp: SMS TV: Interactive Television Reinvented, van Dusseldorp & Partners (2002).

[8] Hao Zheng, Tuija Aalto, Jorma Kivelä, Artur Lugmayr, Erik Bäckman, Jere Hartimaine, Sauli Niskanen, Timo Manninen, and Juha rapeli. 2011. Predicting TV in the year 2013. In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11). ACM, New York, NY, USA, 295-299.

[9] Nielsen: Double Vision – Global Trends in Tablet and Smartphone Use while Watching TV. http://blog.nielsen.com/nielsenwire/online_mobile/double-vision-global-trends-in-tablet-and-smartphone-use-while-watching-tv/ (April 2012)

[10] Nielsen: Cross-platform is the New Norm. http://www.nielsen.com/us/en/insights/reports-downloads/2011/cross-platform-is-the-new-norm.html (October 2011)

[11] Law, E., Roto, V., Hassenzahl, M., Vermeeren, A., Kort, J.: Understanding, Scoping and Defining User Experience: A Survey Approach. In Proceedings of Human Factors in Computing Systems conference, CHI’09. 4–9 April 2009, Boston, MA, USA (2009).

[12] J.P. Evain, “The Multimedia Home Platform”, EBU Technical Review, Spring Issue, 1999.

[13] ISO. "ISO/IEC 13522-5:1997 - Information technology -- Coding of multimedia and hypermedia information -- Part 5: Support for base-level interactive applications".

[14] Clarasó JA, Baldo D, Benelli G, Daino GL and Zambon R. Interactive Digital Terrestrial Television: The Interoperability Challenge in Brazil. International Journal of Digital Multimedia Broadcasting, 2009.

[15] Hybrid Broadcast Broadband TV, ETSI Standard TS 102 769.

[16] Open IPTV Forum (OIPF), 2010, http://www.openiptvforum.org. Accessed on June 12, 2012.

[17] Tru2way, http://www.tru2way.com/. Accessed on June 12, 2012.

[18] Boxee, http://www.boxee.tv/.

[19] "Boxee See Internet TV Moving Away From The Computer To Mobile And Connected TV". Worldtvpc.com. December 29, 2011.

[20] Google TV, http://www.google.com/tv/.

[21] "Google TV 2.0 gains Honeycomb, Android Market", Clint Boulton, October 2011, http://www.linuxfordevices.com/c/a/News/Google-TV-2-rolls-out/.

[22] Microsoft Mediaroom, http://www.microsoft.com/mediaroom/.

[23] TVBLOB, http://www.tvblob.com/.

[24] Ubuntu TV, http://www.ubuntu.com/devices/tv.

[25] "CES 2012: free Ubuntu TV has service and revenue fees", Jack Schofield, January 2012, http://www.zdnet.com/ces-2012-free-ubuntu-tv-has-service-and-revenue-fees-4010025161/.

[26] XBMC, http://xbmc.org/.

[27] MeeGo for Smart TV, https://meego.com/devices/smart-tv.

[28] Yahoo Connected TV, http://connectedtv.yahoo.com.

[29] TiVo, http://www.tivo.com/.

[30] Motorola, Video Consumer Premises Equipment, http://www.motorola.com/Video-Solutions/US-EN/Products-and-Services/Video-Consumer-Premise-Equipment.

[31] Infonetics Research, Set-Top Boxes and Subscribers Quartely Market Size, Share, and Forecasts, March 2012.

[32] Echostar, http://www.echostar.com.

[33] Pace, Set-top Boxes, http://www.pace.com/global/our-portfolio/set-top-box/.

[34] Cisco, Home Solutions, http://www.cisco.com/web/consumer/products/prodcat-tvsets.html.

[35] Technicolor, http://www.technicolor.com/.

[36] MythTV Open Source DVR, http://www.mythtv.org/.

[37] Freevo Home Theater Platform, http://freevo.sourceforge.net/.

[38] Moovida Media Player, http://www.moovida.com/.

[39] Windows media Center, http://windows.microsoft.com/en-US/windows/products/windows-media-center.

[40] Linux Media Center (Linux MCE), http://www.linuxmce.org/.

[41] E. Tsekleves, et al, “Semi-automated Creation of Converged iTV Services: From Macromedia Director Simulations to Services Ready for Broadcast”, Journal of Virtual Reality and Broadcasting, Volume 4, no.17, 2007.

[42] Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster, UIML: An Appliance-Independent XML User Interface Language, Proceedings of the Eighth International World-Wide Web Conference, 1999.

[43] MPEG-7. Multimedia Content Description Interface. ISO/IEC 15938, 2001.

[44] Specification for a Lightweight Microbrowser for interactive TV applications, based on and compatible with WML, ETSI TS 102 322.

[45] Kulkarni Vijay and Anupama Nandeppanavar, “Interfacing Social Networking Sites with Set Top Box”, ADVANCES IN COMPUTING AND INFORMATION TECHNOLOGY, Communications in Computer and Information Science, 2011, Volume 198, Part 1, pp. 460-471.

[46] Nested Content Language 3.0 Part 8 – NCL Digital TV Profiles, 2010, http://www.ncl.org.br.

[47] Digital Video Broadcasting; Portable Content Format Specification 1.0, ETSI TS 102 523.

[48] W3C, Scalable Vector Graphics, 2010, www.w3c.org.

[49] HTML5, W3C Working Draft, 24 June 2010, Ian Hickson, http://www.w3c.org/TR/html5/.

[50] W3C, The Extensible HyperText Markup Language (Second Edition), 2010, www.w3c.org.

[51] W3C, Extensible Markup Language 1.0 (Fifth Edition), 2008, http://www.w3.org/TR/REC-xml/.

[52] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[53] IETF RFC 2141: "URN Syntax".

[54] M. Driver, R. Valdes, and G. Phifer. Rich Internet Applications Are the Next Evolution of the Web. Technical report, Gartner, May 2005.

[55] StatCounter Global Stats. Top 8 Mobile Operating Systems from May 2011 to May 2012. http://gs.statcounter.com. Accessed on June 13, 2012.

[56] OpenCable Application Platform Specification, OCAP 1.0 profile, CableLabs, 2004.

[57] Digital video broadcasting (DVB); Multimedia Home Platform (MHP) specification 1.0.3, ETSI Doc. No. ES 201 812 V1.1.1, 2003.

[58] Digital video broadcasting (DVB); Multimedia Home Platform (MHP) specification 1.2.2, ETSI Doc. No. TS 102 727 V1.1.1, 2010.

[59] Digital video broadcasting (DVB); Globally Executable MHP (GEM), ETSI Doc. No. TS 102 819 V1.2.1, 2004.

[60] Prado, E., Franquet, R., Soto, M.T., Ribes, X., Fernández Quijada, D., "Tipología funcional de la televisión interactiva y de las aplicaciones de interacción con el televisor" en Zer, Vol. 13, núm. 25, 2008, pp. 11-35.

[61] iSuppli: Smartphones to Account for Majority of Cellphone Shipments by 2015. http://www.isuppli.com/Mobile-and-Wireless-Communications/News/Pages/Smartphones-to-Account-for-Majority-of-Cellphone-Shipments-by-2015.aspx.

[62] BBC R&D White Paper #134. Portable Content Format: a standard for describing an interactive digital television service. 2010, http://www.bbc.co.uk/rd/pubs/whp.

[63] Van Deursen, D., De Keukelaere, F., Nachtegaele, L., Feyaerts, J., Van deWalle, R., "A scalable presentation format for multichannel publishing based on MPEG-21 digital items". Lecture Notes in Computer Science, vol. 4105, pp. 650–657. Springer, Heidelberg, September 2006.

[64] H.-T. Chiao, F.-C. Chen, K.-S. Hsu, and S.-M. Yuan, "Video everywhere through a scalable IP-streaming service framework," in Proceedings of the 3rd International Symposium on Wireless Communication Systems (ISWCS '06), pp. 190–194, September 2006.

[65] Portable Content Format Java Tools (PCF-JTools). 2010, http://code.google.com/p/pcf-jtools/.

[66] Lam, T.C., Ding, J.J., Liu, J.C. "XML Document Parsing: Operational and Performance Characteristics". Computer 2008 vol.41, pp 30-37.

[67] S. Pfeiffer and C. Parker. Accessibility for the HTML5 <video> element. Proceedings of the 6th International Cross-Disciplinary Conference on Web Accessibility (W4A'09), Madrid, Spain, 20-21t April 2009, pages 98-100.

[68] Matroska. SRT Subtitles. http://www.matroska.org/technical/specs/subtitles/srt.html.

[69] Gerber, J. Library Jquery.srt.js. http://v2v.cc/~j/jquery.srt/.

[70] Dale, M.; Metavid.org. SRT text tags with languages and categories. http://metavid.org/w/extensions/MetavidWiki/skins/mv_embed/example_usage/sample_timed_text.php.

[71] Le Hegaret, P. HTML5 DFXP Player prototipe, W3C. http://www.w3.org/2008/12/dfxp-testsuite/web-framework/START.html.

[72] Timed Text (TT) Authoring Format 1.0 – Distribution Format Exchange Profile (DFXP), W3C Candidate Recommendation, 16 November 2006. http://www.w3.org/TR/2006/CR-ttaf1-dfxp-20061116/.

[73] Danilo, A. Lights, Camera, Action. Proceedings of the 7th International Conference on Scalable Vector Graphics, Mountain View, CA, USA, 2-4 October 2009, http://svgopen.org/2009/papers/3-Lights_Camera_Action/#d4e69.

[74] Rodriguez-Alsina, A.; Talavera, G.; Orero, P; and Carrabina, J. Analysis and Design of a Subtitling System for Ambient Intelligence Environments. Proceedings of the 5th International Symposium of Ubiquitous Computing and Ambient Intelligence - UCAmI 2011, Riviera Maya, Mexico, 5-9 December 2011.

[75] Release 2 Specification – Volume 5, Declarative Application, v2.0, Open IPTV Forum, Setember 2010, http://www.oipf.tv/Release_2.html (accessed on 18.03.2012).

[76] Digital video broadcasting (DVB); Portable Content Format Specification 1.0 XML Schema; TS 102 523 V1.1.1 Schema.

[77] Paulson, L. "Building rich web applications with AJAX". IEEE Computer, 38(10), 2005, 14-17.

[78] Garrett, J." AJAX: A new approach to web applications". Adaptive path, 2005.

[79] Smullen, C. W., Smullen, S. A. "An Experimental Study of AJAX Application Performance". Journal of Software, vol. 3, p 30, 2008.

[80] Jussi Teirikangas, 'HAVi: Home Audio Video Interoperability', Helsinki University of Technology, 2001. http://www.csun.edu/~andrzej/COMP529-S05/papers/jussi_teirikangas.pdf (accessed on 18.07.2012).

[81] A. Rodriguez, M. Moreno, E. Cespedes-Borras, J. Carrabina, "Platform-independent Web-Based iTV Services Through DVB-PCF Transcoding". Proceedings of the IEEE IC-BNMT 2009. 18-20 Oct. Beijing, China.

[82] MAG S.L., http://www.mag.es/.

[83] XleTView, http://sourceforge.net/projects/xletview/files/XleTView/xletview-0.3.6/ (accessed on 19.07.2012).

[84] Porteiro, M. The Use of Subtitles to Treat Speech-Language Disorders. MONTI 14. Approved for publication.

[85] Pereira, A. Criteria for elaborating subtitles for deaf and hard of hearing adults in Spain: Description of a case study. In Anna Matamala and Pilar Orero. Listening to

Subtitles. Subtitles for the Deaf and Hard of Hearing. (2010) pages 87-102. Bern: Peter Lang.

[86] Lorenzo, Lourdes. Criteria for elaborating subtitles for deaf and hard of hearing children in Spain: A guide of good practice. In Anna Matamala and Pilar Orero. Listening to Subtitles. Subtitles for the Deaf and Hard of Hearing. (2010) pages 139-148. Bern: Peter Lang.

[87] Blender Foundation. Elephants Dream, 2006. Open Movie available at www.elephantsdream.org (accessed on 18.03.2012).

[88] Bartoll, Eduard & Anjana Martínez Tejerina. The positioning of subtitles for deaf and hard of hearing. In Anna Matamala and Pilar Orero. Listening to Subtitles. Subtitles for the Deaf and Hard of Hearing. (2010) pages 69-87. Bern: Peter Lang.

[89] César, P., Bulterman, C. A., Jansen, A. J. Usages of Secondary Screen in an Interactive Television Environment: Control, Enrich, Share and Transfer television Content. In Proceedings of 6th European Conference EuroITV 2008, Springer, 168-177.

[90] Blender Foundation. "Sintel", 2010. Open Movie available at www.sintel.org (accessed on 18.03.2012).

[91] Likert, R. A technique for the measurement of attitudes. Archives of Psychology, Vol 22 No. 140, 1932, 55.

“Seven and a half million years later.... Fook and Lunkwill are long gone,
but their ancestors continue what they started.”

Douglas Adams (“The Hitchhiker's Guide to the Galaxy”, 1979)