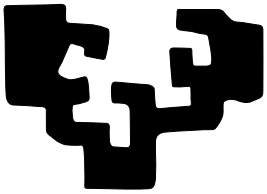
eman ta zabal zazu



Universidad del País Vasco
Euskal Herriko Unibertsitatea

DEPARTMENT OF COMMUNICATIONS ENGINEERING

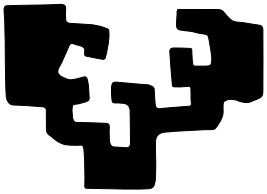
OPTIMISATION OF THE USER EXPERIENCE ACROSS MULTI-SCREEN MEDIA SERVICES

by:
Ana Domínguez Fanlo

Supervised by:
Dr. Alberto Lafuente Rojo
&
Dr. Mikel Zorrilla Berasategi

Donostia – San Sebastian, X Xth X, 2019

eman ta zabal zazu



Universidad del País Vasco
Euskal Herriko Unibertsitatea

DEPARTMENT OF COMMUNICATIONS ENGINEERING

OPTIMISATION OF THE USER EXPERIENCE ACROSS MULTI-SCREEN MEDIA SERVICES

by:
Ana Domínguez Fanlo

Supervised by:
Dr. Alberto Lafuente Rojo
&
Dr. Mikel Zorrilla Berasategi

Donostia – San Sebastian, X Xth X, 2019

Dedication Bla Bla

Bla bla

Abstract

The evolution of Internet-connected devices, as well as the changes in the way media is produced, distributed and consumed, have promoted the mobility and ubiquity of broadcast and media services. Consequently, the audiovisual sector has been transformed into a hybrid ecosystem where content is distributed across multiple devices. Moreover, a single user very often consumes content from more than one device at a time.

This novel context brings highly flexible experiences where the content not only has to be adapted to any target device but also requires an adaptation to multi-device environments, composed of a number of devices that are being used simultaneously by one or multiple end-users sharing an experience remotely. Accordingly, the user interface becomes a key factor to facilitate understanding the application and to provide an intuitive interaction method across multiple screens.

However, to the best of our knowledge, no existing adaptation models are available to dynamically and seamlessly adapt such a multitude of content to multi-device and multi-user contexts.

To address this gap, this research proposes a methodology which provides as an outcome an adaptation model for the user interface of multi-device media services within the audiovisual and broadcasting field, general enough to be easily adapted to many different use cases and scenarios and ready for any technological update.

The proposed methodology is the outcome of a extensive research that arose from a deployment of a hybrid Broadcast-Internet multi-device service with broadcasters, which has shown a set of hints to consider. This hints enabled to identify and characterise the user interface elements according to a set of design factors and formalise an adaptation model for hybrid

broadcast-Internet multi-device services, which has been implemented and validated in terms of quality, efficiency and universality.

Finally, this research also specifies how the mentioned methodology could be applied to other fields different from the broadcast analysing the challenges, the user needs and the specifications in each field.

Resumen

Bla bla bla Spanish

Acknowledgements

Bla bla bla

Gracias

Ana Domínguez Fanlo

December 2019

Contents

List of Figures

List of Tables

Part I

Introduction

CHAPTER

1

Scope of the research

1.1 Motivation

We are currently witnessing a transformation of the audio-visual sector towards a new hybrid TV ecosystem, where broadcast and broadband services coexist [Claudy12]. Although TV continues being the central device for audiovisual media consumption, the proliferation of devices such as mobiles, tablets and laptops together with the supply of Internet capabilities to TV sets, has entailed considerable changes in media production, distribution and consumption [Obrist et al.15][McGill et al.15]. This hybrid ecosystem opens a door to enhance traditional TV shows towards innovative TV experiences, moulding themselves to the new consumption habits and patterns of the users. TV viewers are no longer focused on a single content source or device. They look for additional content on a second screen and they socially interact with friends and the community, related or not to the mainstream media. Furthermore, users are getting used to a multi-device content access, demanding services at any time and from any device [Eastman and Ferguson12]. Consequently, the consumption of audiovisual and TV programmes has evolved, from viewing them in a stand-alone device to a multi-platform environment, being complemented by websites, applications, online-video streaming, chat rooms and so on.

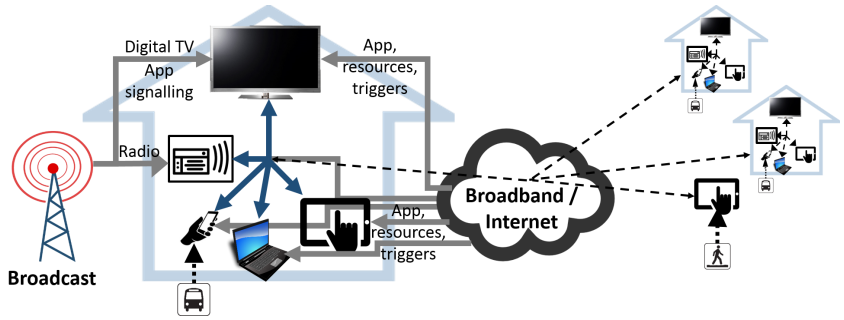


Figure 1.1: General overview of the Hybrid TV and multi-device experiences [Zorrilla et al.15a]

Hybrid TV and *multi-screening* concepts are linked to multi-platform environments [Baumann and Hasenpusch16] and they differentiate TV developments between two segments. Hybrid TV is seen as a concept addressing technologies such as IPTV [ipt] [Maisonneuve et al.09], OTT-TV[Baumann and Hasenpusch16] and HbbTV [hbbd], while multi-screening mainly refers to customer behaviour. The latter involves second-screening for related activities, dual screening for unrelated activities and social TV for social networks related activities. The ensemble of both concepts increases the choice offered and TV viewers are expected to switch from a passive form of TV consumption towards an interactive and online multi-screen experience. In addition, traditional broadcasters are trying to adapt their business to these new viewing habits.

Figure 1.1 is extracted from the novel hybrid TV ecosystem analysis in [Zorrilla16], which is an extension of the one depicted by [Claudy12]. It shows the general overview of the hybrid TV and multi-device experiences and represents the coexistence of the broadcast and broadband channels. On the one hand, traditional broadcast capable devices, such as TV sets and radios, are more and more supplied of both broadcast and broadband capabilities, whereas other devices, such as tablets, smartphones and laptops, can access to the same applications and contents over the Internet, providing the user with a consistent experience across multiple devices at the same time.

Several research projects have contributed to this field, such as the MediaScape [Med] European FP7-ICT-2013-10 project, ended in May 2016, which focused its work on helping broadcasters and application developers to easily provide socially engaging media-driven experiences across multiple screens for hybrid broadcast-Internet

content. MediaScape created open specifications and libraries on top of HTML5 and Web technologies, thus enabling the marriage of the TV, PC and Mobile worlds. The resulting libraries are based on the *Created Once and Published Everywhere* (COPE) concept [nem]. This means that the broadcaster will create a single application code, and the application itself will adapt to the multi-device environment of the user, providing association mechanisms, cross-device synchronisation and a dynamic self-regulated adaptation.

Regarding the advantages of multi-screen systems, they enable broadcasters and application developers to provide their users with more content related to the mainstream media. They also offer opportunities for controlling the experience as well as for allowing the user to include new ways of personalisation, social sharing and consumption of media content from various sources [Cesar et al.08]. However, besides obvious advantages delivered by multi-screen media, more screens demand higher cognitive load for viewers to understand what they watch and to correlate content from different video streams. The required visual attention also increases and needs to be distributed across multiple displays situated at various locations in a three dimensional space [Vatavu and Mancas15].

In this context, the user interface becomes a key factor in order to provide the users with a good experience across multiple devices at the same time, facilitating the understanding of the whole application and providing an intuitive interaction way. When broadcasters or application developers are facing a single-device user interface, they typically define a template to organise the items within a layout, usually describing different behaviours of the layout for disparate target devices. For instance, in Web applications developers can provide a CSS template with Media Queries that will adapt the user interface of the application to the final features of the device.

However, when developers are dealing with multi-device applications, this approach becomes unmanageable. Regarding multi-device adaptation, [Zorrilla et al.15a] provides a Web-based distributed adaptation architecture for media-driven multi-device applications, enabling broadcasters to provide the users with a single experience through multiple devices at the same time. In that solution, the system dynamically decides which elements of the application are be shown on each device, splitting the application through different devices simultaneously, in order to provide a consistent and coherent view to the user. This means that the list of elements available on each

device depends on the changing context of the users and will update dynamically when a new device is added or removed. In this scenario, it would be tedious, very expensive and unaffordable for developers to provide an explicit rules to organise the elements on a user interface for all the possible combinations, and even more for each target device. The proposed multi-device adaptation solution in [Zorrilla et al.15a] provides arbitrary divisions to arrange the elements in a responsive user interface following some given hints and rules, and creating a specific layout template according to the circumstances.

Nevertheless, it is still not easy for broadcasters and application developers to create a set of rules and hints to select the best user interface based on some contextual information, since they are not used to this contextual-based rules. There are aspects in interface design, such as the functionality and usability that are well known aspects in the field of Human Computer Interaction (HCI), where the developers have stronger criteria [McNamara and Kirakowski06]. For example, for a background device such as a TV, developers might want to allow overlaps between the different elements with Picture-in-Picture-like interfaces, while preventing elements to appear below the visible area of the screen to avoid scrolling. This pattern could be completely different for a touchable device, in the same way that it will be very dependant on the nature of the media application and its elements. Moreover, in addition to the traditional HCI parameters, there is a new wave in the field emphasising the importance of aesthetic aspects in interface design for the users' likability and system acceptability [Bauerly and Liu08] [Altaboli and Lin11].

In this context, the discussion of the user interface elements and design factors that are involved in a multi-device media application could become a valuable research. On the one hand, all the considered dimensions could contemplate the functionality, usability and aesthetic aspects addressed in the literature for single-device interface design. On the other hand, the factors should contemplate the new interface design aspects introduced by the multi-device dimension, not yet addressed in the literature, as one main contribution of this work. Furthermore, it would be desirable to provide a user interface model for media applications in multiple devices simultaneously, on top of the user interface elements and design factors previously defined. This model would aim to be as general and flexible as possible, for it to be parametrised by developers according to the needs of the specific application and over well-known HCI concepts.

From these HCI criteria parametrisation, the general model could conclude in an automatically created set of rules to be applied in a multi-screen system, and the rules would decide the distribution of the content across the connected devices and the arbitrary layout templates to be used according to the specific contextual information of the multi-device dimension.

1.2 Hypothesis

The interoperable and standard-based solution presented in [Zorrilla16] fosters the broadcast-Internet convergence and allows broadcasters and media application developers to easily create applications on top of the COPE concept. In this context, the working hypothesis is constructed as a statement of the following expectations:

1. It is possible to define a general methodology that will end with an adaptation model to seamlessly adapt the user interface of multi-device media services.

    (a) A large-scale pilot deployment of a hybrid broadcast-Internet multi-device service for a live TV programme on top of the Web-based distributed architecture presented in [Zorrilla et al.15a] could provide very valuable knowledge regarding the experience with the broadcaster, the application development and the end user usage of the service.

    (b) With the learnt lessons an overall methodology which faces the muti-device adaptation taking into account a wide range of design factors could be defined.

    (c) The methodology will end in a model able to both distribute the content across devices and generate a layout template for each connected device.

    (d) The model could provide a near real-time adaptation while guaranteeing a good quality.

    (e) The model may need to be iterative in order to take into account the context information mainly produced by the user but also related to environmental and system factors.

    (f) The model could be complemented with learning processes that allow to modify or refeed it with context information.

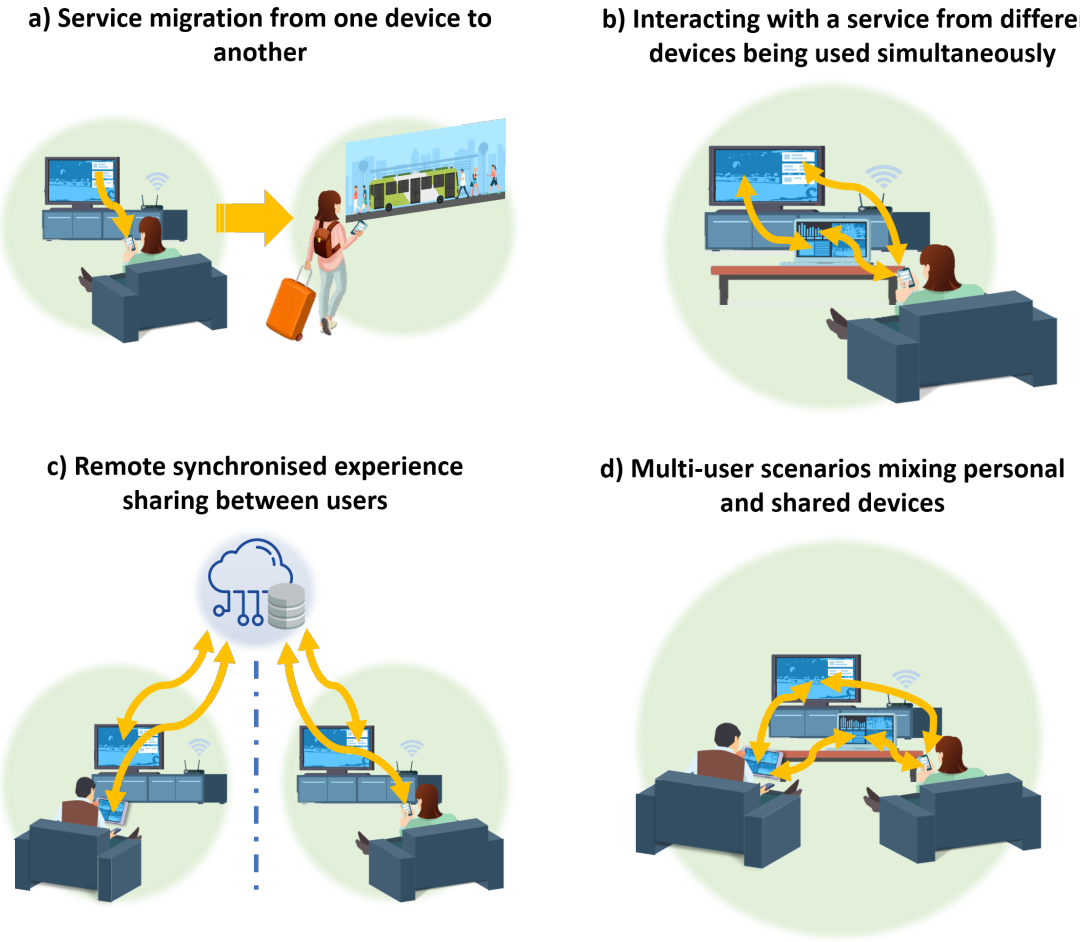


Figure 1.2: Different multi-device scenarios which could be found in the broadcast field

2. This methodology will be adaptable and extensible to other application fields, adjusting to the specific requirements of each case:

   (a) It could be general enough to be extended to any type of content or device and also ready for technological changes.

   (b) It could be used in different scenarios and use cases within the broadcast field such as the ones shown in Figure 1.2

   (c) It could be useful for other sectors where multiple devices are used at the same time, such as Industry 4.0 or crisis management environments where the role of videowalls could be relevant.

1.3 Objectives

The main objective of this work is to provide a methodology which provides as an outcome an adaptation model for the user interface of multi-device media services, general enough to be easily adapted to many different use cases and scenarios and ready for any technological update. In order to reach that goal it is necessary to overcome the following challenges:

- Objective 1: Generate a real pilot deployment of a hybrid broadcast-Internet multi-device service for a live TV programme using a rule-based adaptation process that allows to draw some conclusions in order to face the definition of an adaptation model for the user interface of multi-device media services.
- Objective 2: Define a methodology that goes an step further than explicit adaptation rules and allows to generate more automatic and simple adaptation models for applications based on the COPE (Create Once and Publish Everywhere)[nem] concept and the trend of componentizing the Web [Savage15]. In addition, the objective behind the methodology is to ease developers and broadcasters the creation of seamless and self-adaptive applications.
- Objective 3: Define a flexible and extensible adaptation model in such a way that it is still good not only for every use case, scenario and technological change but also for other sectors different from the broadcast. For that, it will be necessary to identify and characterise several user interface elements and design factors, but the methodology will conclude on an adequate model given a specific context.
- Objective 4: Provide example implementations in order to validate the methodology and the model in terms of quality, efficiency, adaptability, flexibility and extensibility.

1.4 Methodology

The research work performed in this Ph.D. has followed a methodology based on the two loop design cycle described in Figure 1.3. Based on the motivation for this research topic, a working **hypothesis** has been constructed as a statement of expectations. To

achieve the two main **contributions** of the research, a two stage procedure has been followed:

On the first stage, a real pilot deployment of a hybrid broadcast-Internet multi-device service has been designed, implemented and evaluated:

- **Design**: First of all, the design objectives have been clearly defined to be addressed during the architectural design of the different aspects of the solution.
- **Implementation**: A real pilot has been implemented following the architectural design and the guidelines of a broadcaster.
- **Evaluation**: The deployment has been evaluated providing the developed service to the audience and analysing the usage statistics of more than a thousand users.

Once the first stage has been completed the hypothesis has been adjusted taking into account the lessons learned from the previous deployment. Then, a second stage has been carried out in which a methodology has been defined to generate adaptation models for multi-device media services general enough to be ready for every use case as well as for every technological change. For that, this second stage has been divided into three steps:

- **Methodology**: A methodology has been defined with different steps, to gather information that allows to identify all the elements and design factors that play an important role in the adaptation process of a multi-device user interface.
- **Modelling**: Taking into account all the information gathered along the steps of the previous methodology, a model has been defined which includes the characterisation of the involved user interface elements, the adaptation process and an evaluation model.
- **Validation**: Implementation examples of the model have been developed in order to validate its quality, efficiency, adaptability, flexibility and extensibility.

The two stage procedure has helped to refine the design and implementation of the target solution, leading to the final contributions of the research.

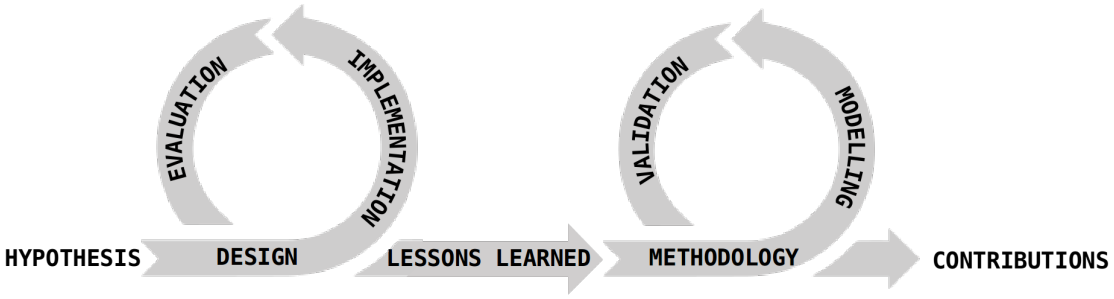


Figure 1.3: Design cycle followed as the research methodology

1.5 Contributions

The main contribution of this Ph.D. research has been the definition of an adaptation model for the user interface of multi-device media services, general enough to be easily adapted to many different use cases and scenarios and ready for any technological update. As a result of the objectives mentioned in Section 1.3 the main contribution can be translated into three specific outcomes:

1. Design and implementation of a large-scale deployment of a hybrid broadcast-internet multi-device service for a live TV programme which shows the lessons learned about the current hybrid broadcast-Internet ecosystem and multi-device live services.

2. Design and implementation of a methodology that provides as an outcome an adaptation model for the user interface of multi-device media services, which has been formally described and validated in terms of quality, efficiency and universality. This includes the identification and characterisation of all the user interface elements involved in the adaptation process.

3. Identification of different fields in which the proposed methodology could be applicable.

Figure 1.4 illustrates the three contributions of the research to address the definition of an adaptation model for the user interface of multi-device media services. Furthermore, it shows the readiness of the model to be adapted to technological updates as well as to be integrated with continuous adaptive learning processes.

The contributions delivered the following publications:

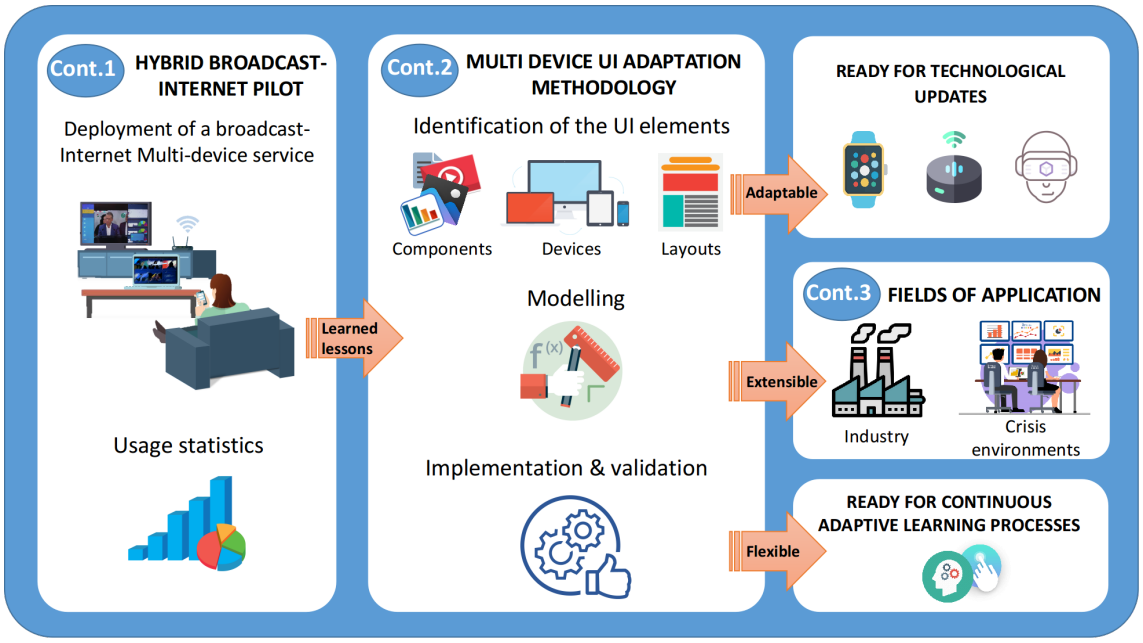


Figure 1.4: Contributions of the research

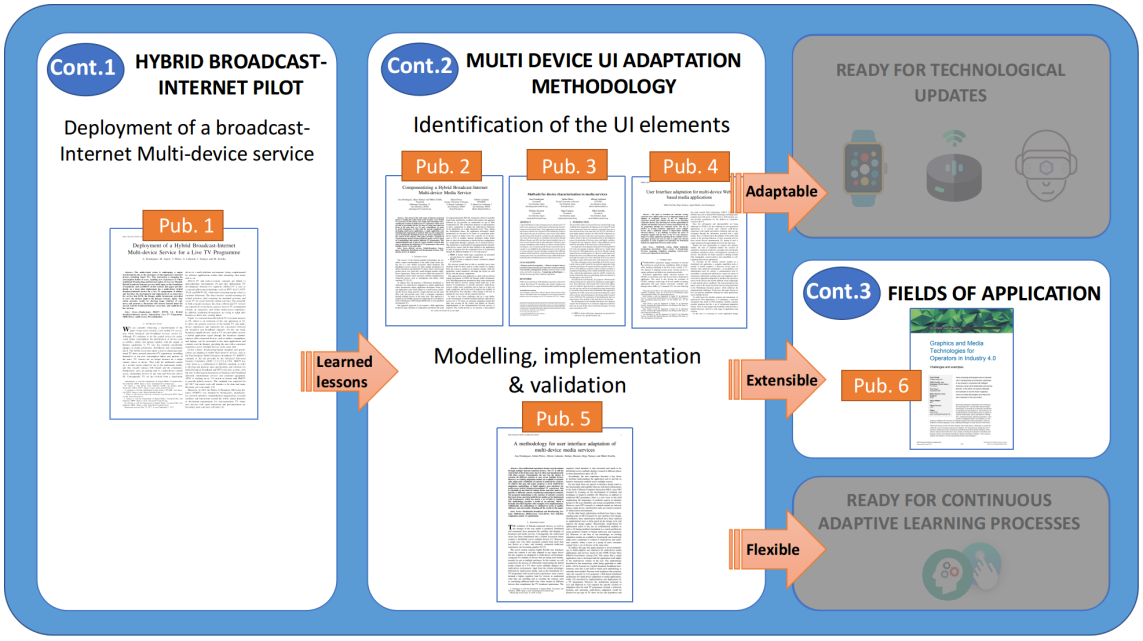


Figure 1.5: Publications of the research

- Publications related to Contribution 1:
  - Pub. 1: *Dominguez, A., Agirre, M., Flörez, J., Lafuente, A., Tamayo, I., & Zorrilla, M. (2017). Deployment of a hybrid broadcast-Internet multi-device service for a live TV programme. IEEE Transactions on Broadcasting, 64(1), 153-163.*
- Publications related to Contribution 2:
  - Pub. 2: *Dominguez, A., Tamayo, I., Zorrilla, M., Flórez, J., & Lafuente, A. (2018, June). Componentizing a Hybrid Broadcast-Internet Multi-device Media Service. In 2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB) (pp. 1-6). IEEE.*
  - Pub. 3: *Dominguez, A., Florez, J., Lafuente, A., Masneri, S., Tamayo, I., & Zorrilla, M. (2019, June). Methods for device characterisation in media services. In Proceedings of the 2019 ACM International Conference on Interactive Experiences for TV and Online Video (pp. 118-128). ACM.*
  - Pub. 4: *Zorrilla, M., Tamayo, I., Martin, A., & Dominguez, A. (2015, June). User interface adaptation for multi-device Web-based media applications. In 2015 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (pp. 1-7). IEEE.*
  - Pub. 5: *Dominguez, A., Florez, J., Lafuente, A., Masneri, S., Tamayo, I., & Zorrilla, M. (Submitted in 2019, November). A methodology for user interface adaptation of multi-device media services. IEEE Transactions on Broadcasting*
- Publications related to Contribution 3:
  - Pub. 6: *Posada, J., Zorrilla, M., Dominguez, A., Simoes, B., Eisert, P., Stricker, D., ... & Guevara, M. (2018). Graphics and media technologies for operators in industry 4.0. IEEE computer graphics and applications, 38(5), 119-132.*

Figure 1.5 shows the relation between the objectives, the contributions and publications.

1.6 Document structure

This thesis has been structured as follows. Part I presents an introduction to the research scope, focusing on the motivation for the research, the main objectives, the hypothesis, the methodology and the main contributions of the Ph.D. work.

Part II overviews literature related to the hybrid ecosystem, the involved specifications and technologies in ubiquitous media applications and the adaptation and optimisation of the user interface of web applications.

In Part III the research results are described in three main chapters:

- Chapter 3 describes the deployment of a large-scale pilot of a hybrid broadcast-Internet multi-device service for a live TV programme and the lessons learned (Contribution 1).
- Chapter 4 describes a methodology that enables to generate an adaptation model for multi-device media services together with its implementation and validation (Contribution 2).
- Chapter 5 describes how the proposed methodology could be applicable in other fields beyond broadcast (Contribution 3).

In Part IV the main conclusions of the research can be found, including a discussion that enables future work.

Finally, Part V provides other publications of the author and his Curriculum Vitae as an appendix, while Part VI contains the bibliography.

Part II

State of the Art

CHAPTER

2

Background and Related Work

2.1 Overview

As explained in Section 1.1, there is a new trend towards hybrid broadcast-Internet multi-device services which are not still easy to develop for broadcasters and application developers due to a lack of rules or hits to select the best user interface based on some contextual factors. To provide the basis of the research that overcomes that challenge, this chapter is divided in three sections. Section 2.2 and Section 2.3 gather the background of the hybrid TV ecosystem and the technologies for interoperable multi-device services respectively, while Section 2.4 describes the related work regarding the optimisation of the user interface in multi-device media services. This last section goes over previous work dedicated to multi-device or multi-content optimization in similar environments like the target one to be able to identify the unsolved problems.

2.2 Analysis of the hybrid TV ecosystem

This subsection presents an analysis of the environment and technologies involved focusing on Hybrid broadcast-Internet multi-device services. The following is a list of statistics and facts about the hybrid ecosystem.

1. Future of Broadcast Television Initiative (FOBTV) was founded in 2012 by broadcasters, manufacturers, network operators, standardization organizations, research institutes and universities around the world, aimed primarily at developing requirements for next-generation TV where new services with smart interaction and personalization are becoming more and more relevant [Zhang et al.14].

2. Broadcast capable devices are still the most used for the consumption of media services and thus, the most appropriate to reach a higher audience. According to the Nielsen report of Q3 2018 [nie], TV and radio account for 57% of the average audience among adults aged 18+.

3. Smart TV shipments [staf] have constantly increased over the years. From the 109.3 Million Smart TVs shipped in 2014, this increased to 240.2 Million in 2018 and reached 265.3 Million in 2019.

4. According to the TV and Media 2016 study by Ericsson ConsumerLab [eria], in 2016, 31% of users browsed Internet content related to what they were watching, 19% of users participated in online discussions about the content being watched, and lastly, 20% of users used the second screen to watch two or more programs at the same time. Those percentages were 23%, 17% and 15% respectively in 2014. Moreover, according to the same study in 2017 [erib] in 2020, it is expected that only 1 in 10 consumers to be stuck watching TV only on a traditional screen, a 50 percent decrease compared to 2010. Therefore consumers are evolving into higher-usage and higher-spending multi-screen viewers.

From the above facts we can conclude that although there is not too much experience around hybrid multi-device experiences, the market shows a clear trend towards them.

For that, there are broadcast-related standards and specifications to provide interactive experiences within Digital TV, providing solutions to add second screen devices to the experience in cooperation with the TV.

Nested Context Language (NCL) [Soares et al.09] is the declarative language of the Brazilian Terrestrial Digital TV System supported by its middleware called Ginga. NCL is standardised in the ITU-T Recommendation H.761 for IPTV services. As a glue language, NCL relates media objects in time and space without restricting or imposing

any media content type, including media objects with imperative and declarative code written using other languages [Soares et al.10]. Moreover, NCL provides declarative support for presenting distributed DTV applications on multiple devices. Presentation of media objects in NCL applications can be associated to devices using an abstraction called device classes and the parent device, for instance the TV, controls the secondary devices. [Soares et al.09] presents a solution for the exhibition of multiple devices in cooperation in DTV Systems with NCL.

HbbTV is a pan-European specification published by ETSI to allow broadcasters signalling, delivery through the carousel, rendering and controlling the application with a HTML based browser in the TV or set-top box, providing a hybrid broadcast-Internet experience [Merkel11]. In February 2015, the 2.0 specification of HbbTV was published with new functionalities. In contrast with versions 1.0/1.5, HbbTV 2.0 uses HTML5 and a set of related Web technologies including many modules from CSS level 3, DOM level 3, Web Sockets, etc. Regarding the addition of companion screens to the TV, an HbbTV 2.0 application on a TV can launch an application on a suitably enabled mobile device. Moreover, HbbTV 2.0 enables an application on mobile to remotely launch an HbbTV application on a TV, based on DIAL[1].

HbbTV has come about as a combination of different standards in order to develop and promote open specifications and solutions for hybrid broadcast-broadband and IPTV television systems, with the aim of allowing harmonisation of broadcast and broadband delivered entertainment services and consumer equipment.

Figure 2.1 shows the relevant aspects within each block of HbbTV specifications 1.0 [hbba], 1.5 [hbbb] and 2.0 [hbbc] for the development of HbbTV multi-device services. The Figure extends the HbbTV overview in [hbbd], focusing on the multi-screen aspects of the specification and detailing the different features introduced on each one of the HbbTV versions. The Open IPTV Forum (OIPF) [oip] provides the JavaScript API for TV environments through the Declarative Application Environment (DAE) [DAE], basing its development on World Wide Web Consortium (W3C) [w3c] specifications, as well as also defining the media formats. Furthermore, latest HbbTV 2.0 specification includes Companion Screen features to discover, launch and communicate second screen

[1]DIAL Discovery And Launch protocol specification. Version 1.7.2 (2015) `http://www.dial-multiscreen.org/dial-protocol-specification`

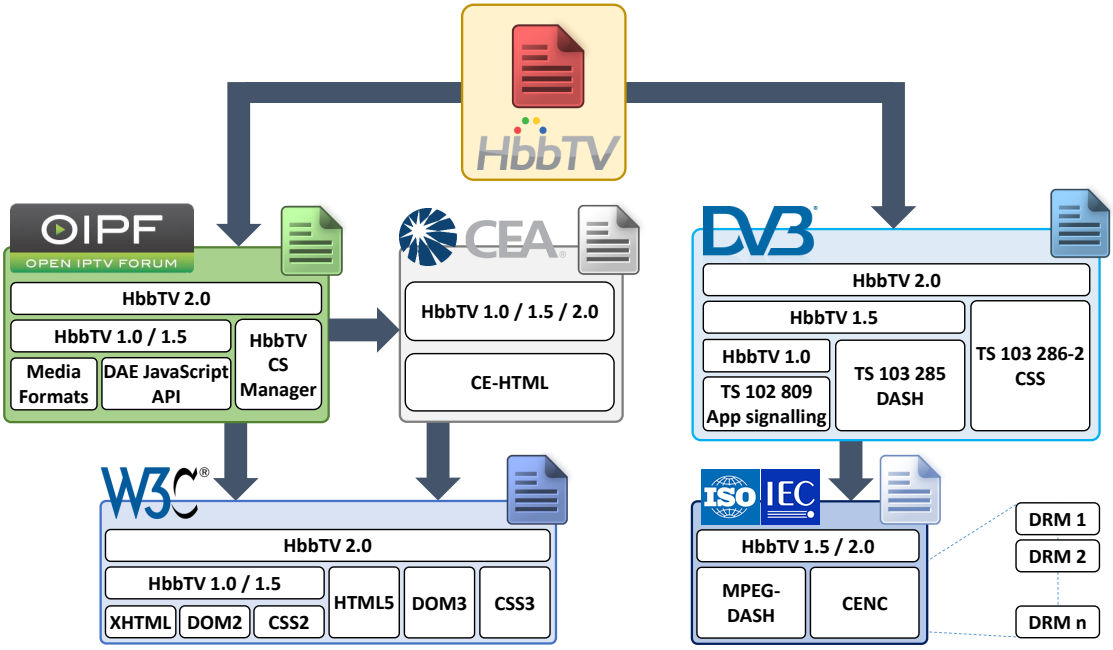


Figure 2.1: HbbTV overview focused on multi-device services

devices with the TV in local home networks. More specifically, HbbTVCSManager embedded object is the responsible for discovering the Companion Screens with a running Launcher and sending application launch/install information to them. In the same way, it responds to discovery requests from Companion Screens and launches HbbTV applications in the hybrid terminals. Additionally, HbbTVCSManager is responsible for providing service endpoints for application to application communication feature. The Consumer Electronics Association (CEA) [cea] defines the application languages (HTML [htmc], CSS [css] and JavaScript [js] including AJAX [aja]) established by W3C as well as the DOM [dom] event handling. A significant change is appreciated in W3C's block from HbbTV 1.5 to HbbTV 2.0, where supported features evolve from XHTML, DOM2 and CSS2 to DOM3, CSS3 and HTML5. These features are supposed to be an enormous benefit for HbbTV multi-device services but commercial devices with full HbbTV 2.0 implementations are not available yet. DVB [DVB] is the one that allows application signalling and its transport via broadcast or HTTP. Moreover, the latest HbbTV 2.0 specification enables the identification and synchronisation of timed content and event triggering between hybrid terminals and personal devices through Companion Screens

and Streams (CSS) interface. Finally, the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) [iso] [iec] provides compatibility with leading Digital Rights Management (DRM) and support for the MPEG-DASH [Sodagar11] adaptive streaming.

Furthermore, Operator Applications (OpApps) [opa] specification is an extension to the core HbbTV specification to support operator application. It describes how the HbbTV browser can run both HbbTV broadcaster applications and operator applications at the same time.

Finally, Advanced Television Systems Committee (ATSC) [ATSa] [Chernock et al.16] is the group in charge of the development of the Digital TV in USA, Canada, South Korea and Mexico. ATSC has launched its 3.0 version [ATSb] which has been designed to offer support for newer technologies, including HEVC for video channels of up to 2160p 4K resolution at 120 frames per second, wide color gamut, high dynamic range, Dolby AC-4 and MPEG-H 3D Audio, datacasting capabilities, and more robust mobile television support. The new standard is built to have interaction with second devices, providing opportunity to broadcasters to engage on consumers' devices as well as the TV. The first major deployments of ATSC 3.0 occurred in South Korea, with the country's major television networks launching terrestrial ATSC 3.0 services in May 2017 in preparation for the 2018 Winter Olympics.

These broadcast-related standards, even though they allow mechanisms to launch applications on a companion screen or provide cooperative multi-device application, they do not avoid specific developments for each compatible TV. Moreover, developers have to define the behaviour of the application on each device at the development stage, specifying how to visualise the application on the TV and on the second screen. Developers typically need also to handle the communication between the devices, usually limited to a local network communication, for the contextual synchronisation of the experience.

2.3 Technologies for interoperable multi-device services

This subsection presents the available technologies to create interoperable multi-device services. The development of this kind of services is directly related with three main challenges as stated in [Zorrilla16].

- Multi-connection: The application itself has to discover the services and other available devices, associating multiple devices to a single user and multiple users to a shared experience, as well as providing mechanisms for a cross platform authentication

- Cross device synchronisation: All the content has to be perfectly synchronised across different devices, taking into account all the media sources, as well as the associated data

- Multi-device adaptation: The application itself has to be adapted to the end-user's dynamic multi-device context, providing a single experience through multiple devices at the same time

If these three challenges are overcome, a versatile and interoperable solution is obtained, independent from the specific target device or platform. The following subsections present the technologies available to face each of the previous challenges.

2.3.1 Discovery and association

When connectivity has to be provided between different devices in an entertainment environment, it is essential to avoid the human intervention as much as possible and make easier the discovery of new services and the association of devices. Nowadays, these are the existing technologies for discovery and association of devices.

- Bluetooth: a wireless technology for data exchange in short distances. Bluetooth protocols simplify the discovery and configuration of the services between devices. Bloetooth devices can announce all the services they offer with a high level of security.

- NFC: a set of standards for smart phones and similar devices to establish radio communications in a proximity context. Typical applications include wireless transactions, data exchange and simplified configuration of more complex communications such as WiFi.

- QR code: Quick Response (QR) is the registered brand of a barcode type designed to store data such as an URL in an efficient way. Its success is due to the easiness to read it and a higher storage than the traditional barcodes. It is commonly used in second screen applications.

- Acoustic patterns: acoustic patterns can be emitted by a device to be captured by the surrounding devices. An URL could be mapped to different inaudible frequencies, creating an acoustic code in order to associate two devices.

- Named Web Sockets: Named WebSockets[2] is an especification that defines the mechanisms to discover and connect pairs of WebSockets that share the same name of service in a local network. The evolution of Named WebSockets is known as Network Web Sockets[3].

2.3.2 Synchronisation

In order to achieve the synchronisation among devices, W3C through the *Multi-device Timing Community group*[4] proposes the standardisation of temporal objects called *Timing Object*, in order to build a model that allows an accurate synchronisation and time controlling Web-based multi-device applications. It consists on instantiating a timing object in each device that is connected to a online shared temporal source. In order to obtain a perfect synchronisation, the Timing Object supports any velocity and acceleration and any jump in the timeline.

Built on top of the previous work *timingsrc* can be found which is a programming model based on Timing Objects in order to achieve an accurate synchronisation in multi-device Web applications. Within this model three different services can be found:

- MediaSync[5]: it allows the synchronisation of HTML5 videos through a JavaScript library and it is based on a study of the behaviour of media elements in different browsers.

[2] http://namedwebsockets.github.io/spec/
[3] https://github.com/namedwebsockets/networkwebsockets
[4] https://www.w3.org/community/webtiming/
[5] http://webtiming.github.io/timingsrc/doc/online_mediasync.html

- Sequencer[6]: a generic tool to process temporal data. It provides entry and exit events for data used with Timing Objects.

- Shared Motion[7]: it provides an online service to distribute the time to different Web agents.

2.3.3 Adaptation

There are available solutions and in-progress language recommendations in the field of adaptive interface of Web applications and its responsive design addressing the adaptation for a single device. However, these approaches do not face the adaptation of an application in the multi-device domain.

Responsive Web design is a key factor in Web development, targeting devices with different features such as connected TVs, laptops, tablets and mobiles. Their heterogeneous displays and characteristics requires the responsive Web design to provide an adequate viewing experience automatically adapted for each device.

HTML features a number of fall-back mechanisms to adjust the content to the device capabilities. Two different approaches to design responsive Web applications can be deployed. On the one hand, a server-side detection based upon information embedded in HTTP requests. The result can be used to return a suitable media content, e.g. a video, where the server selects an optimal bitrate and resolution mode for the requested content based on the device in question.

The second approach for responsive Web applications, covered by the HTML mechanisms, is based on client-side detection and adaptation, enabling a wider set of possibilities. HTML5 includes attributes (*srcset*) that allow application developers to provide multiple resources in varying resolutions for a single image [W3C Editor's Draft14]. Audio and video tags (<*audio*>, <*video*>) permit the definition of different source types and let the browser choose which format to use on the current device (audio/ogg, audio/mp3, etc.). Another powerful client-side mechanism is based on *media queries (@media)* [W3C Recommendation12]. These simple filters can be applied to CSS styles in order to change the properties based on characteristics of the end-device. Even for

[6] http://webtiming.github.io/timingsrc/doc/online_sequencer.html
[7] http://webtiming.github.io/timingsrc/doc/shared_motion.html

showing suitable media content at the client, the trend with streaming protocols like MPEG-DASH moves the adaptation from the server to the client.

Nevertheless, the most widespread way to achieve adaptive interfaces lays on designing the CSS style sheets to be applied to different displays. Therefore, the alternatives often match the application design with the device context, adapting the layout and the user interface. W3C proposes some basic recommendations [W3C Recommendation10] to improve the user experience of the Web when accessed from mobile devices.

There are several frameworks that support the development of responsive applications. Gridster.js[8] boosts the dynamic grid layouts for drag-and-drop actions based on jQuery. CSS3 Flexbox [W3C Working Draf 14] empowers the layout definition in the CSS by describing a box model optimised for the user interface design, where the children of a flex container can be laid out in any direction –both horizontal and vertical – in a flexible way. CSS3 Regions [W3C Working Draft14b] allows elements moving through different regions where CSS Regions library[9] expands the browser support with broader feature coverage. Grid-layout [W3C Working Draft14a] defines a two-dimensional grid-based layout system, optimised for user interface design. The XML XUL[10] language has been created by Mozilla to design user interfaces. Kontx[11] is the proposal of Yahoo to develop TV apps. EnyoJS[12] is a framework to develop HTML5 apps focused on layouts and panels design. Bootstrap[13] is a framework for developing responsive mobile-first projects on the Web that includes a grid system to scale the layout as the device or viewport size changes.

Web Components [web] is the umbrella term for a set of new W3C specifications that give developers the primitives needed to build reusable and interoperable elements. The individual specifications that make up Web Components map almost directly to the specific features needed to create a truly interoperable element.

- The Custom Elements specification [cus] describes how an element might be given a name, define an API surface area, and respond to different events in its

[8] Gridster.js website. `http://gridster.net/`
[9] CSS Regions library, April 2014. `https://github.com/FremyCompany/css-regions-polyfill`
[10] Mozilla XUL. `https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL`
[11] Kontx. `https://developer.yahoo.com/connectedtv/kontxapiref/`
[12] EnyoJS. `http://enyojs.com/`
[13] Bootstrap. `http://getbootstrap.com/`

lifecycle. That way, personalised elements can be easily reused in different web applications.

- The HTML5 <template> element [htmb] provides the set of features needed to declare inert DOM subtrees ready to be instantiated in the final DOM.

- The Shadow DOM spec [sha] provides the mechanism for encapsulation. It introduces the notion of a "Shadow Root" which is a separate, scoped tree that mantains functional boundaries between DOM trees and avoiding, for instance, accidental interference by CSS selectors or DOM manipulation methods.

- The HTML Imports specification [htma] provides this mechanism to load HTML-based dependencies in HTML.

With these four crucial new features Web developers are provided with the platform-level primitives needed to create truly reusable custom elements, with all the power of native HTML elements.

The JavaScript implementations of the previous specifications support Custom Elements, HTML Imports, and Shadow DOM across the last two versions of major browsers starting with IE10, Safari 7, and the evergreen browsers Chrome and Firefox. Moreover, Web Components-based libraries such as X-Tag[14], Polymer [15], and Bosonic[16] rely on some of the polyfills[17] for broad browser support, and include optimizations around the heavier parts of the polyfills to achieve production-ready performance.

Apart from Web Components, other frameworks such as Angular or React also allow component-based Web development. These frameworks are not based on any standard and therefore, they present less dependencies.

The combination of components and *media queries* boost the development of responsive multi-device Web applications and they are core technologies, among others, of the next generation of Google apps [Techrunch13].

All these languages, recommendations and frameworks for adaptive interfaces and responsive design are very useful for smooth local adaptation but need to be extended

[14] http://x-tags.org/.
[15] https://www.polymer-project.org/1.0/.
[16] http://bosonic.github.io/.
[17] A polyfill is a downloadable code which provides facilities that are not built natively in a Web browser

towards the multi-device dimension. They are all designed to adapt an application to a device aiming the adaptation depending on the device features. However, they do not consider that an application can be running in one or more devices simultaneously, including only part of the application.

2.4 Optimisation of user interface of multi-device media services

This subsection describes the attempts to optimise the user interface of multi-device media services found in the literature. Optimisation methods have been a long-standing topic in HCI research for user interface (UI) design. Nevertheless, these optimisation methods have been explored as supplemental ways to help speed up the design cycle and improve the design quality. Theoretically, model-based UI optimisation refers to the use of combinatorial methods to solve a UI design problem formulated as a search problem by using predictive models of human behaviour and experience [Oulasvirta17]. However, to the best of our knowledge, no existing adaptation models are available to dynamically and seamlessly adapt such a multitude of content to multi-device and multi-user contexts, where a user or a group of users consumes content from a set of devices at the same time.

With the advent of the aforementioned new and interactive media experiences, Layout Appropriateness (LA) [Sears93] has become a key term for the evaluation of user interfaces. The LA concept assigns a cost to every layout according to specific metrics such as the time required to accomplish a task or the users' preferences. Therefore, the appropriateness of a given layout is computed by weighting the cost of each sequence of actions by how frequently the sequence is performed. In this context, Sketchplore [Todi et al.16] is defined as the first user interface optimiser that uses different metrics to optimise the spatial and colour aspects of interactive layouts.

However, in a multi-device system where a user can be consuming content from multiple screens at the same time, the typical template-based approach cannot work, as the set of elements to be shown on each device can change over time, depending on the number of components to visualize and the number of devices connected to the system. In a multi-screen environment, such as that addressed in [Zorrilla et al.15a],

the content is divided into logic elements with the Web components specification [web] following an object-based broadcasting approach [Armstrong et al.14]. In this context, the system will dynamically decide which elements of the application will be shown on each device, simultaneously splitting the application through different devices, to provide a consistent and coherent view to the user, which means that the list of elements available on each device will depend on the changeable context of the users and will dynamically update when a new device is added or removed. In this scenario, it is tedious, very expensive and unaffordable for developers to provide an explicit template to organise the elements on the user interface of each target device, considering all the possible combinations. Thus, the solution described in [Zorrilla et al.15a] provides generic and arbitrary divisions to arrange the elements in a responsive user interface following some hints given in the application code and creates a specific layout template depending on the context of the user.

Regarding cross-device adaptation, [Brudy et al.19] aimed to create a unified terminology to facilitate cross-device research; [Dong et al.16] identified technological and business factors that represent a barrier to create useful, usable and delightful multi-device experiences; [Grubert et al.16] presented the relevant challenges for the design, development and use of mobile multi-device environments; [Santosa and Wigdor13] provided a field study of multi-device workflows in distributed workspaces; [Wäljas et al.10] revealed the key elements of the user experience associated with cross-platform interactions and proposes an initial conceptual framework that can be used to guide the design of cross-platform web-services; [bbc] explored how second screens might enhance live TV shows with additional web content and external links; and [Jokela et al.15] provided a user study on people's current practices in combining multiple devices in their everyday lives.

Apart from the previous studies and the identification of challenges, the XDBrowser [Nebeling17] [Nebeling and Dey16] project investigates how web browsers can be improved to better support parallel multi-device usage and provides a proof-of-concept implementation that segments web pages and distributes the parts across devices. Webstrates [Klokmose et al.15] synchronises elements across devices operating on the level of the Document Object Model (DOM) while maintaining exact copies on different devices. [Sarkis et al.18] allowed for the re-use of existing single-screen applications to automatically create synchronous multi-screen applications that analyse the code of the

application and classifies different elements, such as HTML tags or event listeners. Other frameworks work with graphical aspects of user interfaces, for instance, when spanning one image across multiple screens [Rädle et al.14] [Schreiner et al.15]. Other prior works on cross-device interfaces, such as [Frosini and Paternò14] or [Yang and Wigdor14], have proposed methods for synchronising elements across devices. All of these frameworks allow for the development of new multi-screen applications but they all require developers in the development stage to explicitly define how to distribute interface components across displays.

There are also rule-based approaches, such as [Husmann et al.17] or [Nebeling17], which provide insights into cross-device interaction patterns. However, they are focused on very specific functionalities and do not scale to many devices or multi-user scenarios. Vistribute [Horak et al.19] proposes a framework that identifies important properties and relations for distributed visualisation interfaces. Vistribute also provides six heuristics that can guide in the automatic distribution of visualisations in changing device setups together with their web-based implementation. However, those heuristics are focused on data analysis visualisations and therefore they are again not general enough to extend them for the adaptation of other contents in different fields.

AdaM [Park et al.18] proposed an optimisation-based approach that automatically distributes elements to available devices by solving the many-to-many assignment problem. AdaM distributes the UI elements based on an objective that maximises the usefulness of an element on a device while simultaneously maximising the completeness of the UI form from the user's perspective. However, the problem of a layout of elements on a device is not addressed in this work, as it is assumed to be performed via responsive design practices common in web design.

In the mentioned literature there is a lack of an adaptation model that takes into account the following aspects:

- Ability to distribute and adapt such a dynamic amount of content to any device.
- Good performance at run-time and in real-time without having a negative effect in the user experience.
- Usefulness for any multi-device and multi-user context.

- Provision of a systematic basis for the adaptation that frees the developer from the duty of defining a wide range of explicit rules.

However, the definition of the model itself is not straightforward, which is why this work proposes a methodology to adapt the user interface of multi-device media services.

Part III

Research Results

CHAPTER

3

Hybrid broadcast-Internet pilot

3.1 Overview

The audio-visual sector is undergoing a major transformation due to the emergence of heterogeneous connected devices, including Smart TVs. This reinvention is changing the consumption habits of users, enabling new experiences in which traditional broadcasting and Internet media services are blended. Hybrid broadcast-Internet services build upon on the foundation of standards, such as HbbTV. In this context, this chapter provides details on a large pilot deployment for a multi-device hybrid broadcast-Internet service for a live TV programme. It defines and describes the development and deployment of an innovative live service that EiTB, the Basque public broadcaster, provided to cover the election night in the Basque Country, Spain. This work provides results by showing usage statistics of end-users, and presents a discussion with lessons learned about the current hybrid broadcast-Internet ecosystem and multi-device live services.

3.2 Motivation

The analysis in Section 2.2 shows that there are standards, technologies and solutions that enable the delivery and consumption of live multi-device media services. However,

there is not yet a clear methodology, and there are still difficulties in providing a completely self-regulated hybrid service that will work under any circumstances following the COPE paradigm.

In order to find out the causes of the mentioned difficulties, it was decided to build an innovative hybrid broadcast-Internet multi-device service and test it through a large-scale pilot by means of the broadcaster. Prior to the deployment of the large-scale pilot, some questions were identified and answered which would be later checked while working in the real environment:

1. **Question 1: Are broadcast-driven hybrid multi-device services accompanied by a live TV programme interesting for users?** Users might like this kind of applications, since their consumption behaviour is moving towards using more than one device at the same time to learn more about the content they are watching on TV. However, since there are no broadcast-driven hybrid services with a live TV programme, users do not have a reference with which to align their expectations. HbbTV-based applications are typically used for catch-up TV, instead of enhancing broadcast programmes with additional content. Furthermore, at this time, the second screen activity is always performed through an independent application and users are not used to associating different devices in the same session while being in front of the TV. Therefore, in order to make a multi-device service which is related to a live programme successful, users will need a completely usable user interface together with the guidelines on how to use the service.

2. **Question 2: Are broadcasters ready to cope with hybrid multi-device experiences?** With broadcasting having always been the unrivalled giant, the environment has changed and now a huge amount of sources are available. Therefore, in order to stay in the game, broadcast technology needs an upgrade, and HbbTV represents a great opportunity. Broadcasters know this and that is why more and more TV channels have their own catch-up HbbTV services.

   The problem here is that going ahead with broadcast-driven multi-device experiences for live programmes means a major change in their workflow and business model, and change has never come easily for them [Claudy12]. Broadcasting has always slowly adopted new technologies, and this is due to the highly regulated environment in which they operate and the time required to achieve a

consensus. This regulation also generates a sense of fear of changing to such interactive multi-device experiences, as they would lose the complete control of the environment they have enjoyed until now.

In this context, broadcasters and application developers see the potential of broadcast-driven multi-device services but may feel it is an uncontrolled, unstable, immature and complex technology ecosystem to provide broadcast-driven multi-device services together with a live programme. This holds back the creation of these kinds of services, and that is why they currently limit their HbbTV services to catch-up TV. Probably, they would welcome simplicity in many dimensions if they had to assume these kind of deployments.

3. **Question 3: Are HbbTV implementations mature enough for broadcast-driven multi-device services?** HbbTV standard has significantly evolved since its creation in 2009. It has become the interactive TV standard in many European countries, and it has generated movement across the entire globe. HbbTV 1.5 version, published in 2012, brought important improvements over the 1.1 version, which lead to the implementation of a big amount applications, most of the times catch-up TV services. The standard, and its deployment in the market, is mature enough when talking about common single device applications.

   HbbTV 2.0 and 2.0.1 versions, published in 2015 and 2016 respectively, brought a powerful toolset including advanced Web technologies towards the interoperability with HTML5 and Companion Screen features, that could completely change the interactive TV towards multi-device services. However, HbbTV 2.0 and 2.0.1 versions are not commercially available at the moment in most of the countries. Therefore, there is still a lack of reference implementation of the standard when talking about multi-device services, leading to disparate HbbTV implementations in TV sets from different manufacturers, even if they have the same HbbTV version. There is also a lack of a certification ecosystem for HbbTV applications, to fully guarantee their compliance.

4. **Question 4: Are HTML5 and HbbTV convergent enough to solve interoperability problems?** HbbTV has appeared as a standard in order to facilitate the

convergence between the Web and TV. However, TV set manufacturers still include heterogeneous Web browsers in their devices, adapting them to HbbTV. Accordingly, they typically include more features defined by W3C than the minimum included in the standard. As illustrated in Figure 2.1, HbbTV 2.0 includes new developments in this HTML5-HbbTV convergence, adding HTML5 features, DOM3 and CSS3. However, it is quite common to find HbbTV 1.0 and 1.5 devices that provide some HTML5 features, even if it is not required by the standard. Furthermore, it is difficult for application developers to find the minimum HTML5 features to provide interoperability with HbbTV and follow the COPE paradigm of one application running in all devices (TV sets but also in laptops and mobiles). Moreover, the mentioned compatibility problems may arise the necessity of making the development as automatic as possible to prevent broadcasters from defining details of implementation and adaptation.

5. **Question 5: Are hardware and processing capabilities of the current TV sets enough to run broadcast-driven hybrid services smoothly?** TV manufacturers might be limiting their TV sets' processing capabilities just to the current needs. The lack of advanced and demanding HbbTV services means that there is no need for excessive processing capabilities. Consequently, this may be leaving the TV set one step behind the rest of devices towards COPE based applications. In this context, efficient implementations may help to bridge the gap between TVs and the rest of devices, at least until this kind of services are more known and attractive.

3.3 The innovative hybrid broadcast-Internet multi-device service

Election day is a very important event in which more and more information is exchanged. Data, statistics, news, exit polls or social network activity generate a huge amount of content from many different sources that are shared and consumed through different services, applications or devices.

In this context, EiTB, the Basque Public broadcaster, is the main information source for the election coverage in Basque Country (Spain), offering an up to the minute live TV programme on the night providing the exit polls' results and the official results together

with expert debates. Furthermore, they had a Website showing all the data. However, being aware of the fact that elections are a social issue involving millions of people, the broadcaster wanted to go a step further and provide an innovative and interactive service together with the live programme, including additional information.

During the MediaScape research project, a scenario to enhance a Live TV programme with additional content has been addressed [D21], enabling interactivity and personalisation. EiTB has been part of the Advisory Board of the project and their interest in the election night, among other use cases, has been highlighted since the beginning. MediaScape made it possible to create a first version of the elections prototype, which afterwards evolved into a real pilot once the project ended.

The developed service has followed a bidirectional approach. On the one hand, it has followed a *Technology push* approach, as the test has been used in order to validate on a large scale the libraries developed in MediaScape. For this reason, all the developed functionality has been included, resulting in a complex application. Whereas, on the other hand, it has been combined with a *Market pull* approach where the broadcaster wanted to provide users an innovative, useful and practical experience to follow the election night in front of the TV.

In greater detail, the service complemented the TV programme with the following content:

1. **Different cameras** located at the headquarters of the five main parties, all of them providing live streaming.

2. **Social network activity** through a list of filtered tweets.

3. **Different graphics and tables** showing all the real-time counting data for each part of the region, as they do in their website.

Furthermore, the user had the option to associate the TV to different devices such as smartphones, tablets or laptops, by scanning a QR code or typing in a shared URL. This was possible through a remote server providing a shared state between different devices [Zorrilla et al.15a], and consequently the devices could be in different networks (e.g. the TV connected to the home Internet gateway and the second screen with 3G/4G. Thus, the application content could be distributed across more than one device being

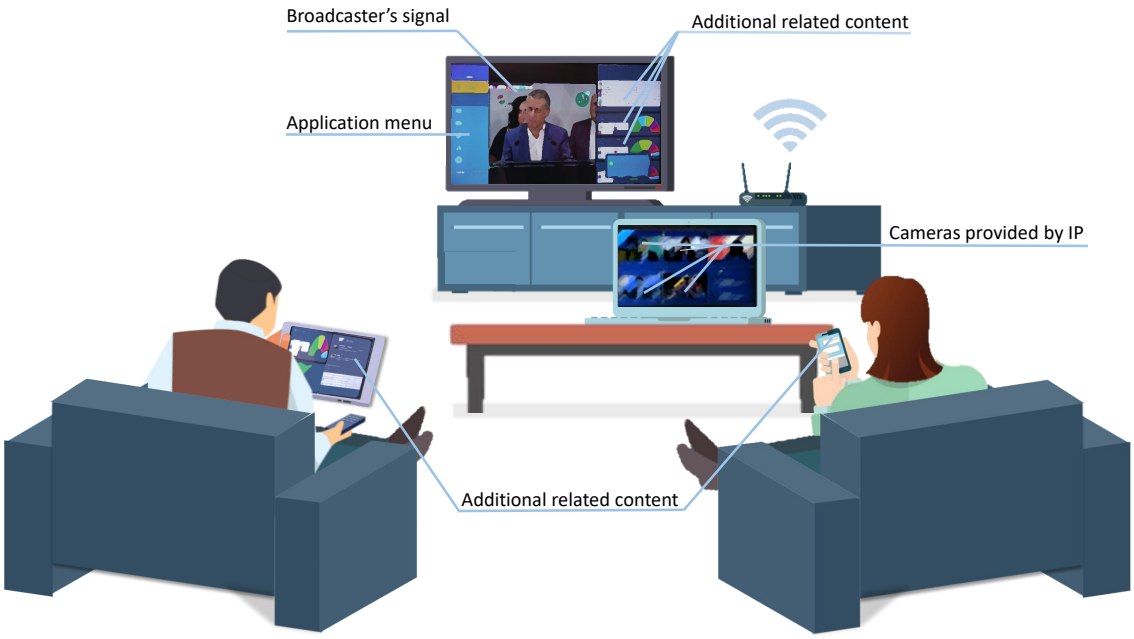


Figure 3.1: Application running on four devices simultaneously. In this case, TV and laptop are shared while users interact and personalise the application through their tablet and smartphone. The complete demonstrator can be found in [vim]

used simultaneously. Finally, all the content could be organised through different templates in each device, giving the user a chance to personalise their experience, both in terms of content and design. Figure 3.1 illustrates an example of the use of the implemented service with two users watching different content types working on four devices simultaneously.

Lastly, it is remarkable that although election day is always a very important event for the broadcaster, with one of the highest audiences of the year, they wanted to test the HbbTV multi-device live service, but of course, under some requirements due to all the risks that a live service in which nothing is pre-recorded entails. These requirements are presented in Table 3.1.

3.4 Implementation and deployment

In this Section we describe the details of the implementation and the real deployment of the interoperable architecture provided in [Zorrilla et al.15a], that provides a unique

Table 3.1: Broadcaster's requirements for the service

Term	Requirement
Access to the service	Access only through the TV as a first device. Red button for broadcaster's official catch-up TV HbbTV service. Blue button for elections application. Service available in the two main channels of the broadcaster ETB1 and ETB2.
Content template	A layout in which no content overlapped the mainstream.
Broadcast always in TV	The broadcast always visible in the TV. Even if technically possible, access to live streams only through second screens.
Control	Every interaction through the second screen also possible through the remote control.
Synchronisation	Synchronisation of all content (media sources and data coming from broadcast and Internet) is not critical. The broadcaster prefers to follow a best-effort approach of providing all the content when it is ready, rather than introducing a delay on the broadcast to enable synchronisation.
Compatibility	Devices having newest browsers, in principle from 2014.
Assume little risk	Limited announcement of the service to end-users. A management application to control the streaming cameras in real time. The broadcaster would be able to switch on/off of the available cameras during the whole emission, to stop individually each one if required.

and consistent experience across multiple connected devices. For this purpose, the *Created Once and Published Everywhere* (COPE) concept was applied during development by means of standard Web technologies. Therefore, the application has been built as an HTML Web page, treating any device (TV set, tablet, smartphone and laptop) as a part of the ecosystem and including the broadcast world simply as a new type of resource. Hence, the application was able to automatically adapt itself to specific target devices, providing the user with a consistent experience through multiple devices at the same time, showing parts of the application on each one of the screens on an adaptive layout.

The carried out implementation addressed two main scientific challenges described in Chapter 2 of these kinds of services, such as the device and service discovery described in [Zorrilla et al.14] and [Ziegler13], and the multi-device adaptation mentioned in [Zorrilla et al.15a], [Zorrilla et al.15b] and [Paternò and Santoro12].

Cross device synchronisation was another relevant scientific challenge for hybrid multi-device services [Zorrilla et al.13] [van Deventer et al.16]. The architecture presented in [Zorrilla et al.15a] enables the synchronisation of all the experience using Shared Motions [motb], and could be easily deployed through the service hosted by Motion Corporation [mota]. However, the synchronisation of all the content (all media sources and data coming from broadcast and Internet) was not critical for the broadcaster in this specific live pilot. Adding synchronisation mechanisms would require to include a delay in the broadcast content. Instead, the broadcaster preferred to follow a best-effort approach of providing all the content when it was ready (see Table 3.1 with the requirements of the broadcaster).

Figure 3.2 depicts the service implementation. In the backend side, concerning the service basic structure, three servers and a database were implemented:

1. Node.js server: provides all the main Web services and libraries as well as the application specific logic.

2. Mongo DB database: this is the database in charge of the mapping service, the one which saves all the sessions and shared data and then distributes them correctly among the connected devices. Therefore, it is the one which guarantees the persistence of the application.

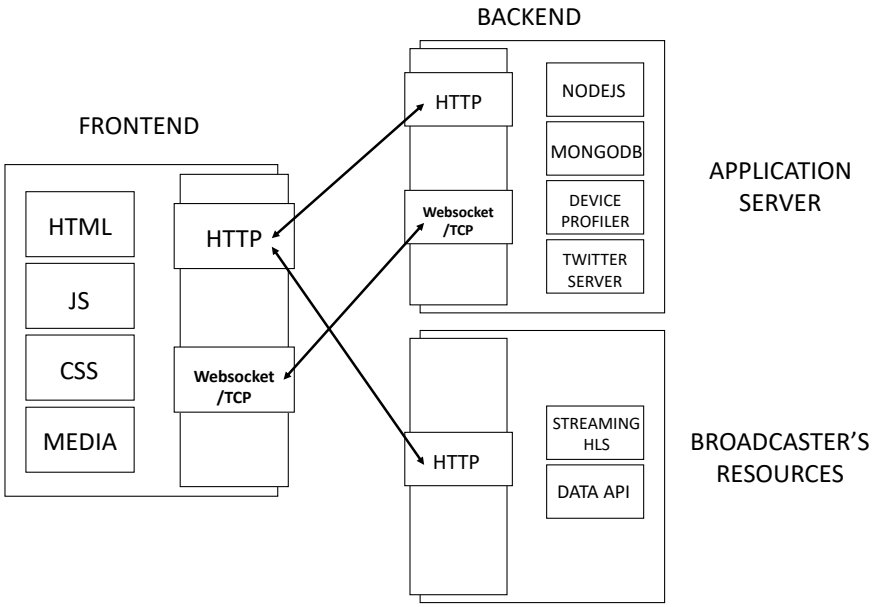


Figure 3.2: Service implementation

3. Device Profiler server: in order to detect the type of devices being used in each session or group of devices, with the aim of distributing the elements of the application through them, and basing the decision on an adaptation rule.

4. Twitter server: responsible for filtering the selected hashtags and accounts from Twitter social network and offering them as a data stream.

Moreover, with regard the application content, the streaming and the necessary counting data for the graphics, these resources were all provided by the broadcaster in real-time by means of five HLS streams and a data API.

Going through the frontend, the application itself was built on top of standard Web Components [web] to create a Web application in terms of functional elements and therefore, to have the option of distributing and adapting the content across all the connected devices. Web components are a set of Web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in Web pages and Web apps.

Finally, the communications between frontend and backend were deployed through HTTP and Websockets.

Table 3.2: Characteristics of the nodes in the deployment

Element	Description	Instances
Load balancer	Nginx static file server 8 cpu 1GB of bandwidth 32GB of memory	1
Node.js	Xeon server 1 cpu at 2,6 GHz 16GB memory	1
Mongo DB	Improved SSD until 20,000 operations per second	1
Device profiler	Xeon server 8GB of memory	3
Twitter server	Xeon server 8GB of memory	3

Concerning the deployment of the servers, the system was scaled in order to cover all the potential users taking into account the initial conditions for the app to work. Referring back to Table 3.1, in principle the application could only work on TVs with the latest browsers. Following the broadcaster's estimates, the number of potential users that could have an HbbTV 1.5 TV (with a newer browser) was 15,000, so the deployment addressed the most challenging (very unlikely situation) scenario where all the potential users access the application concurrently. The deployment of the servers were hosted in Amazon [aws] scaling each server according to its design specifications for the estimated load.

The deployed architecture remained as shown in Figure 3.3 and the characteristics of each one of the nodes correspond with the ones in Table 3.2.

3.5 Results

This section provides a report on the results obtained during the pilot test. All these results have been extracted from the log of the application server, which saved all the User Agent strings of each connected device, and from the analytics modules added both to the Web application and to the servers. The methodology employed has been

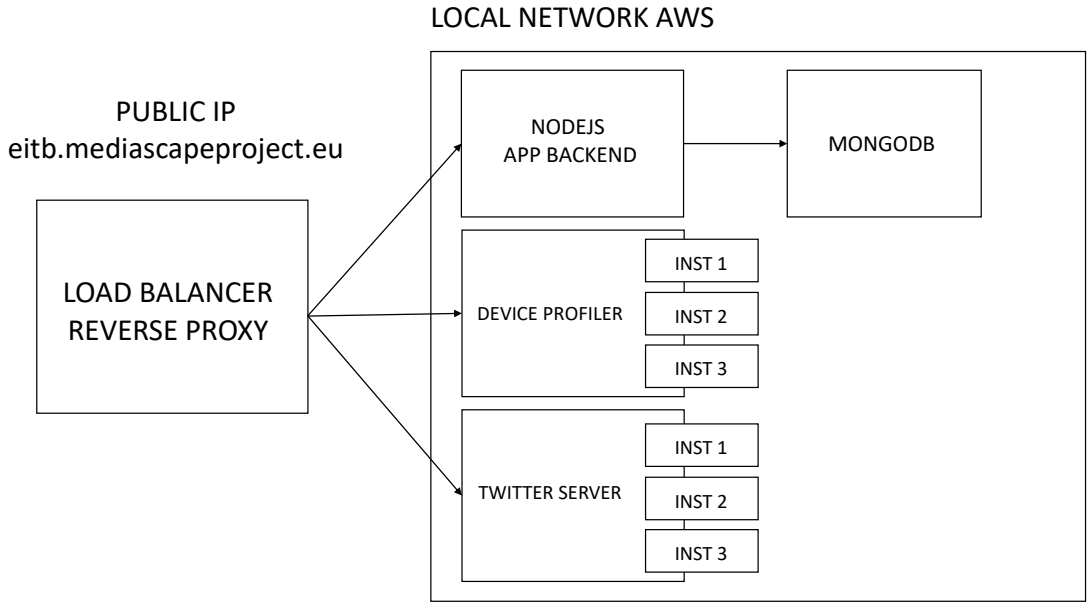


Figure 3.3: Service deployment in Amazon

to analyse all the terms appearing in each User Agent and group all the connected devices according to varying parameters. From this point, the extracted data from the User Agent strings are presented regarding the general terms, the brand, the age of the models, the HbbTV version, the time using the application, the number of sessions and the type of associated devices. Moreover, the results obtained have been compared and contrasted with the data provided by the analytics in order to verify them.

3.5.1 General results

As mentioned in Section 3.4, the amount of concurrent potential users for the service was 15,000. From this point, the presented data correspond to the interval of time between 06:00 pm 25th September 2016, when the server was switched on, and 06:30 am 26th September 2016, when the server was switched off. In that interval, the broadcaster had an audience of 1,161,000, from which 160,000 watched the whole election night TV programme. From this audience, in order to calculate the number of HbbTV devices that received the "press blue button" message, the broadcaster has only the data of HbbTV devices per hour interval. Those data show that the maximum number

Table 3.3: Hbbtv summary

HbbTV version	Unique sessions		Worked		Failed	
	Value	%	Value	%	Value	%
1.0	206	17%	51	25%	155	75%
1.5	974	82%	593	61%	382	39%
2.0	15	1%	9	60%	6	40%
TOTAL	1195	100%	652	55%	543	45%

of devices in the one hour interval was 18,942 from 20:00 to 21:00. Then analysing the rest of the intervals, the estimated total number of single devices was 25,000. Finally, of that number, there were 1,195 users of HbbTV devices who tried to run the application. When classifying the TVs in terms of working or not working devices in the pilot test, the load point in which all the needed files were requested and the content with the main menu was rendered has been taken as a threshold. From this point, when talking about working or not working devices, the mentioned criteria is used. Thus, of the 1,195 users that tried to start the application, 652 devices (55%) succeeded while 543 (45%) failed.

3.5.2 **HbbTV version**

Table 3.3 shows the summary of the data in terms of the HbbTV version. According to the registered User Agents, of the 1,195 HbbTV connected devices, 82% were HbbTV 1.5 while 17% were HbbTV 1.1 and 1% were HbbTV 2.0. Regarding their performance, a quantum leap is appreciated from HbbTV 1.0 to HbbTV 1.5, increasing the working success by 36 percentage points. Moreover, although HbbTV 2.0 devices were detected through the User Agent string, there are no commercial devices with this version completely implemented, so they could be considered as HbbTV 1.5 devices, as the success rate is almost the same.

3.5.3 **Brands and models**

Table 3.4 shows the summary of the data in terms of brands. As can be seen, there were TVs of many different brands, with the majority belonging to only a few brands. LG was the most used (28% of users had an LG) and was also the one in which the application worked best (78% of the LG devices worked). There was also a significant amount of

Table 3.4: Brands summary

Brand	Single Sesions		Worked		Failed	
	Value	%	Value	%	Value	%
LG	332	28%	258	78%	74	22%
Samsung	305	25%	166	54%	139	46%
Panasonic	294	25%	112	38%	182	62%
Sony	171	14%	93	54%	78	46%
Philips	61	5%	14	23%	47	77%
Hisense	8	1%	4	50%	4	50%
Other	24	2%	5	21%	19	79%
TOTAL	1195	100%	652	55%	543	45%

Samsung (25%), Panasonic (25%) and Sony (14%) devices. However, the success percentage decreases in comparison to the LGs, being 54% for Samsung and Sony, and 38% for Panasonic.

On the other hand, Figure 3.4 shows the results taking into account the age of the TV models trying to load the application. There were only 7 TVs from 2011 and none of them worked. From 2012, there were more TVs, but most of them did not work. Out of interest, the only working brand was LG. Looking at TVs from 2013, a meaningful increase of the total amount of TVs can be seen. A considerable number of them did not work, but the number of working LGs increased and some Sonys also worked. Most of the models accessing the application were from 2014 and 2015. Looking at 2014, TVs of all brands worked and for the first time, there were more working TVs than not working TVs. Referring back to the broadcaster's requirement for devices from 2014 or newer, 2015 had the highest number of working TVs. However, for 2016 year there was not enough data to draw conclusions. There were not a lot of TVs and besides, a high percentage of them did not work when the newest models are supposed to work better.

3.5.4 **Associated devices**

As described previously, the user had the option to associate different devices to the TV, creating a unique experience around the TV. Of the 652 working TVs, there were 619 (95%) users who used only the TV and 33 users who successfully associated another

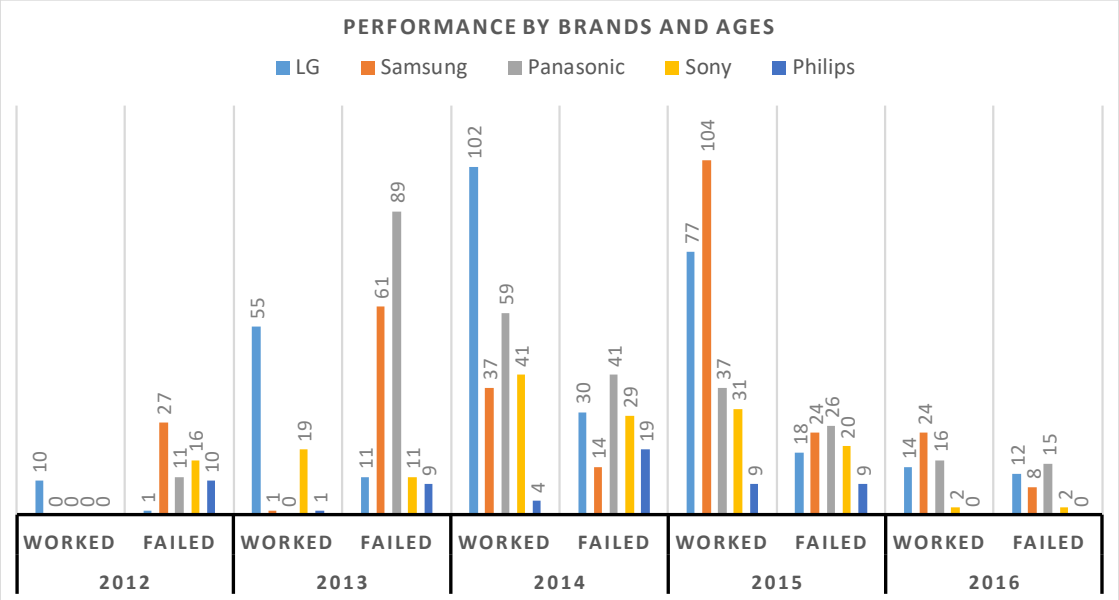


Figure 3.4: Summary of the performance by brands and ages

Table 3.5: Type of associated devices

Device	Unique sessions			%
smartphone	22	Android	20	62.86%
		iOS	2	
Tablet	11	Android	7	31.43%
		iOS	4	
Laptop	2		Windows	5.71%
Total	35			100%

device to the TV. Most of these users, 31 (4.7%), associated only one device and only 2 (0.3%) associated two devices.

Table 3.5 summarizes the type of the mentioned 35 second and third screens, most of them being smartphones and tablets.

3.5.5 Time spent using application

During the same time interval, Figure 3.5, Figure 3.6, Figure 3.7 and Figure 3.8 show how long users spent using the application. The time interval scale for these graphs has been chosen according to that used by Google Analytics [ana] in a parallel analysis. In the

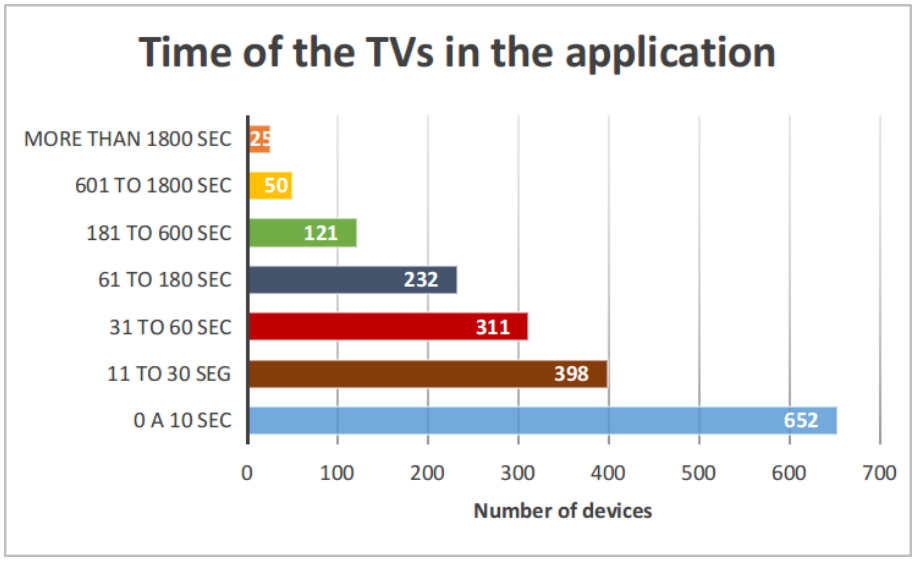


Figure 3.5: Time spent using the application on TVs classifying the devices along all the different durations.

case of the TVs, 61% of the devices stayed connected more than 10 seconds whereas for second screens this percentage was even higher (91%). So connecting a second screen device may indicate a higher interest in the service. Apart from that, concerning the TVs, most of the users (75%) opened the application once, 15% did so twice and the rest 3 or more times. The same happens with the second and third screen devices, with 46% of users connecting once, 31% twice and the remaining 3 or more times.

Finally, the maximum number of simultaneous working devices was 54 at 19:24:44. This fact is far from the 15,000 potential concurrent users estimated in the most challenging scenario, thus indicating that the deployment architecture has been under-used. Figure 3.9 shows the concurrency during the whole time the service was available.

3.6 Discussion and lessons learned

Once the obtained results have been presented, this section provides a set of lessons learned with the aim of discussing different aspects of hybrid broadcast-Internet multi-device services. The discussion has been driven by giving answers to the questions formulated prior to the test and presented in Section II. The summary of the lessons learned is provided in Table 3.6.

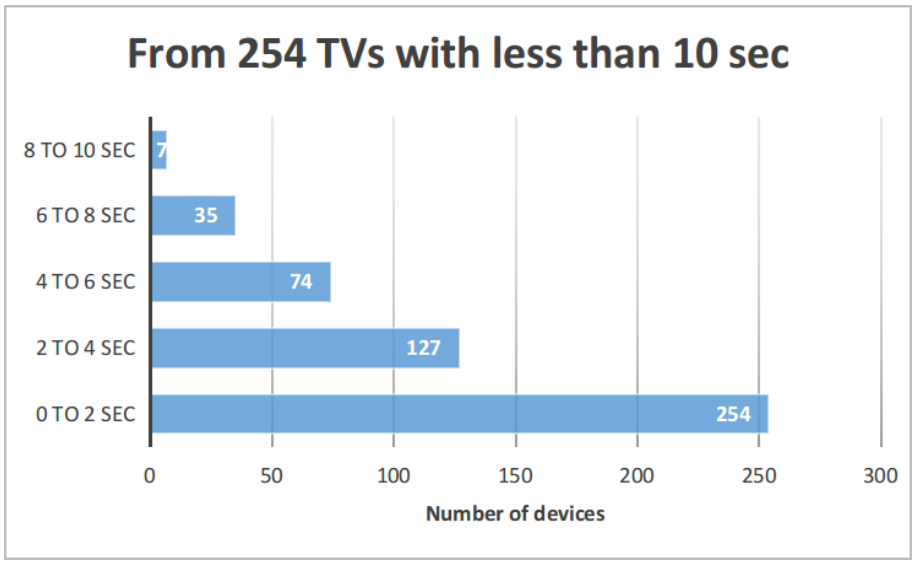


Figure 3.6: Time spent using the application on TVs showing the detail of the devices being in the application less than 10 seconds.

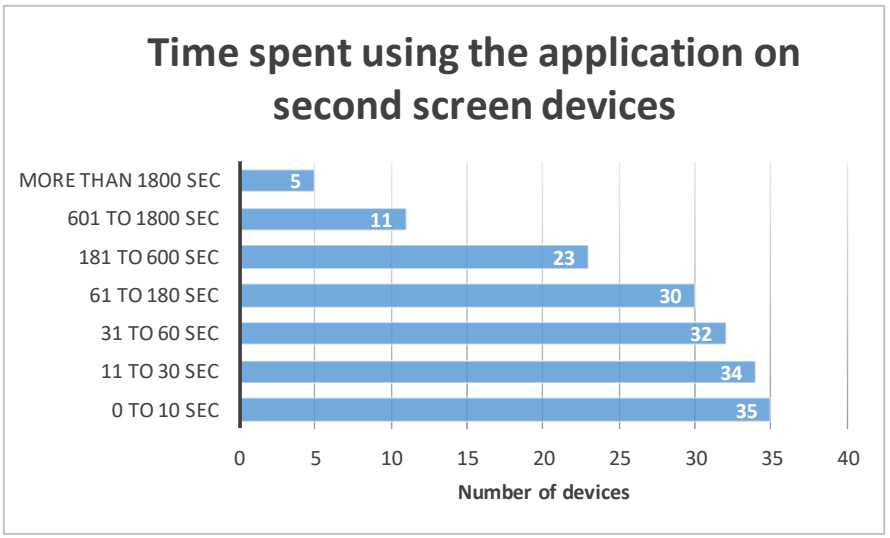


Figure 3.7: Time spent using the application on second screen devices

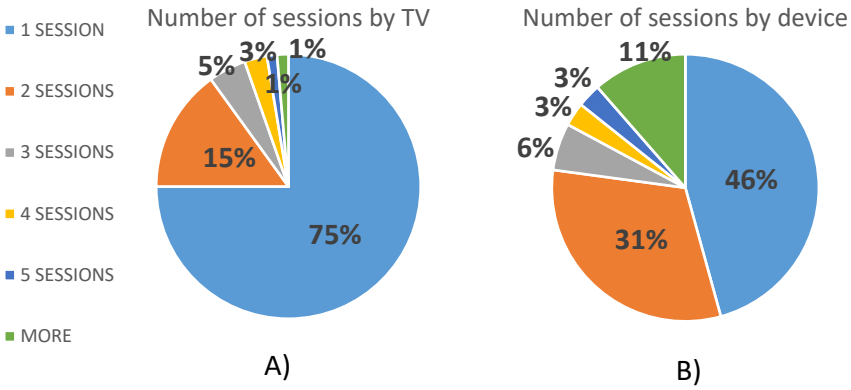


Figure 3.8: Number of sessions by device. A) Shows the sessions in TVs and B) shows the same data for the rest of devices

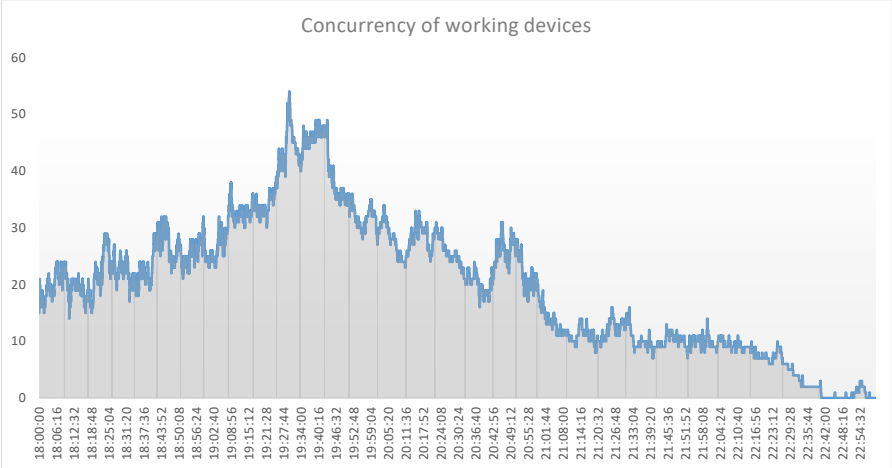


Figure 3.9: Concurrency of working devices from 18:00 to 23:00. The data from 23:00 to 06:30 are not shown, as they are negligible.

Table 3.6: Summary of lessons learned

Question	**Lessons learned**
1. Are broadcast-driven hybrid multi-device services accompanied by a live TV programme interesting for users?	Broadcast-driven hybrid multi-device services with a live TV programme are interesting but still unknown for the users.
2. Are broadcasters ready to cope with hybrid multi-device experiences?	Broadcasters are aware they have to adapt to new consumption experiences within the new hybrid ecosystem but they still have uncertainties about the technology and business models.
3. Are HbbTV implementations mature enough for broadcast-driven multi-device services?	Although HbbTV provides a set of standards that lay the foundations for hybrid services, it needs to overcome ambiguities in the multi-device field towards a consistent and mature specification.
4. Are HTML5 and HbbTV convergent enough yet to solve interoperability problems?	Despite newer HbbTV versions that increasingly include HTML5 features, current HbbTV commercial devices do not provide complete convergence between HbbTV and HTML5.
5. Are hardware and processing capabilities of the current TV sets enough to run broadcast-driven hybrid services smoothly?	Hardware and processing capabilities of current TV sets are not powerful enough compared to laptops or smartphones to run broadcast-driven hybrid multi-device services smoothly.

3.6.1 **Answer to question 1: Broadcast-driven hybrid multi-device services with a live TV programme are interesting but unknown for the users.**

Question 1 formulated that services offering a complete environment like the one described in the paper should succeed among users, based on the fact that TV consumption habits are moving towards the use of a second screen. However, since there are no broadcast-driven hybrid services for live programmes yet, users do not have a reference on which to base their expectations and they need to know what advantages there are to the service and have guidelines on its use.

The limited variety of offered HbbTV catch-up TV services together with the lack of advertising and user guides does not help and causes a loss of interest among users. There is such a degree of dissatisfaction that, according to the broadcaster, some people call to ask how the "red button message" could be removed. They may see it as a useless, unknown and hardly beneficial service when in fact, it could be just the opposite.

In the pilot test, the service was hardly advertised and even less so presented as an advantageous system to follow the election night in a personalised way. This could explain the low amount of users that pressed the blue button to run the application. Of the 25,000 TVs that showed the blue button message, only 1,195 pressed it, just 4.78% (See Section 3.5.1). Furthermore, the fact of having two messages at the same time for different applications, as seen in Table 3.1, could have distracted or annoyed the users. Without the appropriate information about the service, users might have thought that it was another catch-up TV service.

On the other hand, using a second screen was a difficult task, probably because of the lack of habit of the users associating devices in the same session while watching TV. Seeing the results in Section 3.5.4, of the total amount of working TVs, only the 5% of them associated a second screen device to the TV. Therefore, even though the concept of the second screen seems to be embraced nowadays, at least when using independent applications, the second screen percentage was too low.

In this context, it could be surmised that the User Interface becomes a key factor in the design and development of hybrid broadcast-Internet multi-device services. Following with the association of devices in the pilot test, the mechanisms used for the association were the scanning of a QR code and the typing of a supplied URL. Perhaps,

this represented an excessive task while being in front of the TV and thus, more transparent mechanisms are needed. For instance, this could be in the form of a notification arriving to the device connected to the same home network as the TV. For that, the new HbbTV 2.0 specification includes a Companion Screen discovery API, enabling HbbTV applications to discover Companion Screens for launching and installing Companion Screen applications. This feature surely will help make broadcast-driven multi-device live services more widespread.

Finally, it should be mentioned that although the number of second screens was low, users with a second screen stayed connected to the application longer as can be seen in Section 3.5.5. Therefore, it could be considered that users knowing how to associate a second screen enjoyed the service and promoted larger consumption time in the TV programme.

To sum up, this kind of services seem to be interesting but unknown for users, both in terms of existence and usability. Therefore, some efforts should be made to focus on the user, who ultimately decides whether the services succeed or not. First of all, the services have to be presented as beneficial systems, announcing them as much as possible. Second, an intuitive and usable User Interface has to be built across all the dimensions of the application. And third, some surveys should be prepared for future studies in order to collect the opinions of the users about the application provided. With these facts, we believe the user experience will substantially improve, and consequently, broadcast-driven hybrid multi-device services together with a live program could succeed.

3.6.2 **Answer to question 2: Broadcasters are aware they have to adapt to new consumption experiences within the new hybrid ecosystem but they still have uncertainties about the technology and business models.**

Question 2 considered that broadcasters feel the current technology ecosystem is too unstable, uncontrolled and immature to provide broadcast-driven multi-device services together with a live programme in the highly regulated environment in which they work.

In practice, broadcasters are interested in testing innovative live multi-device services such as the one described in the paper, but are also being scrupulously prudent about it. The aforementioned pilot test has been a good example for that. The broadcaster wanted to test the application and make it available for the whole audience during such an important event like election night. However, as explained before, the lack of information available to the user could have been a critical factor for such a low number of users successfully using and enjoying the service.

The broadcasters worrying about the highly regulated environment in which they operate was also checked. The broadcaster was particularly prudent in controlling every detail of the provided innovative service. This was clear in the days prior to the test, as they asked for a mechanism to be able to switch off the streaming cameras that were available during the whole emission, in case someone was saying something he/she should not.

Another measure taken was a default layout for the application, avoiding overlaps of any content with the mainstream (see Figure 3.1) and making it clear what the most important thing was for them. Furthermore, they expressly required the mainstream to be always on the TV, not allowing it to be removed or changed to another IP video signal. The main reason was their business model based on the results provided by audience counters. These audience counters take into account only devices showing the broadcast signal including the audio, discarding any other overlapping or replacing it, even if the additional content comes from the same broadcaster signal. The same happened with the advertisements, which gives them a much smaller profit when shown through an HbbTV application.

Therefore, it seems that broadcasters will not provide an official broadcast-driven multi-device service for a live programme until they are confident it will not cause them any problems and will not affect TV audience ratings, until they are convinced of the user benefit to justify the development of multi-device experiences, until they are confident users will embrace them positively, and especially not until they see a profitable business in using it. In this context, they should encourage simple and automatic developments that help them to settle in the hybrid ecosystem without having to worry about the adaptation of the services to the wide range of available devices. Furthermore, a variety of those kinds of services offered as a trial versions and starting with low-audience programs would provide very valuable data to improve the developments. That way,

users would start getting used to these kinds of experiences and would increasingly demand more of them, and manufacturers and advertisers would find themselves obliged to work in this direction too.

3.6.3 **Answer to question 3: Although HbbTV provides a set of standards that lay the foundations for hybrid services, it needs to overcome ambiguities in the multi-device field towards a consistent and mature specification.**

According to Question 3, even if there is a standard to follow, due to the recent incorporation of advanced Web technologies and multi-device features, there is a lack of reference implementation of the standard in this area. Consequently, there is a wide variety of levels of implementation of the standard in each TV set, and most of the times these differences are not only down to the HbbTV version.

From the data obtained during the test, some devices appeared in the server log with the "HbbTV 1.3.1" term in their User Agent. The HbbTV 1.3.1 version maps to HbbTV 2.0 new standard, which is not supposed to have any commercial TV set available, at least in Spain. Thus, they probably had some features of the latest update implemented, but there is no way they could count on the whole standard.

On the other hand, Question 3 also stated that there are differences between TV sets from different manufacturers even if they have the same HbbTV version. As it has been seen in Section 3.5.3, there were differences in the performance depending on the brand and the age of the TV, as well as among the software integrated in each TV. Each brand has its own software (including its HbbTV browser characteristics), and no two Smart TV platforms are alike. Actually, this fragmentation is not only among TV brands and ages, but also across different models of the same age as can be deduced from results in Figure 3.4. In that graph, focusing on the same brand over the years, it can be seen that performance was different depending on the year, but also within models of the same year. Moreover, it is up to the user to update the firmware, leading to even more variance. This greatly complicates the development of interoperable HbbTV applications, as there is no common pattern when programming and each TV has to be tested separately, which is costly and time consuming. In case of the tested multi-device application, five different TV models had been tested before the pilot test: two different

LG models, two Samsung and one Panasonic. Each TV presented a different problem that had to be specifically solved. In addition, Panasonic, which was the most tested TV before the testing day and was supposed to work best, surprisingly turned out to be the brand with the highest number of not working TVs.

Therefore, the partial implementation of the standard and the fragmented market being in need of consolidation are creating a kind of barrier to broadcast-driven hybrid multi-device services, when the entire Smart TV industry should look to standardisation in order to have a consistent and mature specification. That way, it would be easier to provide more choice and better experiences.

3.6.4 **Answer to question 4: Despite newer HbbTV versions that increasingly include HTML5 features, current HbbTV commercial devices do not provide complete convergence between HbbTV and HTML5.**

Question 4 considered that there are heterogeneous browsers among TV sets because they typically have more W3C features than the minimum required by the HbbTV standard. This was also checked in the pilot test.

In Section 3.5.2, it has been seen that there were differences in performance depending on the HbbTV version of the TV on which the application ran. As previously mentioned, the application in principle worked on devices with the most recent browsers. However, Table 3.3 shows that 25% of the HbbTV 1.1 TVs worked. This means that the compatibility problem is not related with the HbbTV version but with the extra HTML5 features implemented in the browser. But of course, there is a direct relation due to the fact that HbbTV 1.0 TVs are older. Therefore, compatibility is worse because the browser is older and it has less features implemented.

More specifically, the main problem for compatibility was the use of standard Web Components. As seen in section 3.4, the front-end was built on the top this standard to create an application in terms of functional elements. Web Components can be used with any JavaScript library or framework that works with HTML. They are based on four specifications (Custom Elements [cus], Shadow DOM [sha], HTML Imports [htma], and HTML Template [htmb]) which are only supported natively by Chrome [chr] and Opera

[ope] browsers. However, a polyfill [wcp] was used, which is a code in order to simulate the missing browser capabilities as closely as possible. The fact here is that this polyfill uses several HTML5 features that not all the browsers of the commercial TVs have implemented yet. Some efforts were made to overcome that problem, but in the end, each TV is equipped with a different browser having different levels of implementation of the HTML5 standard and complete compatibility is no easy task.

Another detected concern regarding interoperability was the difference on the strictness that the browser requires of application developers. The same application that worked perfectly on a Panasonic VIERA 2014, did not even start in a newer Samsung SUHD 2016. The reason was that the code syntax was not strictly programmed and things that were assumed negligible for the application to work caused the service go down with the first file request made to the server. The kind of things that ruined the application on that TV were, for example, an empty line on the first row of the code of the requested file containing the HbbTV header and a big amount of file dependencies, or even the way in which HTML tags were closed (`<link/>` instead of `<link></link>`). With such a high level of programming strictness happening only in some cases, it is very difficult to build a completely interoperable application. Thus, the rigorous definition of these requirements would also help towards the spread of HbbTV multi-device services. It is remarkable that the application has been developed following the standard rigorously, but in the same way these examples have been identified, other similar questions could be the problem in some TV sets.

Then, reducing the differences between the wide range of TV set browsers and defining the minimum required HTML5 features could mean decreasing cross-device differences for broadcast-driven multi-device services for a live programme following the aforementioned Created Once and Published Everywhere (COPE) concept, instead of creating multiple applications for each one of the target devices or platforms. Apart from that, more automatic developments would avoid complex implementation tasks and adaptation rules definition, that usually fall to broadcasters.

3.6.5 **Answer to question 5: Hardware and processing capabilities of current TV sets are not powerful enough compared to laptops or smartphones to run broadcast-driven hybrid multi-device services smoothly.**

Question 5 considered that TV sets could be one step behind other devices in terms of processing capabilities with regard COPE paradigm based applications.

When classifying the TVs into working or not working devices in the pilot test, the load point in which all the needed files were requested and the content with the main menu was rendered has been taken as a threshold. The estimated average loading time was 30 seconds, which is quite high compared to applications we are used to running on a computer (10 sec), or even on a smartphone (15 sec). Thus, it is quite possible that the user, bored while waiting, closed the application or changed the TV channel before loading was complete.

This makes it clear that TV sets have poor processing capabilities and running anything more complicated than basic streaming becomes a challenge. If we want the TV set to be the central hub in the connected devices ecosystem, then in order to run these kinds of applications together with laptops or mobiles, a significant update to the TV sets' hardware is required. Manufacturers should consider supplying them with more powerful resources in order to be able to run technologies as HTML5, Javascript and CSS smoothly. If not, broadcast-driven hybrid multi-device services, such as the one described in this paper, will continue to be too slow in order to retain the end-user, and they will continue using other devices for that purpose.

A complementary solution to the loading speed of HbbTV applications would be to save the application in the cache when instructed by the end-users.

Additionally, the application could be maintained running even when the channel is changed. However, this means some hard negotiating among different broadcasters competing for the same audience and they are certainly not going to permit content from the competence to obscure their mainstream.

Finally, taking into account the time required and the difficulties to apply all the previous recommendations, more efficient developments should be carried out, light enough to be better processed by the poor hardware provided to the available smart

TVs. A good example for that would be to generate an automatic adaptation model that avoids to overload the smart TV with many different contents.

3.7 Conclusions

This chapter provides the definition of an innovative hybrid broadcast-Internet multi-device service for a live TV programme, its large scale pilot deployment based on the architecture presented in [Zorrilla et al.15a], and lastly, all the lessons learned and conclusions drawn during the large scale pilot with the Public Basque Broadcaster EiTB.

This work presents some questions formulated regarding the large scale pilot in order to provide such a service for live TV programmes, and then provides an answer for those hypotheses based on the results of the pilot.

The following aspects are the most important conclusions, summarising the lessons learned when creating and deploying hybrid broadcast-Internet multi-device services for a live TV programme:

- The whole Smart TV industry, from the specifications and standards to the manufacturers, should look into the rigorous definition of some minimum requirements in terms of both hardware and software so as to decrease cross-device differences in relation to broadcast-driven hybrid multi-device services. Developers should also work on more automatic developments that allow to create COPE based services that do not require complex implementations and the definition of a high number of adaptation rules.

- That way, broadcasters, which need time and consensus to upgrade to new technologies, would change their perception of an immature and unstable environment to another one full of opportunities for new business models. Hence, they would be convinced of the end-user benefit to justify the development of multi-device experiences and they would start opting for broadcast-driven multi-device services to enrich TV programmes. They would be motivated to test trial versions of a wide variety of services, advertising them and ensuring that users know about their existence, know how to use them and appreciate them.

- Consequently, users, who at the moment do not see many advantages to the available HbbTV catch-up services, would also change their mind. They would see broadcast-driven hybrid multi-device services as comfortable applications that provide them with information that they would otherwise have to look for on their own. They would increasingly demand them and would renew their TV set more often. Furthermore, seeing the trend of second screen consumption habits, surely these kind of services for certain live programs would be an absolute success.

- Moreover, it has to be said that despite the difficulties identified today, software and hardware capabilities of the TV sets may improve over time, and be capable of running hybrid services smoothly on TV sets in a mid-term. The reason is that manufacturers are supplying more and more advanced features and resources to their new models, but these upgrades are slower than desired. So if an acceleration of the widespread and success of broadcast-driven hybrid multi-device services is wanted, more efficient and automatic services should be developed, light enough to be processed by smart TVs at the same level as it is done in the rest of devices.

- Lastly, it has to be taken into account that every effort for improvement has to be focused on the user who is the deciding factor in whether the application succeeds or not. In this context, the User Interface design is the key in guiding the user across all the connected devices that are placed in a three dimensional space.

These conclusions encouraged us to go in depth in a research line focused on the definition of an adaptation model that automatically generates usable user interfaces of multi-device services basing its decisions on the characterisation of all the involved elements and a high number of design factors. The mentioned adaptation model is seen as a tool to ease broadcasters the adaptation of their contents to any connected device or set of devices being used simultaneously, taking into account the context information and ready for the technological updates that will undoubtedly arrive. All of this will be managed through the definition of some parameters instead of a high number of explicit rules and the model will allow to run the adaptation process efficiently in all the connected devices. This work is presented in Chapter 4.

CHAPTER

4

Adaptation methodology and model

4.1 Overview

New audiovisual experiences involve several contents through multiple internet-connected devices. The TV is still the central hub of the living room, but it is often used simultaneously with other screens. Consequently, the user has the chance to consume all different contents at once across multiple devices. However, no existing adaptation models are available to dynamically adapt such a multitude of contents in multi-device contexts. To address this gap, this chapter proposes a novel multi-device adaptation methodology to build adaptive user interfaces for multi-screen hybrid broadcast-broadband TV experiences that is extensible to any kind of content, device and user, and is applicable to different contexts considering technological evolution. The proposed methodology is the outcome of extensive research that arose from a previous multi-device media service deployment with broadcasters, which has shown a set of hints to consider. The methodology provides a model which is formally described together with examples of its implementation. Additionally, the methodology is validated in terms of quality, efficiency and universality, including all the results in this paper.

4.2 Overview of the adaptation methodology

The aim of the adaptation is to provide a suitable distribution of the contents of a hybrid broadcast/broadband programme into different device screens in the living room. This content should be provided proactively and responsively, according to the context conditions and changes.

The proposed adaptation methodology aims to define a general multi-device adaptation model for hybrid broadcast/broadband services, such as the one deployed in [Domínguez et al.17]. However, instead of creating specific adaptation rules for each TV programme, they are made extensible for any kind of situation: any kind of TV programme or content, any device type, and any kind of UI layout template, to be used by any kind of user and in any context, and applicable even if technological changes occur (including new devices going further than AR/VR headsets or smart watches, new context conditions, new ways of interaction with TV or media, etc.).

This novel methodology is the outcome of an extensive research that identifies the elements, properties and criteria that should take part in the adaptation process [Zorrilla et al.15a] [Domínguez et al.17] [Zorrilla et al.13] [Dominguez et al.18] [Dominguez et al.19]. More specifically, our approach is based on the definition of a general adaptation model that allows efficient implementations to be derived for the target multi-device broadcast environments. The effect of the context in the final user interface outcome is also considered. Figure 4.1 shows an overview of the approach.

In the following Sections, the adaptation model is described in detail, while a specific implementation of the model is presented in Section 4.5. The adaptation model is described in three main parts:

1. Identification and characterization of the elements of the user interface (Section 4.3).

2. A flexible, modular, two-step characterization of the adaptation process based on the aforementioned UI elements (Section 4.4).

3. Finally, the role of the context as a drive of the adaptation process, which will be described on the basis of the implementation example (Section 4.5.8).

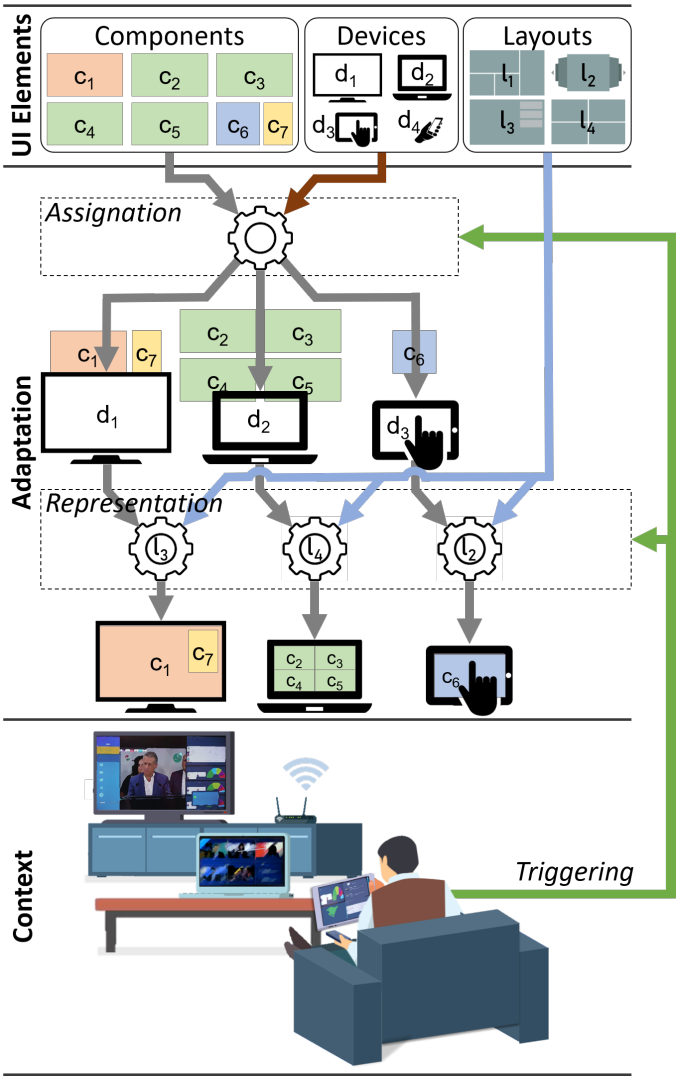


Figure 4.1: Overview of the proposed methodology

To summarize, three types of elements of the user interface have been identified. On one hand, a hybrid broadcast/broadband media service will have different pieces of content, such as the main programme (typically the broadcast), other media resources (multi-view live cameras, on-demand complementary content, etc.) and other types of information (statistics, graphics, banners, etc.). This paper refers to all these elements related to the content as *components*. On the other hand, the *devices* that end-users will use to consume the content are also crucial elements to build an adaptive multi-device user interface. Finally, every screen taking part in the multi-device experience needs to visually organise the components in the display, assigning a location, size and aspect ratio to each of the components, creating user interface *layouts*.

Regarding the adaptation strategy, we divide adaptation into two goals:

1. Maximizing the quality of the distribution of the visual content components across the connected devices.

2. Maximizing the quality of the user interface layout in each device, given the components assigned to it.

Then, the adaptation can be seen as being composed of two steps, that are formalised as functions in the adaptation model, which we call *assignation* and *representation* respectively.

These two goals are interdependent regarding the general goal of maximizing the adaptation process. However, we address the partial goals separately to not only have a clearer description of the adaptation model, but to also obtain a framework for efficient two-step implementations.

The rationale behind this divide-and-conquer strategy is as follows:

- **Conceptual simplicity:** The adaptation problem is divided in two independent, well-defined sub-problems.

- **Performance:** As an optimisation problem, the computational complexity is dramatically reduced when the problem is divided, which is crucial in scenarios that include devices with limited computational resources. Furthermore, decoupling both goals in a two-step implementation allows for a more efficient distribution of processing in the second step.

- **Responsive answer to context changes:** The adaptation can be incremental, often involving the layout generation in just an individual device.

- **A suboptimal result might be preferable:** Finding an optimal solution requires addressing the adaptation problem as a whole. Nevertheless, responsiveness and low latencies are relevant requirements in a UI design, and optimality can be sacrificed if the obtained solution is reasonably good.

- **Previous experience:** A prototype of a two-step adaptation has been tested in a previous pilot [Domínguez et al.17] with positive results. Additionally, broadcast professionals and researchers that were involved in the pilot can better contribute to the definition, implementation and validation of the defined model, overcoming the limitations of the pilot where specific adaptation rules need to be created.

Finally, regarding the context, the proposed approach is prepared to consider all possible circumstances that might influence the adaptation process. This context covers a full spectrum of situations, starting from the preferences and the interaction of end-users, environmental conditions, including lighting or acoustics, available connection bandwidth or processing capabilities, battery condition on mobile devices, the number of components that are being displayed on a device, etc. All of these context situations will feed the adaptation process and drive a new iteration when required.

The following sections formalise the definition of the main blocks proposed in the adaptation methodology.

4.3 User interface elements

To provide an adaptive multi-device user interface, it is important to identify the main elements, which are independent from the context and that can be characterised or specified by the application developer: the content components, the target devices, and the UI layouts.

4.3.1 Content characterisation and componentisation

In recent years the trend of componentising the Web [Savage15] has succeeded among most of the frameworks for web applications development. “Web components” [web] is an umbrella term for a handful of new specifications that give developers the primitives needed to build such interoperable, platform-level features. This componentisation, or modularisation of the content, allows for building a distributed system where each component can be easily moved from one device to another. Therefore, the entire content is divided into logic elements on top of the Web components standard, following the object-based broadcasting approach [Armstrong et al.14]. As a broadcast element (e.g., the main programme, an advertisement or a secondary video) is supported by a web component, we will refer to these broadcast elements as *components*.

The set of components for a specific hybrid broadcast-broadband programme can be formally defined as:

$$C = \{c_1, c_2, ..., c_{N_C}\} \tag{4.1}$$

For simplicity, we will also refer to a single component as c.

Each component could be characterised in terms of a set of properties [bbc][Sarkis et al.18], such as:

- **Attention**: The demand of attention required for that specific component regarding the entire multi-device media service, from the perspective of the content provider of the broadcaster. For example, the main programme is supposed to require high attention.
- **Interactivity**: The degree of interaction that the component allows depending on its type. For example, a chat component requires high interactivity.
- **Processing Requirements**: The CPU/GPU processing and memory demands of a component (e.g. real-time decoding of H265 video streams).
- **Broadcast Requirements**: The demand of a broadcast tuner to decode the TV content.
- **Confidentiality**: The level of privacy required by a component, since it could include personal or customised information that might not be of interest for other

viewers in the same physical space. This is the case, for example, of a secondary video, different from the main programme.

- **Geolocation necessity**: To allow for personalised, location aware setting of the content of a component.
- **Concentration**: The cognitive load demanded by the viewer to assimilate the information provided by a specific component. For example, concentration demand is higher for a social content than for an advertisement.
- **Source**: The creator of the component content, which is relevant, for example, to characterize the user generated content.

This set of properties could be extended to other properties depending on the use case, but we will consider them as stable and independent from the environment, and we will refer to them repeatedly. Nevertheless, for generalisation purposes, we define a set of formal properties:

$$P^C = \{p_1, p_2, ..., p_{N_P}\} \tag{4.2}$$

As we will see, the set of properties can be adapted in a flexible implementation. For example, to describe a simple implementation where $N_P = 3$, we will consider the properties p_1 = attention, p_2 = interactivity and p_3 = processing requirements.

For a given component c, a property vector $P^C[c]$ represents the values of each property, normalized in the range $[0,1]$. In our example, for a component c (e.g., the main programme) the property vector could be:

$$P^C[c] = [1, 0, 0.2] \tag{4.3}$$

which means that the component would fully require attention, would not require any interactivity and would present a low processing demand. Note that for simplicity we are using the same notation for both the set of properties and the property vector.

All of the components in a service could be evaluated in terms of properties that would be quantified differently depending on the content type. To simplify the assignation of the property values to each of the components across the adaptation process, a classification of the components in *component types* enables predefining a set of property values to each component depending on its type.

In general, the set of component types is defined as T^C, and we consider a surjective function that maps each component onto a component type:

$$\mathscr{T}^{\mathscr{C}} : C \rightarrow T^C \tag{4.4}$$

Therefore we can, without loss of generality, associate the same property vector to every component of the same type:

$$P^C[\mathscr{T}^{\mathscr{C}}(c)] = P^C[c] \tag{4.5}$$

An automatic typification of components is proposed in [Dominguez et al.18] and used in our implementation (see Section 4.5).

4.3.2 Device characterisation

Applications developed using web technologies can be run on laptops, desktops, tablets, smartphones, smart TVs and even on consoles or other types of present or future devices. The only requirement is a Web browser and typically a connection to the Internet.

Formally, the set of active devices in a given multi-device experience can be defined as follows:

$$D = \{d_1, d_2, ..., d_{N_D}\} \tag{4.6}$$

Again, there are a variety of properties that allow for differentiating one device from another and are very useful to consider during the adaptation process. For illustrative purposes, here we define an informal set of properties that can be used to characterise the current devices:

- **Screen Size**: The size of the display of the device. Trivially, the TV would have a large size and smartphones a small one. However, this property could integrate other features such as resolution or use distance. In this case it would then be better to consider the *apparent size* of the screen device.

- **Input Capabilities**: The features and mechanisms that a device provides for interaction. Current Connected TVs have low input capabilities, since all the

interaction is through the remote control, while laptops, through the availability of keyboard and touchpad, have higher input capability.

- **Processing Capabilities**: The processing and memory capabilities of a device.
- **Graphic/video Capabilities**: The rendering and video decoding capabilities of a device. Note that current technologies use GPUs for both processing and graphic/video support.
- **Broadcast Capabilities**: The broadcast tuning and decoding capabilities of a device.
- **Privacy**: The level of privacy that a device provides in terms of enabling the ability to not share the content in the display with other viewers in the same physical space.
- **Mobility**: The degree to which a device can be in any location of the living room, which depends on factors such as the size/weight, the power supply or the need for an antenna.
- **Geolocation capability**: The capability of a device to be auto-located, which can boost a personalisation of the contents.
- **Connectivity**: The level of network connectivity that a device is provided with, e.g., 3G/4G/5G, Bluetooth, WiFi, etc.
- **Environmental sensors availability**: The capability of a device to obtain physical or user context values such as the orientation or ambient luminosity. Note that a camera can be considered to be a generic sensor.
- **Aspect Ratio variability**: The set of different aspect ratios provided by the device. A TV provides a fixed aspect ratio, while smartphones and tablets can be turned 90°, providing a second aspect ratio.

To generalize, we define the set of device properties as:

$$P^D = \{q_1, q_2, ..., q_{N_Q}\} \tag{4.7}$$

For a given device d, a property vector represents the parametrisation of a device in terms of properties. Again, considering a simple implementation where $N_D = 3$ and q_1 = screen size, q_2 = input capabilities and q_3 = processing capabilities, a device d (e.g. a Smart TV with a large display and limited interaction and processing capabilities) could be characterised as:

$$P^D[d] = [1, 0.2, 0.2] \tag{4.8}$$

Characterizing a device by its properties can be more easily performed using typification of the devices. For example, since most current devices can be classified into smartphones, tablets, laptops and Smart TVs, we could assign the same predefined properties values to all the devices of the same type. This means that, for instance, the screen size of all the smartphones could be characterised with the same value, which is much smaller than the value for the screen size of TVs.

Formally, we define the set of device types as T^D and, similar to what has been done for components, we consider a surjective function mapping each device to its device type:

$$\mathscr{T}^{\mathscr{D}} : D \rightarrow T^D \tag{4.9}$$

and thus, we can assign the same property vector to every device belonging to the same device type:

$$P^D[\mathscr{T}^{\mathscr{D}}(d)] = P^D[d] \tag{4.10}$$

As we will describe in Section 4.5, in a web multi-device environment such as the one we are considering, web technology provides mechanisms that allow for an automatic typification of devices.

4.3.3 Layout characterisation

Today devices present a set of components to the user as efficiently as possible. The way those components are presented in the screen is called the layout. In terms of our adaptation model, a layout is a mapping of a subset of the set of components C on the screen area of a device $d \in D$.

Formally, a set of possible layouts is defined as:

$$L = \{l_1, l_2, ..., l_{N_L}\} \tag{4.11}$$

Each layout $l \in L$ can be defined in terms of a set of properties. The relevant layout properties are the following:

- **Time sharing**: At a given time t, only a subset of C_k is shown, thus each component is given a time slot of the screen resource.
- **Space sharing**: Components share the screen area, so a component should be reduced in size to fit a fraction of the screen.
- **Overlapping**: A component may overlap other components.
- **Scrolling**: A component may shift smoothly, vertically or horizontally, across the screen. Note that when scrolling, the screen area is shared in both time and space.
- **Distortion**: Components may be distorted to fit the assigned space of the screen both in scale and aspect ratio. If distortion is not provided, either the component should be cropped or an unused area is left.
- **Prioritisation of the components**: Components can be given different ranks, prioritising some over the others, which can be performed spatially (assigning more screen size) or temporally (assigning longer time slots).

Moreover, a set of specific layout parameters are also relevant for a layout selection, including the minimum size of a component, maximum number of components to show at a time, etc. Note that these parameters are related to some device properties (e.g., screen size and resolution), component properties (e.g., attention) and user context (e.g., vision capabilities), among many others.

Again, to provide generalization, we define a formal set of properties that characterises a layout:

$$P^L = \{r_1, r_2, ..., r_{N_R}\} \tag{4.12}$$

For a given number of components to show in a device, the variety of ways to organise them in the screen is very extensive. When searching for a layout solution to allocate n components in m possible locations, the number of possible layouts, N_L, is [Sears93]:

$$N_L = \binom{m}{n} n! = \frac{m!}{n!(m-n)!} n! \tag{4.13}$$

Managing such a number of combinations can be prohibitive when using devices with limited computational resources, as is the case of our target environments. To cope with this problem, we will again use typification. Specifically, we generate a bounded set of layouts on the basis of a reduced collection of layout types or templates. Examples of generic, well-known, user-accepted layout types or templates are Picture-in-Picture, Split and Carousel. Note that a layout template can be characterized by the given set of layout properties. A detailed description of the aforementioned templates, which we use in our implementations, will be provided in Section 4.5.

In general, a layout template can be defined as a spatio-temporal scheme containing shapes that represent placeholders to be replaced by components at that location of the screen. The set of available layout templates is denoted as T^L and a function maps the layouts onto layout templates:

$$\mathscr{T}^{\mathscr{L}} : L \rightarrow T^L \tag{4.14}$$

4.4 Adaptation Model

Once the elements involved in the multi-device user interface adaptation are identified, an adaptation model can be presented. As mentioned in Section 4.2 our aim is to provide a flexible, modular adaptation model for which efficient two-step implementations are possible.

In an adaptation process, the different properties of components, devices and layouts are combined to distribute the components among the connected devices in each moment, and, for each device, a layout for the assigned components is generated.

Our proposal is based on the two partial goals defined in Section 4.2, which will be independently reached through the assignation and the representation steps. Formally,

the adaptation $\mathscr{H}$ can be defined as a composition of two component sub-functions *assignation* (denoted $\mathscr{H}_1$) and *representation* (denoted $\mathscr{H}_2$):

$$\mathscr{H} = \mathscr{H}_1 \circ \mathscr{H}_2 \tag{4.15}$$

To properly define $\mathscr{H}_1$ and $\mathscr{H}_2$, in addition to the aforementioned primitive UI elements (i.e., components, devices and layouts) we introduce two new, interrelated elements.

The first of these elements is the *assignment*. Let A be the set of all possible combinations of the distribution of elements of C on D. An assignment, $a \in A$, is represented as:

$$a = \{(d_1, C_1), (d_2, C_2), \ldots, (d_{N_D}, C_{N_D})\} \tag{4.16}$$

where $C_i \subseteq C$ represents a subset of the system components. As an example, the assignment shown in Figure 4.1 would be $\{(d_1, \{c1, c7\}), (d_2, \{c2, c3, c4, c5\}), (d_3, \{c6\})\}$.

The second intermediate element is a set of layout N_D-tuples, denoted as L^{N_D}, which represent the layouts applied to every device in an assignment. Thus, a layout tuple $l^a \in L^{N_D}$ for an assignment would be:

$$l^a = (l_1, l_2, \ldots l_{N_D}) \tag{4.17}$$

On the basis of these elements, an adaptation function can be formally defined as:

$$\mathscr{H} : C \times D \times A \times L \rightarrow (A, L^{N_D}) \tag{4.18}$$

Regarding the assignation and representation functions, $\mathscr{H}_1$ makes use of the component and device set, C and D, to produce an assignment:

$$\mathscr{H}_1 : C \times D \rightarrow A \tag{4.19}$$

while $\mathscr{H}_2$ outputs a layout tuple to represent the elements of an assignment:

$$\mathscr{H}_2 : A \times L \rightarrow L^{N_D} \tag{4.20}$$

In the following we will describe in more detail assignation and representation.

4.4.1 Assignation

The objective of the assignation is to define which component to show in which of the connected devices at each moment. For that, an assignation function is defined on the basis of the *affinities* between devices and component types.

The affinity between a component and a device is defined in terms of the component and device properties. For each property pair (p_i, q_j) an affinity value is assigned. For example, the component property *attention* would have a high affinity with the *screen size*, and a low affinity with the *interaction capability* of the device. Instead, a high affinity should be defined between the component property *interaction* and the device property *interaction capability*. For a given application environment, affinities can be established, e.g., upon usability tests.

Affinity values are represented by an *affinity matrix* $A_f = \{N_P \times N_D\}$. Then, the affinity of a component c with a device d could simply be calculated as the product of vector properties and the matrix A_f. For notation convenience, we define an affinity function $\mathscr{A}$ that can be used to obtain the affinity of a component c with a device d:

$$\mathscr{A}(c,d) = P^C[c] \times A_f \times P^D[d] \tag{4.21}$$

resulting in a scalar value that represents the affinity between component c and device d.

As an example, consider three devices, *tv*, *smartphone* and *laptop*, as candidates to hold a component (main programme, denoted M). We will consider three component properties: attention, interactivity and processing requirements, and three device properties: screen size, input capabilities and processing capabilities. We now estimate the requirements vector for M, i.e., the property values for this component, as:

$$P^C(M) = [1, 0, 0.2] \tag{4.22}$$

which means that our main programme requires full attention and some processing (such as for rendering HD video), but it does not have any interaction requirements. Additionally, we estimate the screen size, the input capabilities and the processing

resources for each device:

$$\begin{aligned} P^D[tv] &= [1, 0.2, 0.2] \\ P^D[smartphone] &= [0.2, 0.7, 0.7] \\ P^D[laptop] &= [0.7, 1, 1] \end{aligned} \tag{4.23}$$

Finally, the affinity matrix could be the following:

$$A = \begin{bmatrix} 1 & 0.1 & 0.2 \\ 0 & 0.6 & 0.1 \\ 0.2 & 0.3 & 0.7 \end{bmatrix} \tag{4.24}$$

Note that the values chosen for the elements of A, although intuitive, are used only to provide an illustrative example.

To decide an assignment, the values from Eq. 4.22, 4.23, 4.24 can be plugged into Eq. 4.21, obtaining the following scalar affinity values:

$$\begin{aligned} \mathscr{A}(M, tv) &= 1.14 \\ \mathscr{A}(M, smartphone) &= 0.558 \\ \mathscr{A}(M, laptop) &= 1.02 \end{aligned} \tag{4.25}$$

which shows that the TV is the device with the highest affinity with the main programme component.

Despite the fact that $\mathscr{A}$ provides a simple order relation for the assignation function, in practice (as we will discuss in Section 4.5) a multidimensional comparison could be preferable. This comparison can be computed explicitly using the second multiplication of Eq. 4.21:

$$A_f \times P^D[d] \tag{4.26}$$

This equation calculates, for each device, a vector whose elements can be directly mapped to the component's property vector. Following the example from before and applying Eq. 4.26, we obtain the following values:

$$\begin{aligned} A_f \times P^D[tv] &= [1.06, 0.14, 0.4] \\ A_f \times P^D[smartphone] &= [0.41, 0.49, 0.74] \\ A_f \times P^D[laptop] &= [0.8, 0.7, 1.1] \end{aligned} \tag{4.27}$$

The affinity between a component and a device type can then be inspected visually, as shown in Fig. 4.2. The plot shows the properties for the main component M (in blue) and the vectors obtained from Eq. 4.27. From the figure it can be appreciated that in the example the TV appears to be the most suitable device for the main programme component, since differently to the other devices, the curve representing the TV (in red) contains the one representing the main programme (in blue). Using different values for the affinity matrix would have changed the shapes of the curves representing the different devices. By changing the affinity matrix depending on the use case, different criteria could then be applied to decide the best assignation. A more detailed example will be shown in Section 4.5.

4.4.2 **Representation**

The objective now is to establish how components are organised in the devices' screens for a given assignment. Therefore, the representation function $\mathscr{H}_2$ will be applied for each device in D. In general, the problem of layout optimisation is NP-Hard [Drira et al.07] [Singh and Sharma06]. To (partially) circumvent this complexity, in Section 4.3.3 we have introduced layout typification to consider a restricted set of layout templates. Nevertheless, a number of variables remain, mainly:

- Layout template
- Component types
- Component properties
- Device type
- Device properties
- Number of components to be shown

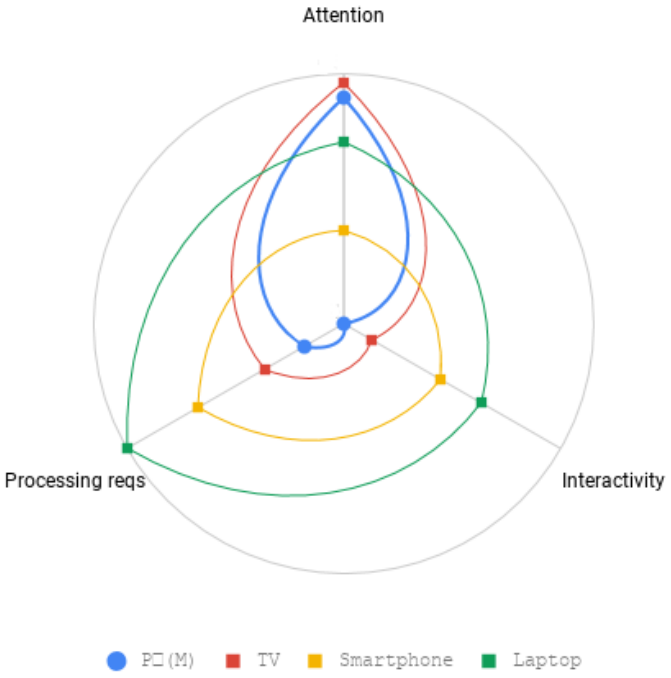


Figure 4.2: A visual representation of the affinity between component M and three device types.

- Criteria for the quality of the layout

Many other parameters could be added (aesthetics, user and physical context conditions...) which for simplicity are not considered here. Furthermore, the typification of components and devices will remove the component and device properties from the list.

All these simplifications result in a model for layout generation that will be applicable in environments with limited devices and where responsiveness is a requirement. The model of a layout generation includes the following elements:

- a set of components $C_i \subset C$.
- a device $d_i \in D$ of type $\mathcal{T}^{\mathcal{D}}(d_i)$ where the layout is generated.
- a set of layout templates, $T^L = \{t_1, t_2, \ldots, t_{N_T}\}$.
- a set of layout quality (and efficiency) criteria, to be defined next.

From a formal perspective, our approach consists of defining a set of abstract quality criteria, Z, with associated quality expressions ρ_k for all criterion k in Z, and a general expression that combines all the quality expressions. In section 4.5.7 we will describe the list of particular criteria used for evaluating the layout model.

The quality of a layout of C_i on d_i for a layout l presents the following general expression:

$$\beta(l, d_i, C_i) = \prod_k \alpha_k^{w_k} \tag{4.28}$$

where w_k is an empirical coefficient that describes the relevance of criterion $k \in Z$ in the global quality of a layout, and α_k is the outcome of the specific function that evaluates the criterion k:

$$\alpha_k = \rho_k(l, d_i, C_i) \tag{4.29}$$

The goal now is to determine the layout template that maximises the quality function. To simplify this, we will assume that for a given template $t \in T^L$ there is a layout l^*, such that $\mathcal{T}^{\mathcal{L}}(l^*) = t$, which is optimal in t. In other words, we leave the task of finding, for a given template, the best possible layout of a set of components on a device

to the implementation. As we will show, this assumption is reasonable if the implemented quality criteria are based on simple geometric parameters, which are intuitive and usual.

Therefore, the problem of selecting a layout template for C_i on d_i can be formulated as:

$$t_j : \max_{j=1..N_{LT}} \beta(l_j^*, d_i, C_i) \tag{4.30}$$

As an example to illustrate layout selection in the model we are defining, we consider three layout templates: Carousel, PiP and Split. These templates are widely used in practice and are representative of both space- and time-sharing layouts. In the absence of an extensive user evaluation, our previous experience [Zorrilla et al.15a] [Domínguez et al.17] suggests that there are some specific criteria that could make the user prefer one template over another, such as the rate of components shown on the screen, the ability to represent each component in its entirety or the efficiency in the use of the screen area.

As we will explain in detail in Section 4.5, we have used the aforementioned criteria to calculate the layouts qualities according to Eq. 4.28. Figure 4.3 shows the resulting qualities of the three layout templates. Eq. 4.30 will decide the best layout template depending on the number of components to be shown. From the picture it is clear that, using the criteria above, PiP is the best layout for a low number of components, but it is surpassed by Split in cases where the number of components matches the number of cells provided by the Split template. Moreover, the Carousel layout is the best layout when there are more than 7 components to show on a laptop.

4.4.3 Evaluating the adaptation quality

In this Section we present an evaluation model for the proposed two-step adaptation. In Section 4.2 we argued that the benefits of a two-step adaptation are at the cost of obtaining sub-optimal results. This is inherent to the decoupling of the goal of the adaptation into two partial goals that are optimized separately and does not guarantee an optimal global adaptation solution. This is because the optimal adaptation could result from an assignment not considered as the best in the assignation step, and thus is not evaluated in the representation step.

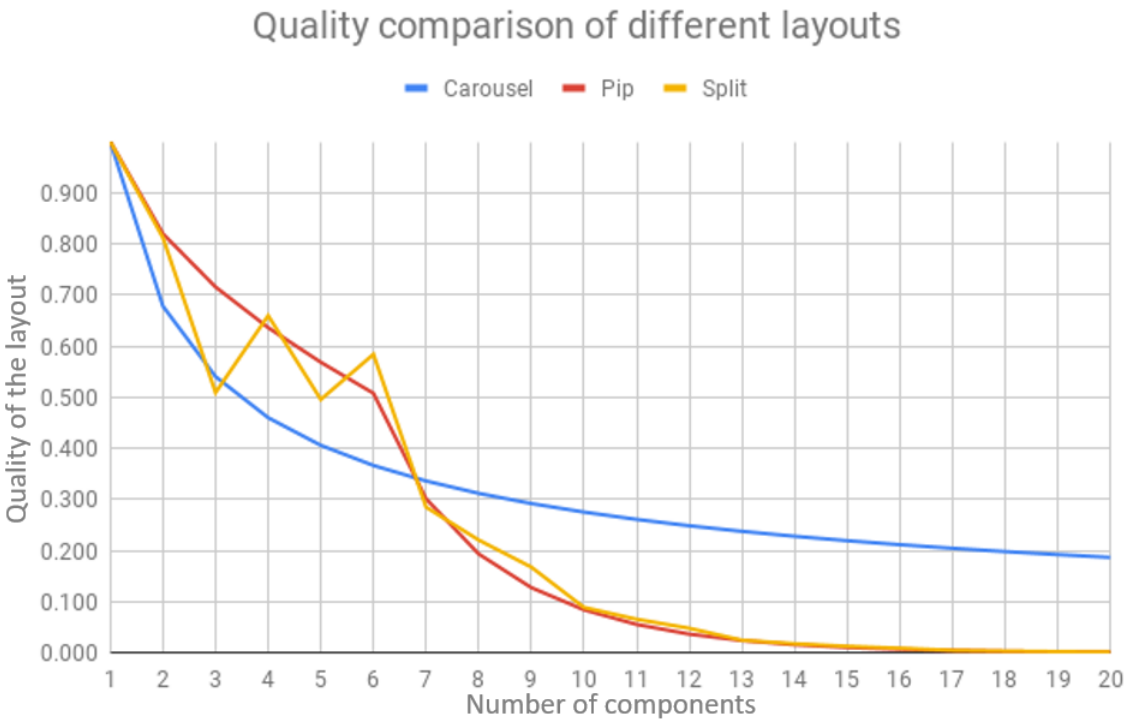


Figure 4.3: Example of qualities comparison of different layouts for a laptop

To set the quality of the solutions obtained by a two-step approach, we will consider the global optimal adaptation solution as a reference. Note that exploring all the solutions space in search of the global optimal solution may not be feasible as an implementation option for an adaptation method in a system with limited devices and real-time constrains such as ours. In Section 4.6 we will provide a quantification of the computational costs to illustrate this.

The approach for our evaluation model is based on ranking the two-step results in relation to the global optimal solutions, according to a quality metric. This metric is computed as the mean of the individual quality of the adaptation for every component in C. Note that the metric involves two criteria that match with the defined partial adaptation goals:

1. The quality of the assignment of the component, i.e., the affinity of the component with the assigned device.

2. The quality of the representation of the component on the assigned device screen for the layout selected.

Let $\mathscr{E}$ be a function that evaluates the quality of the adaptation for a single component $c \in C$ given an assignment $a = \{(d_1, C_1), (d_2, C_2), \ldots, (d_{N_D}, C_{N_D})\}$. Let $l_i \in L^a$ be the layout applied to element $a_i = (d_i, C_i)$ in a. We evaluate the overall adaptation quality of c as the product of the assignment quality and the representation quality. In general:

$$\mathscr{E}(c, a_i) = \mathscr{E}_1(c, d_i) \cdot \mathscr{E}_2(c, a_i, l_i) \tag{4.31}$$

The point now is to provide an estimate for both $\mathscr{E}_1$ and $\mathscr{E}_2$.

4.4.3.1 **Evaluating component assignment**

In Section 4.4.1 we argued about the multi-objective nature of the assignment problem. Since comparing multi-dimensional figures is cumbersome, for our evaluation model we apply the scalar expression of the affinity calculated by function $\mathscr{A}$ according to Eq. 4.21. Note that this is a worst-case estimate for the real assignment performed by the assignation step, since introducing constrains in the assignment, as represented in the example of Figure 4.2, could prevent choosing the best ranked assignment according to Eq. 4.21.

In summary,

$$\mathscr{E}_1(c,d_i) = \begin{cases} \mathscr{A}(c,d_i) & \text{if } c \in C_i \\ 0 & \text{otherwise} \end{cases} \tag{4.32}$$

4.4.3.2 **Evaluating component representation**

Eq. 4.28 defines a function β that outputs the quality of the representation of an assignment $a_i = (d_i, C_i)$ using a layout l_i. As we have shown in Section 4.4.2, quality is measured in terms of the set of criteria on the basis of geometric parameters. Because for a layout the entire set of components C_i should be considered, to obtain an estimation of the quality of the representation of an individual component $c \in C_i$, we will take Eq. 4.28 and assume that every component in C_i has the same impact on the quality of the representation of C_i, i.e.:

$$\mathscr{E}_2(c,a_i,l_i) = \frac{\beta(l_i^*,d_i,C_i)}{\mid C_i \mid} \tag{4.33}$$

4.4.3.3 **Calculating the overall quality**

Combining Eq. 4.31, 4.32, and 4.33, we determine that the quality of adaptation of a component $c \in C$ for an assignment $a_i = (d_i, C_i)$ to be represented by a layout l_i results in the following:

$$\mathscr{E}(c,a_i) = \begin{cases} \frac{\mathscr{A}(c,d_i)\beta(l_i^*,d_i,C_i)}{|C_i|} & \text{if } c \in C_i \\ 0 & \text{otherwise} \end{cases} \tag{4.34}$$

Eq. 4.34 can be applied to every component $c \in C$ for any adaptation solution, i.e., whatever combination of assignments and possible layouts. The overall quality of an adaptation $\mathscr{H} = (C, D, a, L^a)$ can be estimated by the addition of the individual estimations calculated by Eq. 4.34. The resulting expression is:

$$\mathscr{E}^{\mathscr{H}}(C,D,a,L^a) = \frac{1}{N_D}\sum_{i=1}^{N_D}\Big[\frac{\beta(l_i^*,d_i,C_i)}{\mid C_i \mid}\sum_{\forall c \in C_i}\mathscr{A}(c,d_i)\Big] \tag{4.35}$$

Observe that, for simplicity, in Eq. 4.35 we are assuming that $c \in a_i$.

4.5 Implementation

In Section 4.3 and Section 4.4 we described a model for multi-device adaptation in hybrid broadcast. The model is aimed at being general enough to allow for specific implementations, including particular sets of components, current and forthcoming devices, or alternative layout templates. On the other hand, the outputs in the adaptation strongly depend on a set of parameter values that should be configured upon the use of the system in a real deployment, either by manual setting or by an automatic learning process.

The aim of this section is to provide a comprehensive description of how an implementation can be derived from the model.

For this, we describe an implementation of the model based on a set of categorized components, devices and layouts that we consider as representative and reasonably complete for current hybrid broadcast broadband services.

Furthermore, a reduced subset of the defined element properties has been implemented for this prototype, which can be easily extended as we will show in Section 4.6. As already mentioned in Section 4.4.1, the parameters have been chosen to intuitively make sense. Obtaining them using a machine learning approach or by conducting a user evaluation, although it would be convenient when deploying a real application, is outside the scope of this paper. As described in Section 4.2, for efficiency reasons, we apply our adaptation model to follow a two-step implementation. Step 1 is in charge of selecting an optimal assignment of components to devices in terms of affinity criteria. Step 2 is executed in each device to find an optimal layout for the assignment of components to the device. Note again that in Step 2, the layouts evaluation is conditioned by the previous assignment decision in Step 1, hence a global optimal adaptation solution could not be reached by this two-step approach. Instead, a reasonably good solution is acceptable for our purposes, while the two-step approach brings better responsiveness and less energy consumption. Furthermore, considering the dynamic behaviour of a multi-device context, the system could apply small corrections whenever the context changes, providing agile, incremental assignations.

In the remainder of this section, we first describe the specific component and device types we have considered, as well as the layout templates used. Then, we describe

the implementations we have followed for Step 1 and Step 2 of the adaptation process regarding the specific scenarios we have considered, the solutions provided, and trade-offs taken. Finally, we describe a mechanism for triggering the adaptation process with the aim of preserving an acceptable adaptation quality upon context changes.

4.5.1 Component types and properties

We have conducted a research [Dominguez et al.18] to provide component typification. The work proposes a way of componentising hybrid broadcast-Internet media services, based on a methodology that analyses the contents of different TV programmes and groups them in terms of the component properties we have defined in Section 4.3.1. The following subsections describe the mentioned work the focusing on:

- Methodology description
- Componentization of a TV programme

4.5.1.1 Methodology description

Regarding the selection of the broadcast emissions to analyse, the criteria has been to bound the amount of emissions to those that are watched by the highest audience volume. That way, the contents obtained as a result are contents that the audience is used to find at present, having the least impact on the experience and contributing to the usability of the final adaptation model. Taking this into account, the selection of the emissions has been done as follows:

1. **Selection of the 5 countries in European Union with the highest population**, being Germany, France, United Kingdom, Italy and Spain the countries fulfilling that condition.
2. **Selection of the most viewed TV channel from each one of the previous countries.** [stab, staa, stae, stac, stad] show that ZDF, TF1, BBC, RAI 1 and Telecinco are the most watched channels in each country in 2016.
3. **Selection of the most appropriate programmes for Social iTV**. Some programmes are more suitable for interactive experiences [Geerts et al.08] due to

the large amount of content they show or the motivation they generate for the user to share his experience in social networks. From a list of 18 programme genres based on the EBU [ebu] classification, the most suitable genres for synchronous social iTV are: News, soap, quiz, sports, reality show and talk show. The suitability is evaluated depending on how much people talk and share during and after the TV show. All of them have been initially considered but soap and reality show have been finally discarded since soap are not designed to be componentized and reality shows are focused on other type of social interaction, different from what we are looking for. Therefore, the four programme types that have been analysed could be described as follows:

- **News:** Daily scheduled programme that reports current events. News is reported as a series of individual stories and can include a variety of contents.
- **Quiz:** Programme in which a single or a group of contestants compete in answering questions.
- **Sports:** Programme in which a sport competition is broadcasted, preferably live.
- **Talk show:** Programme in which a panel of guests discuss various topics such as politics, celebrities personal life or news summaries.

4. **Selection of one emission per programme for each type and country.** One emmission per programme is considered to be enough since all emissions of the same programme follow more or less the same structure. Emissions from 2016 or 2017 have been searched online in order to ensure contemporary content.

5. **Selection of the first 30 min from each emission**, which experimentally has been verified to be enough to identify the type of elements that could appear on it.

6. **Check the impact of advertisements** in both public and private channels. The previous programmes selection includes both public and private channels, which will allow to detect the mentioned impact.

Table 4.1 shows the summary of the analysed emissions per country, channel and programme type.

Table 4.1: Summary of the selected emissions

Country	Channel	Type	Programme
Germany	ZDF	News	Heute journal
		Quiz	Der quiz champion
		Sports	Ski
		Talk show	Heute show
France	TF1	News	Le journal de 13h
		Quiz	Money drop
		Sports	Football
		Talk show	24 heures en question
UK	BBC	News	News at 10
		Quiz	Impossible
		Sports	Golf
		Talk show	Sundays politics
Italy	RAI1	News	Telegiornale TG1
		Quiz	L'eredità
		Sports	Cycling
		Talk show	Agorà
Spain	Telecinco	News	Informativos Telecinco
		Quiz	Pasapalabra
		Sports	MotoGP
		Talk show	El programa de Ana Rosa

4.5.1.2 **Componentization of a TV programme**

As a result large list of different elements have been identified for each programme type. All of those elements are included in Table 4.2 in an extensive way and the most representative can be found in Figure 4.4. As can be seen, the four type of programmes show a variety of elements along the emission. Pictures 1 to 4 in Figure 4.4 show different video signals for live connections, participant cameras or videos sent by the users while Pictures 5 to 12 in Figure 4.4 show other elements appearing along the emission such as advertisements, headlines, text, data, questions, classifications, or twitter content. Some of these elements in Table 4.2 and Figure 4.4 exhibit certain degree of similarities that suggest the componentization of a TV programme.

Following this criteria, 8 components types have been identified to be used when the TV programme evolves into an interactive multi-device TV show:

Figure 4.4: Examples of different elements found in the analysed emissions

Table 4.2: Summary of the identified elements in the analysed emissions

News	Quiz	Sports	Talk show
• Main video signal • Live connection • Pre-recorded videos • Interviews • UGC • Programme name • Presenter name • Journalist name • Name of the interviewed person • Headlines • Live, place • On-screen graphic • Weather • Maps • Statistics • Text • Twitter content (A tweet overlapping a Twitter page)	• Main video signal • More than one contestant at the same time • Contestant presentation video • Contestant info: name, job, place • Time • Attempts • Scoreboard • On-screen graphic • Questions (different subjects, musical, fill in the gap, photos, etc) • Answers (true/-false, 4 options, fill in the gap) • Info for the casting • Prizes (for the contestant, for the audience) • Audience participation (answer by sms, hashtags to follow) • Adverts (other programmes, first releases)	• Main video signal • Different angles, athletes or race points, on board, etc • Interviews • Replays: plays, overtakes, laps, etc • On-screen graphic • Small detail classifications (fastest lap, distances, scoreboard, starting line) • Extended classifications • Tournament info • Schedule • Circuit, stage information: name, curves, kms, records, mountains, etc • Athletes, coach, teams info • Weather • Previous results • Performance info(speed, distance, time) • Audience participation (raffles by call or sms, hashtags to follow, WebSite) • Adverts (products, brands)	• Main video signal • News • Interviews • Live connections • Debate participants • UGC • On-screen graphic • Headlines • Name of the person talking • Info of the person talking • Breaking news • Time • Politician's cites • Newspapers headlines • Statistics • Graphs • Text • Audience participation (hastags to follow, answers by call or sms, answers by Facebook or Twitter) • Adverts (other programmes)

1. **Main programme**: The mainstream audiovisual content that drives the experience.

2. **Advertisements**: Business-related resources advertising something within the TV programme.

3. **Secondary videos**: Additional videos sourced by the broadcaster.

4. **Banners**: Additional information including notifications, headlines and small texts.

5. **Static Data**: Elements that show relevant data, longer-texts or images, that might not bring interaction by the user.

6. **Interactive Data**: Elements that show relevant data, diagrams or tables that might bring interaction by the user.

7. **UGC - User Generated Content**: Additional videos generated by the viewers.

8. **Social content**: Additional information coming from the opinion of the viewers (social networks, a quiz, etc.).

Our implementations are based on this set of component types, or a subset of it. Every component $c \in C$ in the multi-device experience is labelled with one of the aforementioned types, which provides a straightforward implementation of function $\mathscr{T}^{\mathscr{C}}$. Then, assigning property values to c can be systematically made from predefined values that are mapped to component types. For this prototype implementation, we will use three of the component properties defined in Section 4.3.1: attention, interactivity and processing requirements, following the example initiated in Section 4.4.1.

4.5.2 Device types and properties

In the context of hybrid broadcast-Internet media services, such as the one described in [Domínguez et al.17], there are usually four types of devices involved, also called *core devices* or *core screens* [Nagel15]. We are talking about smartphones, laptops, tablets and Smart TVs. These devices will be the basis for our implementation, even though

the model we have defined is able to incorporate other types of devices, such as smartwatches, VR/AR headsets or future incoming HCI devices.

Device properties will be generically assigned to the considered device types. Most of the properties can be theoretically obtained from the scripting capabilities of Web browsers, frameworks or libraries. However, this is sometimes not possible and often provides false positives, making such information unreliable. Finally, device typification must be performed at run-time.

In this context, we conducted a research in which we used the information about the device made available by the browser to implement device typification. The work is described in [Dominguez et al.19] compares three different methods of Web-based device type detection using the User Agent of the browser and basing their knowledge on different learning systems and statistical models. The following subsections describe the mentioned work focusing on:

- Existing tools for Web-based device type detection
- Device type detection methods
- Empirical comparison

4.5.2.1 **Existing tools for Web-based device type detection**

There are numerous open source and proprietary libraries or scripts that detect specific features and the type of devices. In this section a description of the most relevant ones is provided including an accuracy test of the device type detection open source libraries.

Regarding the detection of specific features of a device, Modernizr [modb] finds which HTML, CSS and Javascript features the browser has to offer. Feature.js [fea] is a lightweight browser feature detection library that determines if a code can be executed in the browser. These libraries provide the available features based on the result of browsers' API calls and while they work well for the detection of some specific features, in some cases they do not provide the required functionalities. On the one hand, browsers' APIs sometimes provide false positives on certain features [fal]. For instance, in a laptop the browser could have an interface for geolocation, but most of the times this capability will not be available since the laptop is regularly used in inner spaces. In this case, knowing the device type could be helpful to infer the final result. On the

other hand, there are characteristics that are related to usage patterns and cannot be obtained through the browser, such as the field of view of the user when using the device or the degree of privacy of a device. Here again, knowing the device type can be helpful to include this type of information in the adaptation process, where the device type could comprehend a set of such characteristics.

Considering the device type detection, Mobile-detect.js [mob] detects if a device is mobile or not and, if so, whether phone or tablet. Detectizr.js [det] is a Modernizr extension to detect device type, device model, screen size, operating system or browser details. Categorizr.js [cat] is defined as a device detection script that takes the User Agent string and categorises the device into desktop, TV, tablet or mobile. UAParser.js [uap] also detects the device type and distinguishes between console, mobile, tablet, smart-TV, wearable or embedded devices.

Amongst these libraries, the most used is UAParser.js, with over 2.5 millions weekly downloads. Mobile-detect.js is also quite popular, with more than 200 millions hits per month, according to jsDelivr, an open-source CDN for npm packages [jsd]. The other libraries are far less popular, with Detectizer.js reporting about 200 thousands hits per month while Categorizer.js being listed as deprecated and without active development since 2015.

Since none of the existing libraries provide details about their accuracy retrieving device features, a dataset [uag] of 3228 User Agent strings has been assembled and for each method described above we have computed the accuracy in detecting the device type. The User Agent strings in the dataset are categorized into four classes:*desktop*, *mobile*, *tablet* or *TV*. The distribution of the devices in the dataset is shown in Table 4.3, while the results of the evaluation are available in Table 4.4. The value N/A is used in the table in case the library does not detect that particular class of devices. Strangely enough, the best results are achieved by Categorizer.js (a deprecated library) with an accuracy of 91.8%. Since Categorizer.js performs the detection using a rule-based algorithm, with the release of newer devices its accuracy is destined to decline over time, since the library is not actively developed anymore. The two most used libraries offer disappointing results: Mobile-detect.js, even though it is designed specifically for mobile or tablet devices, detects the correct device class only 63% of the time. UAParser.js can detect TVs besides mobiles and tablets, but it performs even worse than Mobile-detect.js, with just 61.8% correct detections. Detectizr.js is the library with the

Table 4.3: Independent dataset distribution by device type

Device type	Number of UA in dataset
Mobile	1000
Tablet	1000
Desktop	1000
TV	228
Total	3228

worst performance since it detects the device type correctly only 31.8% of the times. Its detection algorithm is extremely skewed towards the detection of mobile devices, in fact the library reaches a 100% detection of mobile devices, but the recall measure for this specific class is 0.32 (with 0.31 being the theoretical minimum given for a 100% precision on the given dataset).

All these tools compare the user agent against a database or a token list of specific terms and return a device type. A User Agent is a string identifying the browser and operating system to the web server. Table 4.5 shows an example of User Agent for each device type. As can be seen, the contents of the User Agent field vary from browser to browser and although it contains tokens which provide valuable information about the browser and device, a standardized User Agent format does not exist. Moreover, as the number of Web-enabled devices increases, the variety of different User Agent strings grows. Therefore, methods as the aforementioned require updated databases and token lists for identifying correctly new devices, which is a very high maintenance task doomed to fail at some point in the future. Consequently, more efficient and advanced methods, which require less upkeep, are needed.

In the next section we propose three device type detection methods based on the analysis of the User Agent string, but instead of doing a simple comparison against a database, they use more advanced methods that can detect a device even if the given User Agent does not exist in the database.

4.5.2.2 **Device type detection methods**

Since the goal is to obtain the device type from the analysis of the User Agent string, different techniques for text classification have been considered. This includes algebraic models, classical statistical models and machine learning models. In order to carry out

Table 4.4: Accuracy of existing device type detection libraries

Method	Device	Dev. Accuracy	Total accuracy
Categorizr.js	Mobile	0.903	0.918
	Tablet	0.988	
	Desktop	0.997	
	TV	0.332	
UA-parser.js	Mobile	0.835	0.618
	Tablet	0.927	
	Desktop	N/A	
	TV	0.506	
Detectizr.js	Mobile	1	0.318
	Tablet	0.007	
	Desktop	0	
	TV	0.083	
Mobile-detect.js	Mobile	0.748	0.632
	Tablet	0.516	
	Desktop	N/A	
	TV	N/A	

Table 4.5: Examples of User Agent strings

Device type	User Agent example
Mobile	"Mozilla/5.0 (Linux; Android 7.1.1; Moto G (5S) Plus Build/NPSS26.116-64-8) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.126 Mobile Safari/537.36"
Tablet	"Mozilla/5.0 (Linux; U; Android 2.2; en-gb; GT-P1000 Build/FROYO) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"
Desktop	"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36"
TV	"Mozilla/5.0 (Linux; Tizen 2.3; SmartHub; SMART-TV; SmartTV; U; Maple2012) AppleWebKit/538.1+ (KHTML, like Gecko) TV Safari/538.1+"

an empirical comparison of different methods, both in accuracy and performance, one method from each group has been selected:

1. Vector space model amongst algebraic models
2. Logistic regression amongst classical statistical models
3. Neural network amongst machine learning models

First, the vector space model was selected due to its simplicity as the baseline model for user agent based device classification. Logistic regression is a standard method used for text classification because it can be trained extremely fast, its results are easy to interpret and are often very good. Finally, a neural network model was chosen because of the recent success of deep learning methods in the area of text processing. The main idea is that a neural network, having the ability to analyze complex manifolds in multidimensional spaces, could represent an improvement over a linear model (such as logistic regression) in case the distribution of the input data is strongly non-linear. On the other hand, a neural network does not offer an explanation of the output.

Therefore, each approach uses a different method to output a device type class label when a User Agent string is given as an input. The starting point for the implementation of the three device type detection methods has been the publicly available *Browscap* dataset [bro], which will be referred as the initial dataset. This dataset has been processed in different ways in order to implement each mentioned method.

4.5.2.2.1 Vector space model

This algebraic method is based on the vector space retrieval mode which consists on performing comparison and retrieving objects that are likely to satisfy the query of the user. In the statistically based vector-space model, a document or dataset entry is conceptually represented by a vector of keywords extracted from the document, with associated weights representing the importance of the keywords in the document and within the whole document collection [Lee et al.97].

The weight of a term in a document can be determined by the Term Frequency - Inverse Document Frequency (TF-IDF) method in which the weight of a term is determined by two factors: the frequency of the term t in the document i ($tf_{i,t}$) and the

frequency of the term t in the whole document collection (df_t). The weight of a term t in document i is defined as:

$$w_{i,t} = tf_{i,t} \times idf_t = tf_t \times log(\frac{N}{df_t}) \tag{4.36}$$

where N is the number of documents in the dataset and idf_t is the inverse document frequency, a global parameter that gives higher weight to the terms with lower occurrence in the whole dataset. Therefore, the TF-IDF method accentuates terms that are frequent in the document, but not frequent in general in the dataset.

The same method is applied to model the query as a list of keywords with associated weights. Once the dataset terms and the query terms are ranked, the cosine similarity function is calculated for each document in the dataset, which returns the angle between the query Q and the document D_i.

$$similarity(D_i, Q) = cos(D_i, Q) = \frac{\sum_{j=1}^{V} w_{i,j} \times w_{Q,j}}{\sqrt{\sum_{j=1}^{V} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{V} w_{Q,j}^2}} \tag{4.37}$$

The smaller angle returns the most likely result.

In order to build the first device type detection method the initial dataset has been tokenized and a different relevance has been provided to each word of each User Agent string following the TF-IDF method. Again, the same method is applied when a new User Agent is given to obtain a device type. Once the dataset terms and the query terms are ranked, the cosine similarity function is calculated and the system returns the device type of the User Agent that provides the smaller angle. The code of the implementation can be found in [vsm].

The simplicity of the vector space model is one of the advantages of this first device detection method. On the other hand, the results obtained with this method will not be the best ones because of its well-known limitations (for instance, the lower classification accuracy obtained on longer documents). The performance of this method is also problematic: as it is not related to a learning method, the accuracy of the detection strongly depends on the size of the dataset used to create the model, but the time required to perform the detection grows in direct proportion to the size of the dataset.

4.5.2.2.2 Logistic regression

This second method is based on a logistic regression classification algorithm which is a linear model for classification. Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary), but it can easily be extended to the multinomial case. This model takes real-valued inputs and produces as output the probability of the input belonging to a class. Its behaviour is described by the following expressions [Hosmer Jr et al.13]:

$$t = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n + \epsilon \tag{4.38}$$

$$P(X) = \frac{e^t}{e^t + 1} \tag{4.39}$$

where $P(X)$ is the probability of having the event of interest, x_i are the independent variables, α_i are the associated weights to each independent variable which store the knowledge and ϵ represents the error term. Eq 4.39 is known as the standard logistic function. It is a function which takes a set of real values as input and transform them so that they are strictly positive and their sum equals one.

For the case of device type detection, the initial dataset has been used in order to train the model and obtain the values for $\alpha_0, ..., \alpha_n$ which represent the parameters of the method.

When a prediction is performed, the words in the dataset represent the x_i and are evaluated as binary variables: 1 if a word appears in the User Agent string and 0 if not. Thus, given the binary array $[x_1, ..., x_n]$, a set of four Eq 4.38 expressions (one per device: TV, desktop, tablet, mobile) returns the predicted result t for each device type, that evaluated through Eq 4.39 provides the probability of the given User Agent belonging to a device type. The highest probability of the four defines the detected device type. The code of the implementation can be found in [log].

This method is supposed to provide more accurate results than the vector space model method due to its ability to learn the weights α_i that best separate the device type classes. Another valuable advantage of this method is the low computational cost of the training process.

4.5.2.2.3 Neural network

The recent revolution of deep learning with its remarkable results in many different fields led us to choose a neural network model as the third method for the target device type detection system.

Neural networks are computational models based on ensembles of artificial neurons structured in layers stacked on top of each other [Goodfellow et al.16]. Hence, input information crosses the neural network where it is subjected to numerous transformations in order to produce the output values. This input-to-target mapping is achieved after a training process in which data transformation rules are learned by exposure to examples. Repeating the training process over several epochs, the network will adjust the weights of the "axons" (the links between neurons) and improve its accuracy [Werbos75]. However, an excess of training can lead the network to overfit, that is, it increases the accuracy on the training data at the cost of losing the ability to generalize to unseen samples.

In our attempt to obtain an accurate device type detection system, a neural network has been trained with the same initial dataset, which stores thousands of User Agents related to a device type. Figure 4.5 shows a graphical representation of the network architecture. The neural network has been implemented in Keras [Chollet et al.15] and the model consists of 2 blocks of fully connected layers (which use ReLU as activation layer followed by a Dropout layer to reduce overfit) and a final block consisting in a fully connected layer followed by a softmax layer which outputs the probability of the User Agent string to belong to the 4 devices considered. The loss function used in the training process is *categorical_crossentropy* and Adam [Kingma and Ba14] was selected as the optimizer. In total, the network has 516,388 trainable parameters. The network used an initial learning rate of 0.001 which automatically decreased once the validation accuracy stopped improving. The model was trained for a total of 15 epochs.

Once again, the model has been configured in such a way that the system returns a device type when a User Agent is given as input. The code of the implementation can be found in [neu].

This method is supposed to obtain the most accurate results as it is able to automatically adapt the inference rules during the training process and it is not limited to linear decision boundaries like the logistic regression method. The resulting system should

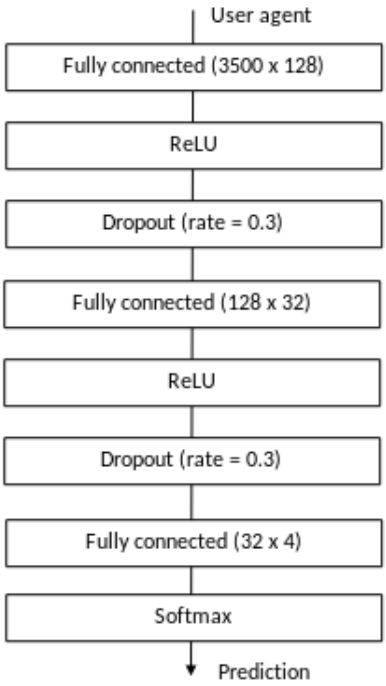


Figure 4.5: Neural network architecture

be therefore able to learn and predict the type of unknown devices. Furthermore, the prediction is rapidly calculated. The only drawback is the time and computational cost of the training process of the neural network.

4.5.2.3 **Empirical comparison**

In this section an empirical comparison of the three implemented methods is provided. As mentioned in Section 4.5.2.2, the starting point for the implementation of the three device type detection methods has been the *Browscap* [bro] dataset, which provides thousands of User Agents strings and their corresponding devices. The implementation has been focused on the four device types that are currently most important in the hybrid TV ecosystem and that can be called core devices or core screens [Nagel15], namely mobile, tablet, laptop/desktop and smart TV. Wearables and VR headsets were discarded for the analysis, since there was no data available in *Browscap*. However, if available, their detection could be easily included. Therefore, the dataset, provided in [uag], included 219,075 User Agents strings whose distribution is shown in Table 4.6.

Table 4.6: Initial dataset distribution by device type

Device type	Number of UA in dataset
Mobile	103,021
Tablet	52,942
Desktop	62,101
TV	1,011
Total	219,075

Once the methods have been implemented, the analysis consisted on checking and comparing the accuracy of the implemented methods and also on testing the performance of all of them. The accuracy of the three methods has been compared through:

- A cross validation: in order to assess the generalisation ability of each method on a part of the input User Agent dataset which was not used to build the models.
- A test with an independent dataset: in order to check how the methods will perform in real conditions. This dataset has been collected by the authors. It is the same dataset used to compute the accuracy of existing solutions as described in section 4.5.2.1 and is provided in [uag].

Obviously, the goal has been to obtain the highest possible accuracy. However, a thorough error analysis has been performed, since the impact of some misclassifications is bigger than others. For example, classifying a mobile phone as a tablet is not as big an error as classifying a mobile as a TV. Mobiles and tablets present similar features whereas TV's features are quite different. This second case would cause a worse effect when distributing the content among devices, for instance, when an element requiring text input is sent to the TV.

The performance has been tested by checking the time required by each method:

- to provide the device type result when a User Agent is given
- to train the employed model

In this case, the main goal has been to deploy a method which provides the device type as fast as possible and without affecting the quality of the user experience. Depending on the purpose of the system, the training time could be critical or not.

Table 4.7: Cross validation results

Method	Accuracy
Vector space model	0.9873
Logistic regression	0.9868
Neural network	0.9959

Finally, the potential of improvement has been analysed. The aim of the work presented in this paper has been to improve the performance of the Web tools in the state of the art through the proposal of different learning and statistical methods. However, further research could still improve the obtained accuracies and that is why a discussion on potential improvements has been added.

4.5.2.3.1 Accuracy validation

4.5.2.3.1.1 Cross validation Cross validation is a validation technique for assessing how the results of a statistical analysis will generalize to an independent dataset. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice [Bishop06]. In order to do that, initial data sets are usually split into training and testing with 80/20 proportion. Therefore, 80% of the data is used to train the system and the remaining 20% is used to perform validation.

As mentioned before, the dataset described in Table 4.6 has been used as starting point for the three methods. Those 219,075 dataset entries have been shuffled and a cross validation has been performed to check the accuracy of each method. Table 4.7 shows the obtained accuracies.

As can be seen, the three methods achieve a high accuracy and the three methods get good results, being the NN the best one with an error rate of only 0.4%, versus about 1.3% of the other two methods.

4.5.2.3.1.2 Independent dataset The three algorithms were also tested on an independent dataset, unrelated from the one used for training. This dataset is the one used to evaluate the performance of existing libraries as described in section 4.5.2.1. The aim of this second validation has been to check how the evaluated methods perform in real conditions, compare them against the state of the art and test if the high

Table 4.8: Obtained accuracy with an independent dataset

Method	Accuracy
Vector space model	0.8175
Logistic regression	0.9483
Neural network	0.9538

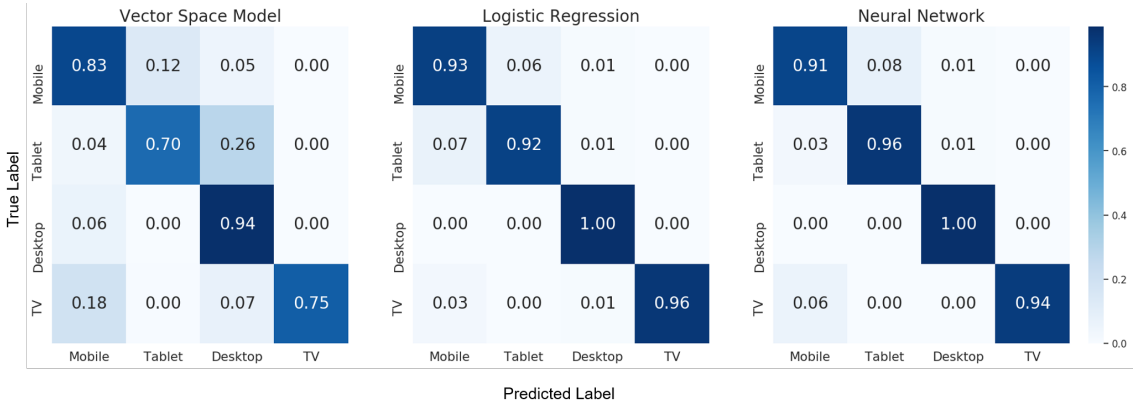


Figure 4.6: Accuracy by device and method

accuracy obtained on the cross-validation dataset is due to the homogeneity of the initial dataset or not. The distribution of the device classes in the dataset is shown in Table 4.3. Those 3228 User Agents have been provided to obtain the device type through the three methods and the obtained accuracies are shown in Table 4.8.

These results follow the expected pattern. Vector space model was expected to be the worse model as the method just compares the distances to the User Agents in the datasets. The lower accuracy in this case might be due to the validation set being more similar to the training set than the independent test set. The method which achieved the highest accuracy is the neural network, although the logistic regression model obtained a comparable accuracy. Both logistic regression and neural network achieved results above the state of the art.

Figure 4.6 shows the accuracy results by each method split by device. As can be appreciated, vector space model is the worse method regarding all devices types. Logistic regression is the best one detecting mobiles and TVs and and neural network is the best one detecting tablets. Finally, he accuracy in detecting desktop/laptop devices is close to 100% in both cases (accuracy is shown to be 1.00 because of rounding). Analysing the classification error it is clear that the hardest problem is distinguishing between mobile

and tablet devices, despite the huge amount of training data available.

In this context, if the distribution of the training dataset in Table 4.6 is analysed, it is easy to appreciate that the dataset is not balanced: the amount of user agent strings for mobile devices is 100 times bigger than the amount for TVs and this can affect the results. The previous fact has a profound impact in the accuracy of the neural network as its knowledge is achieved by exposure to examples. This, on the other hand would not explain the fact that the system can detect TVs better than mobile phones. However, if mobile related failures are analysed, in most of the cases mobiles are perceived as tablets. The reason is that User Agents of both device types present similar terms and the dense neural network is not able to find the correct patterns when combining different terms. Tablets often show the representative "Tab" term which eases the task and that is probably why a higher accuracy is obtained.

Nevertheless, the results in the logistic regression are never lower than 92% for any device class. In this case, an independent polynomial is used for each device type and therefore, the weights associated to each term of the User Agent do not have impact on the other device type probabilities. This method also improves the accuracy for the TVs as the User Agents of the TVs usually show very representative terms such as "smartTV" or "hbbtv".

4.5.2.3.2 Performance validation

The implementation of the three methods presents a different performance both in the prediction and learning processes. These performances have been compared through different tests carried out on a computer with the following specifications:

- Processor: Intel Core i5-4590 CPU @3.30GHzx4
- Memory: 8Gb
- GPU: Nvidia GTX 1080

4.5.2.3.2.1 Time to process the result of a prediction In this case, the test has consisted in calculating the time required by the system to return a device type result given a User Agent string. The test has been repeated with 25 User Agents for each approach. Table 4.9 provides the average value for each method.

Table 4.9: Average time required to process the result of a prediction (on CPU)

Method	Time average (s)	σ^2
Vector space model	0.0512	1.1818E-07
Logistic regression	0.0006	1.19E-07
Neural network	0.0007	2.52E-08

Table 4.10: Average time required to process or learn on the initial dataset (on CPU)

Method	Average time (s)	σ^2
Vector space model	4.1186	0.1683
Logistic regression	29.91243	0.3169
Neural network	212.2969	2.1452

As can be seen, vector space model method is much slower than the other ones returning a result, since it has to calculate the similarity of the query with all the User Agents in the dataset. Logistic regression and neural network present similar results. In these cases, the computational cost of the classification is much lower than in the former. All in all, the three methods could be valid when talking about response time, since the obtained values would not affect the user experience.

4.5.2.3.2.2 Time to process or learn the initial dataset As explained in Section 4.5.2.2 all the models require a previous learning or processing period related to the initial dataset. Therefore, the time required for training has been measured in order to check the computational costs of each method. The training process was repeated for 5 times and Table 4.10 shows the average results. For the analysis the neural network has been trained on CPU. However, the libraries employed to train neural network models are usually prepared to work on the GPU. When trained on the GPU, the model required around 4 seconds per epoch and was able to train in less than one minute.

As expected, the vector space model is the method that requires the least time. The reason is that it does not strictly learn, but it tokenizes the dataset and calculates the weight of each term of the User Agents in order to have a prepared dataset when a query is received.

Logistic regression is the second fastest model. Its learning process consists on updating the coefficients of a polynomial expression according to the terms of the User Agents in the dataset. It needs seven times the time needed by the vector space model, but the obtained accuracy is considerably better.

Finally, neural network method is the slowest training the model and it needs approximately seven times the time needed by the logistic regression to train the model. The training time of a neural network is very sensitive to the size of the training dataset. On the other hand, the generalization capability of the neural network depends of the size an variety of the training set. Since the accuracy results we have obtained for the neural network are similar to the results obtained for the regression method (and the neural network performs near perfect in the cross-validation test), from the results we can hypothesise that, in general, training a neural network can be about one magnitude order most costly than learning a logistic regression model. However, the training time of the neural network has been less than five minutes and, if a GPU is available, training times are comparable to those required for training the logistic regression model. As training is a one-time cost, this result does not exclude neural networks as being a candidate method for device type detection. More complex User Agent strings will increase the overhead, but it is not easy to envisage a real scalability problem.

4.5.2.3.3 Potential for improvement

There are several possibilities to improve the accuracy results obtained by the methods described in the previous section. The training of the models could be improved, considering that the dataset used for training is severely imbalanced. Applying resampling techniques like SMOTE [Chawla et al.02], or adding more samples for the less representative classes would likely improve the methods' accuracy. Even though User Agent strings are very different than text usually used for other classification tasks (e.g. sentiment analysis or language detection), it would be worth investigating approaches that exploit the input structure, for example considering bigrams or trigrams (contiguous blocks of two or three words) instead of single words, or using word embeddings instead of one-hot encoding. In the context of statistical methods, more advanced techniques such as SVM or XGBoost [Chen and Guestrin16], while requiring considerably longer training times, could lead to better classification results. The neural network approach is probably the method that can be improved the most since enhancements can be applied across several dimensions. For instance, tuning the hyperparameters of the network (number of layers, dropout rate, etc.) is a standard way of increasing the network performance. Recurrent neural network architectures such as LSTM [Gers et al.99] are

more suited for processing of text sequences as they keep memory of previous input data and thus are likely better than a standard neural network as the one presented above.

Therefore, in this part of the research we used the information about the device made available by the browser to implement device typification. Nevertheless, there are device properties that are related to usage patterns, such as the degree of privacy of a device, which cannot be obtained from the browser and have not been integrated in our implementation. For this prototype implementation, we will use three of the device properties defined in Section 4.3.2: screen size, input capabilities and processing capabilities. Since the screen size is a key property for layout generation, we will provide a specific definition for this property in Section 4.5.7.

4.5.3 Layout templates

In Section 4.3.3 we described a strategy to reduce the combinatorial possibilities in the generation of layouts. Following the usual typification approach, a reduced set of template layouts is used.

The template to be used in each specific situation will depend on parameters such as the type of device, the number of components to be shown, the nature of the application or the number of devices being used at the same time. In order to support that assertion, some user tests were carried out and published in [Zorrilla et al.15b]. The following subsections sumarize the cited work focusing on:

- Hypothesis formulated before the user tests
- User tests
- Results and conclusions

4.5.3.1 Hypothesis formulated before the user tests

Our hypothesis underlines these four parameters to affect to the User Interface:

- **The device**: As happens in a single-device application, the target device is very relevant to build a responsive Web application. In the same way, the devices involved in a multi-device application are expected also to be relevant.

- **The number of Web Components**: The quantity of pieces of information to be shown in that device can affect on how to present the content.
- **The nature of the application**: This parameter could be important to decide the arrangement of the User Interface. For instance, if there is a main video and related information on the device, or if the video is being displayed on another device and that device is being used only for extra information.
- **Other devices being used at the same time**: We want to know if having a second device being used simultaneously has an impact on how the user wants to arrange the components in the first screen.

4.5.4 User tests

To find out the influence of each parameter, we enclosed them to these options:

- **The device**: 3 different devices. A Motorola Moto G Smartphone in portrait mode, A Nexus 10 tablet in landscape mode, and a Samsung UE40C8000 TV.
- **The number of Web Components**: Showing 3 or 6 components at the same time.
- **The nature of the application**: Two different scenarios. At least one of the components is a video, or there is not a video among the components.
- **Other devices being used at the same time**: Two possible situations. The evaluated device is the only one being used by the user or there is another device as a companion screen.

Testing images have been created simulating a broadcasted live F1 race scenario with all the possible combinations of the first three parameters (see Table 4.11 from Id 1 to 12).

Apart from these 12 context situations, we created 4 more to evaluate the "Other devices being used at the same time" parameter. We defined as a second screen a TV showing two fixed components and presented different contexts in a tablet, making the user think about how to present the content in the tablet, while they were also watching related content in the TV. As an outcome we have 4 new combinations in Table 4.11 from Id 13 to 16.

Table 4.11: Combinations with the defined parameters

Id	The device	The number of web components	The nature of the application	Other devices being used at the same time
1	TV	3	At least one video	No
2	TV	6	At least one video	No
3	TV	3	No videos	No
4	TV	6	No videos	No
5	Tablet	3	At least one video	No
6	Tablet	6	At least one video	No
7	Tablet	3	No videos	No
8	Tablet	6	No videos	No
9	Smartphone	3	At least one video	No
10	Smartphone	6	At least one video	No
11	Smartphone	3	No videos	No
12	Smartphone	6	No videos	No
13	Tablet	3	At least one video	Yes, A TV
14	Tablet	6	At least one video	Yes, A TV
15	Tablet	3	No videos	Yes, A TV
16	Tablet	6	No videos	Yes, A TV

For each one of the 16 combinatorial contexts, we created always four different user interface arrangement patterns:

- **Grid Layout**: Based on the CSS Grid Layout Module Level 1[1] (see example in Figure 4.7a).
- **Picture-in-picture Layout (PiP)** (see example in Figure 4.7b).
- **Menu Layout** (see example in Figure 4.7c).
- **Horizontal Layout** (see example in Figure 4.7d).

The tests were done with 47 users, one by one, being always an expert presenting each one of the 16 context situations. The expert gave them a very brief description of the context of the testing and ask them to choose always the layout they would prefer on that moment to see the F1 race. All the tests were carried out in the Digital Home Lab

[1] `http://dev.w3.org/csswg/css-grid/`

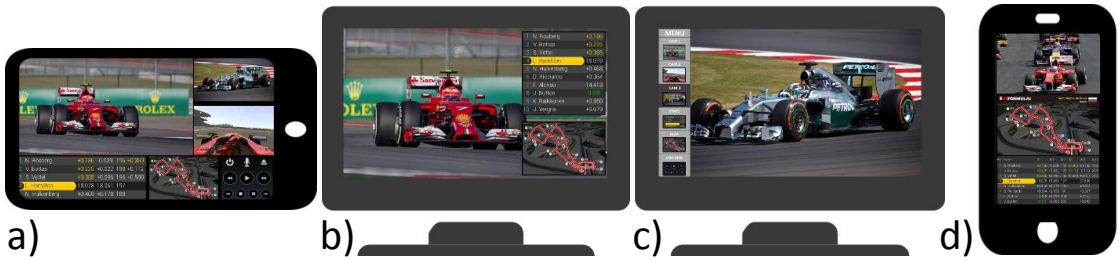


Figure 4.7: a) A grid template layout example on a tablet in the context with ID number 6. b) A PiP template layout example on a TV in the context with ID number 1. c) A menu template layout example on a TV in the context with ID number 2. d) A horizontal template layout example on a smartphone in the context with ID number 9.

Figure 4.8: Images from user tests. In the left a user in front of the context situation with ID number 16 with the menu layout on the tablet. In the middle a user with the context situation with ID number 16 with the PiP layout in the tablet and in the right a user in the context situation with ID number 9 with the horizontal layout on the smartphone.

of Vicomtech-IK4, where there is a similar environment on what we can find on a living room. From the 47 users, 40 of them were researchers in Vicomtech-IK4, with expertise on different fields, and 7 of them were administrative staff people. It took around 15 minutes to perform the test with each user, so around 12 hours in total, divided in three different days. Figure 4.8 presents pictures took during the tests.

4.5.4.1 **Results of the user tests and conclusions**

Table 4.12 presents the answer that the users gave for each one of the 16 contexts. Moreover, the following sub-sections present the behaviour of each one of the users when one of the parameters changed, in order to evaluate which parameter changed their

mind. It measures if the user selected a different layout when the context changed.

Table 4.12: Results of the chosen layouts on each situation

Id	Results LAYOUT: NUMBER OF ANSWERS | PERCENTAGE			
	Grid	**Pip**	**Menu**	**Horizontal**
1	19 | 40%	13 | 28%	14 | 30%	01 | 02%
2	32 | 68%	08 | 17%	06 | 13%	01 | 02%
3	28 | 60%	16 | 16%	03 | 03%	00 | 00%
4	33 | 70%	07 | 15%	05 | 11%	02 | 04%
5	09 | 19%	26 | 55%	11 | 23%	00 | 00%
6	34 | 72%	08 | 17%	05 | 11%	00 | 00%
7	23 | 49%	14 | 30%	07 | 15%	03 | 06%
8	31 | 66%	09 | 19%	07 | 15%	00 | 00%
9	05 | 11%	04 | 09%	20 | 43%	18 | 38%
10	22 | 47%	03 | 06%	19 | 40%	03 | 06%
11	11 | 23%	08 | 17%	09 | 19%	19 | 40%
12	16 | 34%	08 | 17%	15 | 32%	08 | 17%
13	17 | 36%	17 | 36%	13 | 28%	00 | 00%
14	28 | 60%	07 | 15%	10 | 21%	02 | 04%
15	30 | 64%	06 | 13%	10 | 21%	01 | 02%
16	33 | 70%	03 | 06%	11 | 23%	00 | 00%

Analysing the impact of each one of the four parameters defined to be measured in this prototype, we conclude that the four ones are relevant and have a big impact above 45%. This means that at least 21 of the 47 users tested change the mind when one of the parameters changes in the context.

Figure 4.9 shows that the most relevant parameter is the device, with an impact above 59%. As happens for “single device” application, it was also expected to be an important parameter for multi-device applications, where the user is consuming an application from more than one device at the same time. The number of components parameter is also very relevant being around 5 percentage points below. This means that it is important to change the arrangement of the user interface depending the outcome of the adaptation engine while the number of components to be shown changes.

The nature of the application and the number of devices being used at the same time are also relevant parameters, both of them around 46%, to take them into account to develop the final software of the UI Engine.

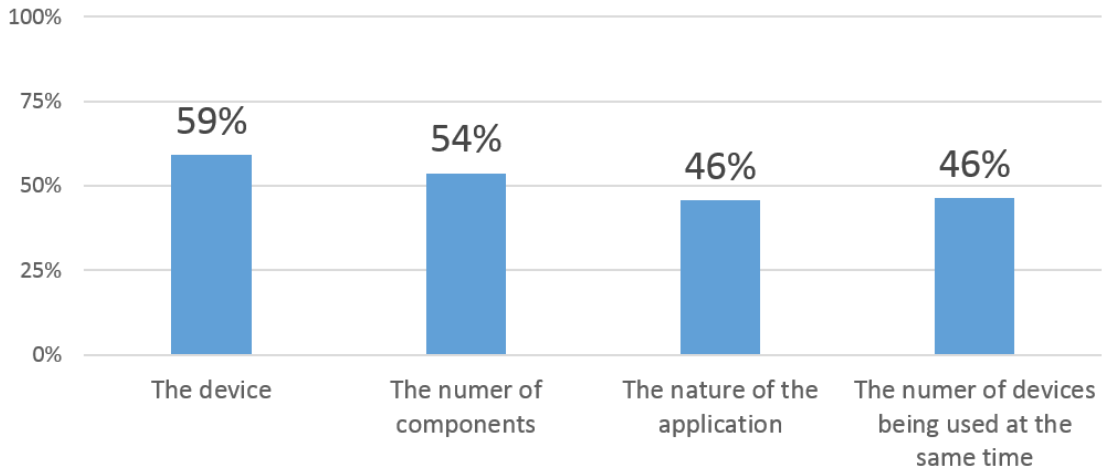


Figure 4.9: Impact of the four defined parameters

These results provide a valuable information in order to have an illustrative idea of how a layout should be chosen when creating responsive User Interfaces for Web-based multi-device media services. However, in order to obtain more accurate results, further research and user tests are needed, using a real multi-device service instead of images simulating it, widening the range of layouts analysed and performing the tests with a higher sample of users with different profiles.

Going on with out prototype implementation, we choose a set of layout templates of standard use in current applications and devices. These are Picture-in-Picture (Figure 4.10A), Split (Figure 4.10B) and Carousel (Figure 4.10C), which we describe in terms of the properties we defined in Section 4.3.3:

1. **Picture-in-Picture (PiP)**: space sharing template in which a component is shown in full screen and the others are overlapped with a smaller size over the first one. PiP shows all the components at the same time to the detriment of the shown area of the first component (overlapped) and the resolution of the secondary components.

2. **Split**: space sharing template in which the screen is divided into a regular grid according to the number of components. Unlike PiP, there is no overlapping. However, empty space appears when the number of components does not match the number of cells established by the template.

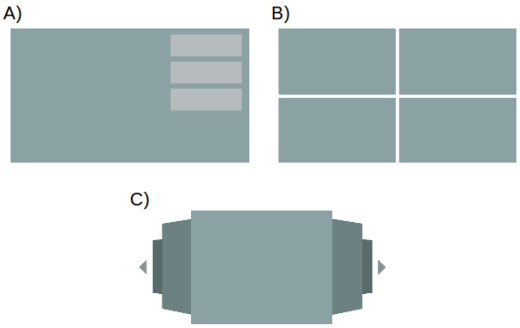


Figure 4.10: Considered layout templates

Table 4.13: Layout properties for each template

Property	PiP	Split	Carousel
Time sharing	✗	✗	✓
Space sharing	✓	✓	✗
Overlapping	✓	✗	✗
Scrolling	✗	✗	✗
Distortion	✓	✓	✓
Prior components	✓	✗	✗

3. **Carousel**: time sharing template in which all the components are organised in a carousel that allows to left- or right-slide all the elements. Carousel is commonly used for image galleries.

A summary of the template layout properties is shown in Table 4.13.

Observe that, contrary to components and devices, for layouts the typification function $\mathscr{T}^{\mathscr{L}}$ does not have practical interest. Instead, in an implementation the aim is, given a template t, finding a layout l such that $\mathscr{T}^{\mathscr{L}}(l) = t$ and that l ranks a high quality in terms of the layout quality criteria explained in Section 4.4.2. Therefore, ideally $l = l^*$.

4.5.5 Implementing assignation

Step 1 of the adaptation process uses the vectors of properties for both components and devices to calculate affinities on the basis of the affinity matrix defined in 4.4.1.

In a real deployment, the matrix values should be carefully set, which involves extensive user evaluation and/or a continuous adaptive learning process. However, the

scope of the present work is aimed at the formulation and validation of an adaptation model. Henceforth we have built the affinity matrix with values obtained by polling a reduced community of users.

Once the affinity matrix and property vectors have been set, the implementation of Step 1 is quite straightforward. Flexible data structures have been implemented to allow for adding or removing components or devices and evaluating all their properties.

To assign a given component, once the affinity vectors for each target device obtained as represented in Figure 4.2, we follow these criteria:

1. Choosing a device fitting every property, or the highest amount of them. In our example, the TV would be the selected device for the main programme component.

2. Minimizing under-fitting values. In our example, in absence of a TV, we would assign the component to the smartphone.

3. Finally, over-fitting values are considered, choosing the device whose affinity values are the closest to the requirement values.

Different implementations could choose different criteria. An alternative implementation could consider only the scalar affinity value (as done in Eq. 4.25). Another implementation could be interested in maximising the usage of resources. Following the example of Fig. 4.2, with this implementation the TV would be the best device, while the laptop would be discarded as choosing it would lead to wasting processing and interactivity resources.

4.5.6 Implementing representation

Step 2 uses the assignments obtained in Step 1 to calculate the layout template in each device.

Our prototype implementation for layout selection is based on the screen size property of the device types (we will add the property of input capabilities in Section 4.6). To properly define the screen size property, we considered the *screen apparent area*, denoted as S.

To measure the apparent area of a screen with diagonal h meters and a normalized 1:1 aspect ratio, we situate the screen at the usual watching distance and calculate the

projected area of the screen to distance 1 meter. Note that this results in more realistic relations than the ones obtained comparing raw screen areas. For example, a screen with a diagonal size h situated at a distance of l meters from the user has the same apparent area as a kh screen at kl meters.

Resolution has been ignored for the screen size definition, since it is not a constraint in current devices. However, some ergonomic aspects should be considered in real deployments. For example, for similar apparent sizes, a TV would provide a more comfortable viewing distance for the user's eyes, while smartphones or table

4.5.7 **Implementing representation**

Step 2 uses the assignments obtained in Step 1 to calculate the layout template in each device.

Our prototype implementation for layout selection is based on the screen size property of the device types (we will add the property of input capabilities in Section 4.6). To properly define the screen size property, we considered the *screen apparent area*, denoted as S.

To measure the apparent area of a screen with diagonal h meters and a normalized 1:1 aspect ratio, we situate the screen at the usual watching distance and calculate the projected area of the screen to distance 1 meter. Note that this results in more realistic relations than the ones obtained comparing raw screen areas. For example, a screen with a diagonal size h situated at a distance of l meters from the user has the same apparent area as a kh screen at kl meters.

Resolution has been ignored for the screen size definition, since it is not a constraint in current devices. However, some ergonomic aspects should be considered in real deployments. For example, for similar apparent sizes, a TV would provide a more comfortable viewing distance for the user's eyes, while smartphones or tablets can be brought nearer more easily.

Table 4.14 shows an example of the estimated apparent areas for the four considered device types as well as some related parameters that we explain next.

A relevant parameter for layout generation, specifically for the Split template, is the minimum apparent size of a component to be comfortably seen by the user, denoted

Table 4.14: Parameters related to apparent areas of screens

Device	Smartphone	Tablet	Laptop	TV
Screen diagonal (cm)	12.5	25	35	150
User distance (cm)	40	50	60	250
Angle (degrees)	17.36	26.58	30.25	30.93
Apparent area, S (m^2)	0.044	0.100	0.127	0.132
Comp. min size, S_c (m^2)	0.02	0.02	0.02	0.02
N. of components, $\lfloor S/S_c \rfloor$	2	4	6	6

Table 4.15: Screen related layout parameters

Parameter	PiP	Split	Carousel
Max N. of main comp.	1	S/S_c	1
Insertion frac., F_i	0.33	0	0
Insertion min size, S_s	$S_c/3$	NA	NA
Max. N. of insertions	$F_i S/S_s$	0	0

as S_c. Indeed, this is a user-related parameter that would deserve extensive user evaluation, in addition to the ability for configuration and adaptation from user context parameters. To provide an example to illustrate the prototype implementation of our adaptation model, we will set S_c to 0.02 square meters. This results, for example, in one half of a smartphone screen at 40cm from the user. For the other devices, Table 4.14 shows the number of minimum size components that would fit into the screen, i.e., S/S_c, conveniently rounded to an integer number.

For PiP layouts, two additional parameters are required: the fraction of the screen area devoted to inserting the overlapped secondary components, F_i, and the size of the secondary components, S_s which in general can be smaller than S_c. Note that F_iůS/S_s denotes the maximum number of insertions in a PiP layout. Table 4.15 summarizes the values for the three templates adopted in the implementation of Step 2.

We have implemented three criteria for the evaluation of the layout quality, according to the evaluation model introduced in Section 4.3.3:

- **Rate of the components shown** (α_1): This criterion evaluates the portion of the components that is shown in the display, and refers to three different aspects: (a) the portion of the number of components, (b) the portion of the component area, and (c) the portion of time that a component is shown. Note that Split layouts

rank the highest in every aspects, while PiP is penalised in (b) by the overlapped area of the main component, and the time sharing layouts, such as Carousel, are penalised in (c) for the time slice where components are shown.

- **Representation of every shown component in its entirety (α_2)**: This criterion is related with how the components shrink to fit to a specific space slot in the space-sharing layouts. The criterion ranks the maximum for Carousel, while it can penalize Split and PiP in two different aspects: (a) shrink in the scale of a component, and (b) distortion of the aspect ratio of the component (the last aspect has not been considered in our example implementation).

- **Efficiency in the use of the screen area (α_3)**: Although we are allowing some degree of distortion, in Split layouts the screen is divided in a regular grid that, for practical reasons, excludes extreme configurations (e.g., many components in a single row). As a consequence, for some specific numbers of components, an unused screen area will be generated, which is accounted for as lost space and penalises this criterion.

Observe that the above criteria, in the framework of the adaptation model defined in Section 4.4, allow not only for the evaluation of the three pure layout templates we are considering for the implementation but also for a combinatorial of hybrid layouts, as, for instance, a carousel showing several components simultaneously, as well as many others.

Once the three criteria have been defined, the quality of each layout can be calculated as shown in Eq. 4.28. We have performed an empirical set up of the internal coefficients in the equation to obtain intuitively realistic outcomes, resulting in the quality graphs in Fig. 4.11 for each device type in terms of the number of components to show. The coefficient used for each of the aforementioned criteria has been $w_1 = 1.4$, $w_2 = 0.3$ and $w_3 = 1.2$ respectively.

Observe that time-sharing layouts (Carousel) maintain a significant quality level for a high number of components whereas space-sharing layouts (PiP and Split) are more suitable for a low number of components, depending on the particular election on the

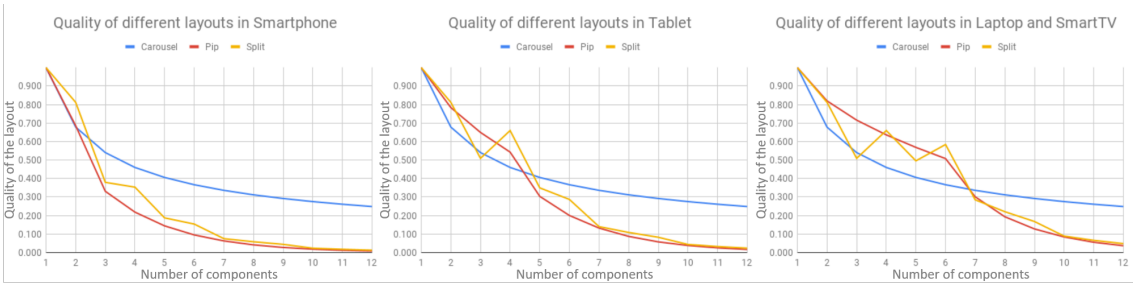


Figure 4.11: Quality of the layouts in terms of number of components

device type and the efficient fit of the components in the screen (e.g. the quality decreases sharply for Split layout when the number of components cannot be arranged in a grid occupying the entire screen).

In a real deployment, the user can be given a way to control the relative relevance of each criteria. Furthermore, an exhaustive user evaluation or an adaptive learning process could find the most appropriated values for the specific deployment and user profile, showing at the same time the power of generalization of our model. Regarding the last point, in Section 4.6 we will show the ability of the model to easily accept extensions and modifications, specifically for including new criteria or properties.

4.5.8 The role of context: triggering the adaptation process

In the previous subsections we have described an implementation of the two-step process. The question now is when the adaptation process should be launched, either from the beginning or only from Step 2 in some specific devices. To address this issue we use context information.

The information about contextual factors and external event occurrences while using a service is usually referred to as context [Hussain et al.18]. In our system, context changes are used to trigger the adaptation process in such a way that the system can update and maintain the user experience at convenient levels.

Usually, context is divided in three parts: user context, physical context and system context. In our scenario, the user context includes the user preferences and user state (e.g., mood or stress level). We will also consider explicit interaction as the user context updates. Physical context parameters can include the location, time, ambient light,

and noise level. Finally, the system context is determined by the state parameters of the devices, including the battery level, connectivity conditions, etc.

As usual, context information is mainly based on sensors and can be elaborated and interpreted by a learning process. In this way, a complete context service could be built on top of our system to boost sophisticated smart environments. Nevertheless, for the scope of this paper, we manage context information as an asynchronous sequence of events. Either directly or indirectly, an event of the sequence is able to promote a run of the adaptation process.

To complete the picture (refer to Fig. 4.1), our multi-device adaptation environment can be implemented as an event-driven system where events can be produced by different situations:

1. A change in the component set C. This change can be produced by the broadcast programme or by an action of the user moving a component from one device to another. As an incremental modification, only Step 2 will be executed in the involved devices.

2. Joining or leaving a device in the device set D, generally involving a rerun of the adaptation process.

3. A context update. For example, when a viewer moves away from the TV, the adaptation process could begin the smartphone to move from a complementary screen to the main element in D. To mention another example, the adaptation process could be triggered in the case of a low battery level in a device, forcing a Situation-2 event.

4. In some cases, described next, an internal event can trigger a complete rerun of the adaptation process.

The last of the situations refers to the fact that the two-step incremental adaptation process could sometimes lead to an unbalanced assignation of components on devices. Devices with many components assigned could rank low for every layout templates in Step 2 and, furthermore, overloaded devices could suffer resource exhaustion, resulting in poor performance, such as in video rendering.

A strategy to avoid this situation is considering resource availability in each device as a system context parameter. When a resource is almost exhausted, an internal event is produced to rerun the entire adaptation process. Note that the strategy should be complemented with the introduction of the resource availability context parameter as a criterion in the assignation step. As Step 1 is implemented iteratively, a low value for resource availability in a device will prevent assigning a new component to the device despite the affinity value.

4.6 Validation

This section provides a validation of the proposed methodology for the user interface adaptation of multi-device media services regarding three different aspects: the quality, efficiency and universality. On one hand, the quality and the efficiency are validated through the evaluation of the two-step adaptation process with experiments that compare the sub-optimal outcome with the optimal adaptation solution (which requires exploring the whole solution tree), analysing the trade-off between the adaptation quality and the computation effort for a responsive implementation. On the other hand, the universality of the proposed methodology is evaluated taking an environment based on a real deployment as a reference [Domínguez et al.17] and overcoming its limitations towards the versatility and flexibility to fit to novel properties, components, devices, use cases, etc.

Figure 4.12 shows a picture of a scenario extracted from the deployment described in [Domínguez et al.17] where a viewer consumes a multi-device TV programme delivered by a broadcaster. That experience has been the starting point to develop the hypotheses of this work. In the same way, it has been useful to synthesize the use cases addressed in this section which are seen as a tool to validate the developed adaptation model.

4.6.1 Evaluation of the two-step adaptation process

To evaluate the two-step adaptation process, the proposed model has been implemented [moda] defining a set of components, a set of available devices, a set of available

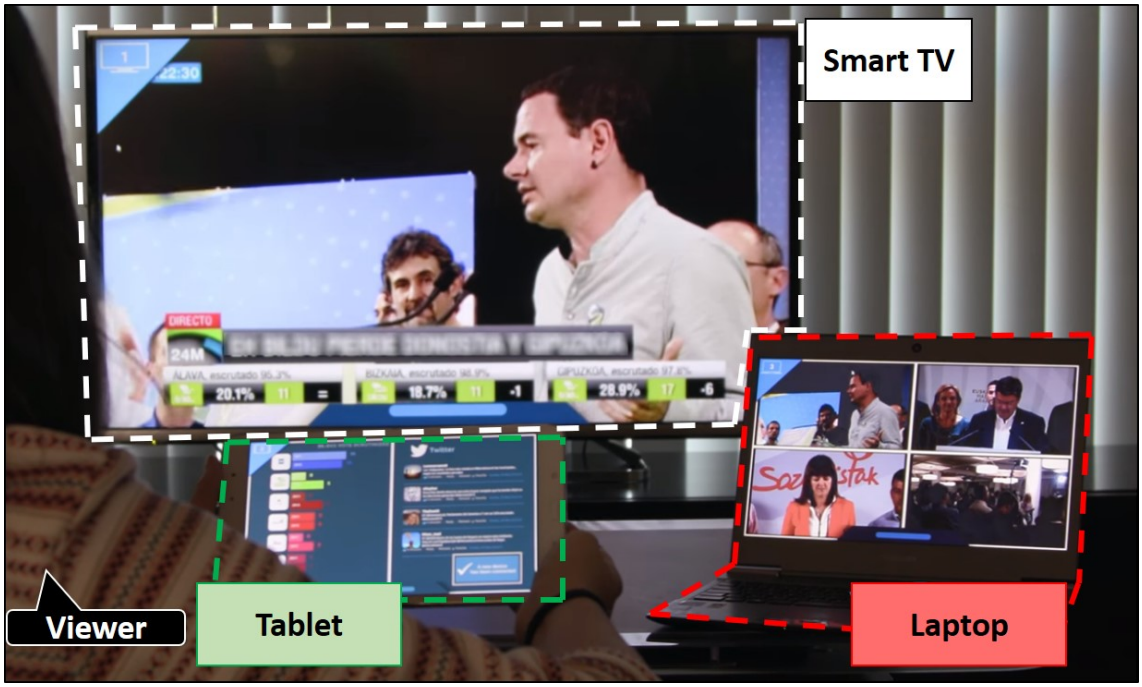


Figure 4.12: A picture of the deployment of a multi-device media service

layout templates, and defining representative properties and parameters for all the user interface elements.

On the one hand, five properties have been defined and evaluated for each of the components. Table 4.16 shows the parametrisation performed in the implementation for all the component types identified in [Dominguez et al.18].

On the other hand, four available connected devices have been parameterised and, again, five different properties have been defined for each device type. Table 4.17 shows the parametrisation performed in the implementation for all the identified device types.

Additionally, Table 4.18 shows the affinity matrix which relates the properties of components and devices.

Finally, three different use cases have been defined, as summarized in Table 4.19, using a subset of components and devices addressed in the implementation.

Table 4.16: Summary of used component types and properties

Properties	Attention	Interactivity	Processing req	Broadcast req	Confidentiality
Main	1	0	0.3	1	0
Sec. videos	0.7	0.3	0.9	0	0.1
Banner	0.8	0.2	0.1	0	0.1
St Data	0.6	0.2	0.2	0	0.3
Dyn Data	0.6	1	0.3	0	0.8
Social	0.3	1	0.3	0	1
UGC	0.3	1	0.7	0	0.9
Adv	0.9	0	0.5	0.8	0

Table 4.17: Summary of used device types and properties

Properties	Screen size	Input cap.	Processing cap.	Privacy	Broadcast cap.
Smartphone	0.2	0.7	0.6	1	0
Tablet	0.6	0.8	0.7	1	0
Desktop	0.7	1	1	0.5	0
SmartTV	1	0.2	0.4	0.1	1

Table 4.18: Affinity matrix

	Screen size	Input cap.	Processing cap.	Privacy	Broadcast cap.
Attention	1	0	0	0.1	0
Interactivity	0.1	1	0.5	0.9	0
Processing req.	0.3	0.3	1	0	0
Broadcast req.	0	0	0	0	1
Confidentiality	0	0	0	1	0

Table 4.19: Summary of the use cases

USE CASE 1	
7 Components	Main Programme, Video 1, Video 2, Video 3, Video 4, Dynamic Data, Social
1 Device	TV
USE CASE 2	
7 Components	Main Programme, Video 1, Video 2, Video 3, Video 4, Dynamic Data, Social
2 Devices	TV, Smartphone
USE CASE 3	
7 Components	Main Programme, Video 1, Video 2, Video 3, Video 4, Dynamic Data, Social
3 Devices	TV, Smartphone, Laptop

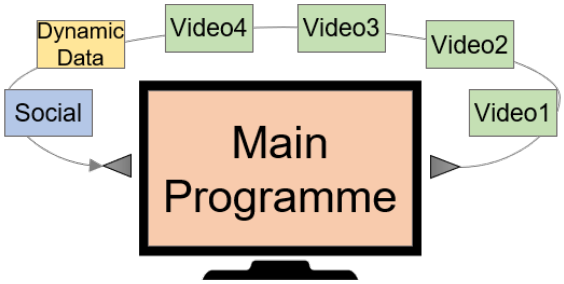


Figure 4.13: Diagram of the adaptation outcome in use case 1

4.6.1.1 Use case 1 - Seven components on a TV

This use case validates the implementation with the aforementioned 7 components having only a Smart TV. As expected, all the components are shown on the TV and the Carousel is the selected layout (see Figure 4.13).

The overall adaptation quality is $\mathscr{E} = 0.28$. In this context, the viewers will be able to mainly follow the TV show, while having the opportunity to perform a kind of content-hopping using the arrows in the remote control to see the other components.

It is worth noting that the adaptation value is severely affected by the inclusion of the *Social* content. This is mainly due to the high interactivity it requires and the low input capabilities the Smart TV is supplied with.

Accordingly, if the social content is not displayed, the overall quality for showing the remaining six components on TV will increase to $\mathscr{E} = 0.52$. On one hand, the quality of

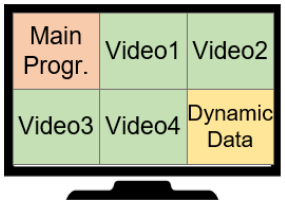


Figure 4.14: Diagram of the adaptation outcome in use case 1 if the social content is not displayed

Figure 4.15: Diagram of the adaptation outcome in use case 2

the assignment increases, since the components are more suitable on average, and on the other hand, the quality of the representation also improves, since with six components, the TV shows the components with the Split layout instead of using the Carousel layout (see Fig. 4.14).

4.6.1.2 **Use case 2 - Adding a smartphone**

If a viewer is consuming the aforementioned 7 components using two devices simultaneously (a TV and a smartphone), the recommended assignation is to show the main programme and the four secondary videos on TV, while showing dynamic data and social components on the smartphone (see Figure 4.15). This scenario represents an overall quality of $\mathscr{E} = 0.66$, increasing the previous values. This mainly occurs because the components that were less suitable for the TV are now on the smartphone and also because the TV now has to represent less components, allowing a PiP layout.

Analysing all the adaptation solutions, we found an optimal quality figure $\mathscr{E} = 0.76$ showing the social content on the smartphone, while displaying the other components in the TV using the Split layout (see Figure 4.16).

In this case our two-step solution offers a quality close to 90% ($0.66/0.76 = 0.87$) of the optimal.

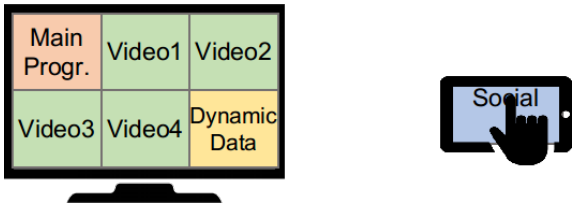


Figure 4.16: Diagram of the best possible adaptation outcome in use case 2

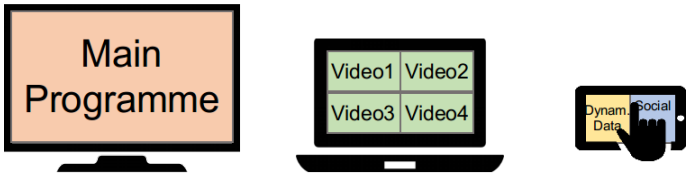


Figure 4.17: Diagram of the adaptation outcome in use case 3

4.6.1.3 **Use case 3 - Simultaneously using three devices**

If a viewer is consuming the 7 components using a TV, a laptop and a smartphone at the same time (as in Fig. 4.12), with the properties and values specified at the beginning of this sub-section, the two-step adaptation process provides the outcome shown in Fig. 4.17.

This scenario ends with an overall quality outcome of $\mathscr{E} = 0.81$. It is interesting to note that with more devices, the outcome quality increases because there are components that fit (almost) perfectly to each of the devices and, with less components on each device, the quality of the representation improves. A discussion of how easy it is for a viewer to consume a content from several devices simultaneously could arise here. Thus, a parameter to penalise the use of multiple devices at the same time could be added, due to the different factors that could contribute to cognitive load, e.g., those listed in [Hart and Staveland88].

In use case 3, the overall optimal solution (shown in Fig. 4.18) results in a quality of $\mathscr{E} = 0.85$. While the optimal quality figure is marginally better than the one obtained with the two-step process, it requires evaluating ($N_D^{N_C} = 3^7 = 2187$) combinations, which is much more time and energy consuming. This was checked through an analysis of the computational efficiency of the model.

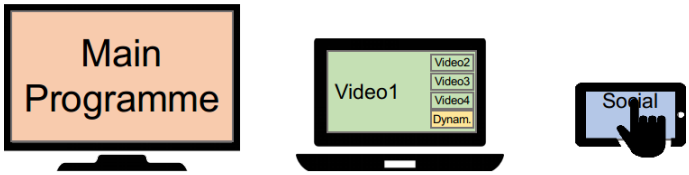


Figure 4.18: Diagram of the best possible adaptation outcome in use case 3

Table 4.20: Computational cost

Device	Tablet	Laptop	TV
Latencies for two-step adaptation (s)			
Scenario 1	0.011	0.003	0.063
Scenario 2	0.026	0.006	0.137
Latencies for optimal adaptation (s)			
Scenario 1	0.671	0.240	1.196
Scenario 2	9.744	3.179	28.680

4.6.1.4 Evaluating computational efficiency

To analyse the computational efficiency, we ran specific performance tests using both the two-step adaptation process and the optimal solution search. Two scenarios have been analysed: scenario 1 considers 7 components, 3 devices, 5 properties per component, 5 properties per device, 3 layout types and 3 layout criteria; scenario 2 considers 10 components, 3 devices, 10 properties per component, 10 properties per device, 3 layout types and 6 layout criteria. Table 4.20 shows the time required by each device to provide a solution to the given scenarios.

In the first scenario, even if the two-step adaptation is much more efficient, neither method would impact the responsiveness of the user experience. However, in the second scenario, the times required to calculate the optimal solution dramatically increase, resulting in unacceptable poor responsiveness.

As a summary of the validation of the two-step adaptation process, the experiments show that, when parameterising the proposed methodology with reasonable figures, the outcome fits adequate multi-device user interfaces, mostly coinciding with the outcome of the specific adaptation rules developed by broadcasters and researchers during the application deployment in [Domínguez et al.17] and providing a responsive (although sub-optimal) outcome efficiently.

4.6.2 Validation of the universality of the methodology

One of the goals of our adaptation model is to be general enough to adapt to technological evolution (e.g., forthcoming devices) and new broadcast trends or user habits, to simplify developers' work and allow for adaptive UIs on top of the adaptation system. That is why the model can be adapted, modified or oriented to other specific use cases and circumstances, while still following the proposed methodology. We have identified three main dimensions that could be added or modified:

- UI elements (types of components, devices or layouts) and properties to adapt the model to technological and broadcasters' changes;
- Evaluation criteria, to adapt the model to the user;
- Additional context parameters that can improve the user experience. These include parameters of user context (e.g., location in the room), physical context (e.g., ambient light), and infrastructure context (e.g., saturation of device capabilities due to an excess of assigned components).

In this sub-section, different examples are provided to extend the proposed methodology and evaluate its universality.

4.6.2.1 Adding a new property to the component and device types

Consider, following the deployment performed in [Domínguez et al.17], that the *Dynamic Data* component shows vote counting through a complex 3D graph type and that the *Secondary Videos* have a high resolution with high decoding demands. In these cases, strong graphic capabilities would be useful.

This will require an additional property to be considered, called *Graphic/video requirements*, for the component types, having a value of 1 for *Dynamic Data* and 0.3 for the *Secondary videos* (see Table 4.21). In the same way, an additional property could be considered for the device types, called *Graphic/video capabilities*, where TVs and mobile devices (both smartphones and tablets) would have a poor value (see Table 4.22) and laptops would have the highest possible value. The affinity matrix will emphasise the impact of the *Graphic/video requirements* with the *Graphic/video capabilities* (see Table 4.23).

Table 4.21: Components parametrisation for Graphic/video requirements

Properties	Graphic/video Requirements
Main	0
Secondary videos	0.3
Banner	0
Static data	0
Dynamic data	1
Social	0
UGC	0
Advertisement	0

Table 4.22: Devices parametrisation for Graphic/video capabilities

Properties	Graphic/video capabilities
Smartphone	0.2
Tablet	0.2
Laptop	1
SmartTV	0

Table 4.23: Affinity matrix considering Graphic/video capabilities and requirements

	Scr. size	Inp. cap.	Proc. cap.	Priv.	Broad. cap.	G/V cap.
Attention	1	0	0	0.1	0	0
Interactivity	0.1	1	0.5	0.9	0	0
Processing req.	0.3	0.3	1	0	0	0
Broadcast req.	0	0	0	0	1	0
Confidentiality	0	0	0	1	0	0
Graphic/video req.	0	0	0	0	0	1

Table 4.24: Input requirements for each layout

Parameter	PiP	Split	Carousel
Ir	0.4	0	0.4

With the aforementioned novel scenario, if we replicate the environment provided by *use case 3*, the outcome of the two-step adaptation shown in Figure 4.18, which is the optimal solution, is obtained instead of the previous one (shown in Figure 4.17), with a quality $\mathscr{E} = 0.85$.

4.6.2.2 **Adding a new criterion for user interface layouts**

Apart from the aforementioned visual criteria to select the best user interface layout in each case (recall that Table 4.14 and Table 4.15 were constrained to parameters related to the screen area), a broadcaster or content provider may want to consider the intrinsic interactivity required by the layout itself. For instance, a carousel layout requires the interaction of the viewer to spin the components of the carousel and select the one to be seen in fullscreen. This interaction will be the most difficult with the remote control of a TV, while easiest with a trackpad on a laptop or a touch-screen device. In the same way, PiP layouts could benefit from this interaction facilities. Regarding the Split layout, however, the interaction is not a relevant parameter according to the opinion of broadcast professionals and researchers that participated in the deployment described in [Domínguez et al.17]. To take these aspects into consideration a fourth criteria α_4 has been considered, named *Interactivity compliance* and defined as the degree in which a layout is manageable in terms of interaction when it is displayed in a specific device.

For the calculation of the aforementioned criteria, an additional layout parameter has been added to those defined in Table 4.15, named *Interactivity requirements* (Ir) and defined as the level of interaction required by the layout to be managed by the user. Table 4.24 shows the values given to Ir for each layout.

Furthermore, an additional dimension has been added to the parameters related to each device (see Table 4.14) that are considered to calculate the quality of a layout. In this case the new parameter has been the *Input capabilities* (Ic) of each device. Table 4.25 shows the values given to Ic for each device.

Figure 4.19 shows the representation of the quality of the layouts in terms of the number of components for Smart TV devices, after adding the criteria of interaction of

Table 4.25: Input capabilities per device

Device	Smartphone	Tablet	Laptop	SmartTV
Input cap. (Ic)	0.7	0.8	1	0.3

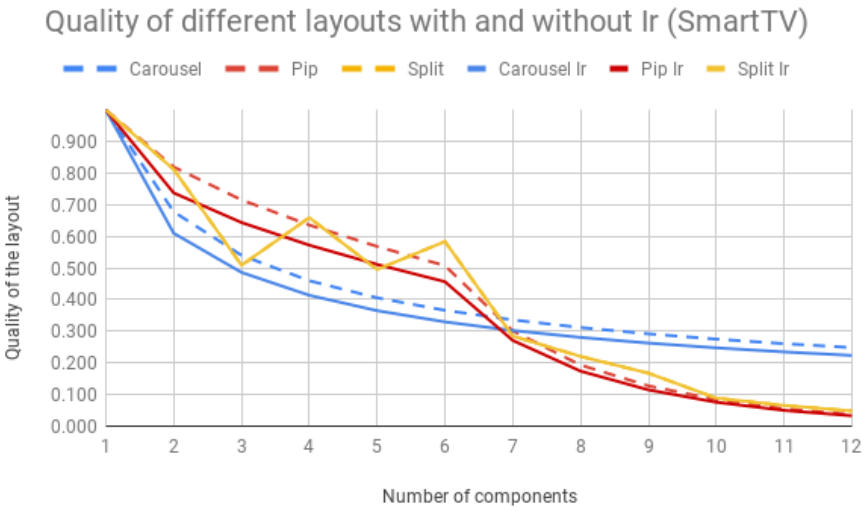


Figure 4.19: Quality of the layouts in terms of number of components with interaction criteria (straight line) and without (dashed).

the layouts. Again, this representation follows Eq. 4.28, with $k \in [1,4]$ and using $w_1 = 1.4$, $w_2 = 0.3$, $w_3 = 1.2$ and $w_4 = 1$ as the weights for each criterion, where w_4 denotes the weight assigned to the new interaction criteria.

As shown in Figure 4.19, the quality of the layouts changes when using the interaction compliance criteria: the quality decreases for both the Carousel and PiP layouts, reinforcing Split as the best layout.

Other user interface layouts criteria could also be considered, such as the prioritisation of a specific component in comparison with the other components represented on the same screen. For example, the Split layouts seem to harmonise the priority of all the components shown in the display, even though the position itself could be a way to prioritise. However, in the PiP layout, the predominance of a component over the other seems to be much clear. A reasonable criterion could improve the quality of the PiP layout for a device that contains the main programme (such as the TV), while promoting a more equitable layout, such as Split, for a device that shows some secondary videos

(such as a laptop).

As a summary, this sub-section shows that the proposed methodology is universal, efficient, flexible and adaptable to any kind of multi-device media service. The model will not change depending on the use case, it just needs to be fine-tuned by adding or modifying parameters.

4.7 Conclusions

This chapter proposes a methodology for multi-device user interface adaptation in media services, such as a hybrid broadcast-broadband TV programme. The definition of this methodology provides a universal adaptation model as an outcome, which identifies and characterises all the elements of the user interface, analyses the role of the context, and provides an approach to address the challenge of dividing the process into two different sub-processes to make it responsive.

The aforementioned research has allowed the model to be formally described and provided a comprehensive example of how an implementation could be performed. Reasonable parameters have been used for this, which provide an outcome that coincides with the output of the adaptation rules developed by the broadcasters and researchers during the application deployment described in Chapter 3 and published in [Domínguez et al.17]. Moreover, the model quality, efficiency and universality have been validated and the following conclusions have been drawn:

- Suboptimal solutions obtained by the two-step adaptation process are reasonably good and meet the expectations of broadcasters and researchers. In the use cases tested they offer a quality of approximately 90% of the optimal.
- The two-step adaptation process is much more efficient than the search of the global optimal solution in terms of the processing time and henceforth of the power consumption. Responsiveness and low latencies are guaranteed through two sequential steps, but not through the exploration of the problem as a whole.
- The developed model is general enough to be extended to any type of content, device, context or evaluation criteria and ready for technological changes as well as continuous adaptive learning processes.

CHAPTER

5

Fields of application

5.1 Overview

The deployment presented in Chapter 3 and the generalisation of the adaptation methodology and model presented in Chapter 4 can be applicable to other fields different from the audiovisual and broadcasting. This means that the same methodology will provide a variation of the model that will be adapted for each case. More in detail, this chapter provides examples of two fields in which the applicability of the solution has been identified, basing on a real technological transfer demand. We refer to Industry and Crisis environments, where there has been the chance to transfer the solution to companies with those demands and it has been seen that the solution was applicable. Next sections describe how Media technologies can improve the actual way of operation in both cases, including the challenges to overcome and the research work and deployments completed so far.

5.2 Industry

Recent developments in information and communication technology (ICT) and the latest Internet-related technologies are opening up revolutionary possibilities for manufacturing and production. The technology surrounding Industry 4.0 and smart factories

is mainly focused in certain fields, such as the Internet of Things (IoT), industrial big data, industrial automation, cybersecurity or intelligent robotics. Nevertheless, it is important to emphasize the relevance of computer graphics and media technologies, under the umbrella of visual computing as key enabling technologies in smart factories of the future.

5.2.1 Media Technologies and multi-screen media services for Operators in Industry 4.0

This section focuses on the role of the operator in the smart factory, which plays a crucial role within the vertical integration dimension in Industry 4.0. Smart factories change the role of operators, making new types of interaction possible between them and the machines. Therefore, human-centricity becomes a key concept, fostering the cooperation of machines and humans toward a more efficient and effective human-automation symbiosis. This concept is related to the Operator 4.0 term mentioned in [Romero et al.16].

In this context, it is interesting to note that many authors explicitly identify augmented reality(AR) as one of the main Industry 4.0 technologies to empower the operator [Funk et al.16]. However, being that AR an important aspect with its own challenges and possibilities, we believe that a broader combination of computer graphics, vision and media technologies can be very important to support operators and to learn from their actions in new smart factory scenarios. In addition, support from knowledge-based and intelligent systems is critical in this context. We will present a set of challenges related to the operator in the context of smart factories along with current examples of relevant related R&D activities in this field.

Figure 5.1 shows the role of operators in smart factories, where three major challenges can be identified:

The first challenge is to support operators performing tasks within a process/workflow. In this context, AR plays a crucial role with improved experiences, projecting instructions with the next steps of the workflow onto objects and tools (Figure 5.1, number 1), or enabling projection-based gestural interaction (Figure 5.1, number 2). Moreover, AR technologies and devices can overcome limitations of operators with

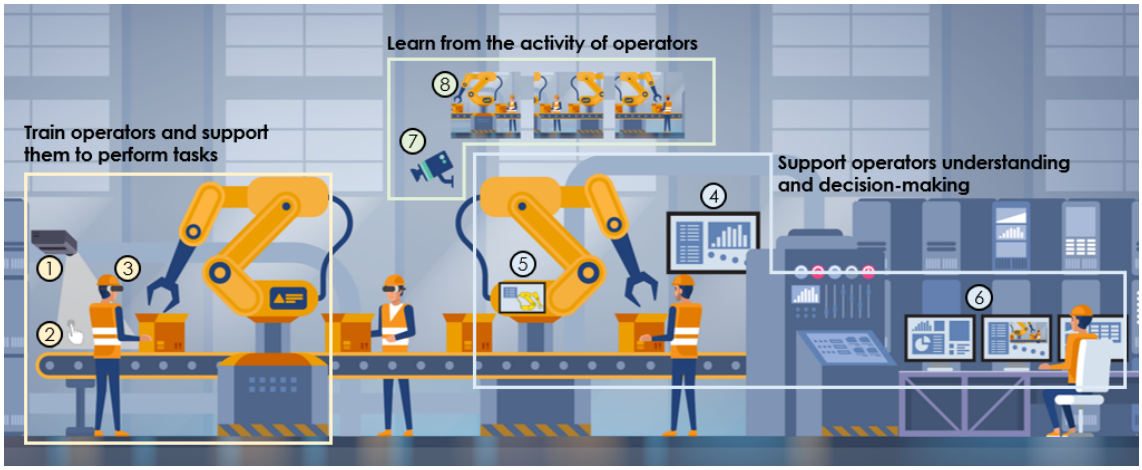


Figure 5.1: Challenges of the operators in Smart Factories

disabilities or special needs(Figure 5.1, number 3).Virtual reality Reality (VR) is also a relevant enabling technology for training activities.

The second challenge is to support operators'understanding and decision-making in all the activities throughout the production process to provide an adequate ecosystem to optimally perform tasks. More versatile and enhanced human-machine interfaces (HMI) are needed, following a human-centred approach and facilitating analytical reasoning through interactive multi-screen interfaces with visual analytics. They need to address operators using different screens, such as shared big screens (Figure 5.1, number 4), and specific machine HMIs(Figure 5.1, number 5)or personal devices (such as smartphones or tablets). All these screens must provide a consistent overview of data and information in the form of graphs or media presentations (Figure 5.1, number 6), personalizing the visualization for each operator's role,and enabling ubiquity to address operators' movement from one set of screens to another.

The third challenge is to learn from the operators'activities to predict specific situations, optimize the process and better organise the smart factory. In this context, video management capabilities (recording, streaming, etc.) are fully required with a complete integration of the Manufacturing Execution System(MES)(Figure 5.1, number 7). On the one hand, video management will enable real-time supervision of the tasks both from inside or outside the premises, while providing an enriched reporting of the production process. On the other hand, artificial intelligence techniques such as pattern recogni-

tion and machine (deep)learning make it possible to learn from operators' tasks and influence in the process(Figure 5.1, number 8).

From the three mentioned challenges the second one is closely related with the work presented in Chapter 3 and Chapter 4 and it leads us to think about how Dynamic multi-screen dashboards for operators in Industry 4.0 would be.

5.2.2 Dynamic multi-screen dashboards for operators in Industry 4.0

Traditional industrial HMIs are very "static," based on old interaction paradigms (WIMP). They include basically some alarms, basic production output data and production planning, and some sensor information. Most of the information offered to operators and engineers is based on static text, pictures in the best cases, and specially data (diagrams, statistics, etc.), shown in external screens attached to the machines, in PC stations close to production, or on normal displays in the shop floor. Production dashboards are a very clear case for this. In order to have better knowledge of the real production environment, a very powerful tool, not sufficiently used in state of the art industrial practice, is needed to capture and use the relevant events in video. It is noteworthy that there is very little adoption of video recording, management, search, etc. in real-world factories.

A recent trend is to include mobile devices for dashboards and other functionalities to some extent, but the approach is based on expensive rugged hardware that runs proprietary software solutions with single-screen paradigms, highly rigid and fixed distribution of information in the screen, and few interaction possibilities.

In this context, it is an interesting opportunity to migrate recent technology advances from other areas with intensive interaction based on dynamic, mobile, multiple screens toward smart factory scenarios. Table 5.1 shows that the broadcast field that apparently is far away from the context, it is quite close in terms of user needs and specifications: we refer to the research advances in the dynamic adaptation of multiple screen visualization and interaction for professional TV experiences, and its synergies with other media (Internet, etc.).

A current research example involves the development of a core dashboard system for an industrial, export-oriented SME that builds tools for heavy lift operations and that owns a modular, portable and monitored Test Bench to perform Load Tests up to 1,000 ton. This is a typical industrial case: An engineer controlling the load test through

Table 5.1: Analysis of the needs of operators in smart factories and TV viewers in broadcasting in terms of dynamic multiscreen visualization and interaction, emphasizing the common aspects

Operator and engineers needs	**TV spectator needs**
Ubiquitous production information in multi-screen format	Ubiquitous media experience in TV screen, mobile devices, tablets, PCs.
Harmonization of heterogeneous data sources (production plan, sensors, CAD, ERP, MES...)	Harmonization of TV broadcast signal with various Internet based information sources
Dynamic synchronisation of sensor data, product design and production planing, even video streams to be able to perform a forensic analysis and look over the processes.	Dynamic synchronisation of social network inputs, internet updates, multiple cameras input.
Personalised experience, relevant to the particular Operator profile and Task requirements.	Personalised experience, relevant to user tastes and interests, interaction preferences, devices.
Automatic localized information according to specific machine, location in plant.	Automatic localized information according to environment (e.g. in a car, in TV proximity, in a mobile)

Figure 5.2: Example of a pilot done in the industry field

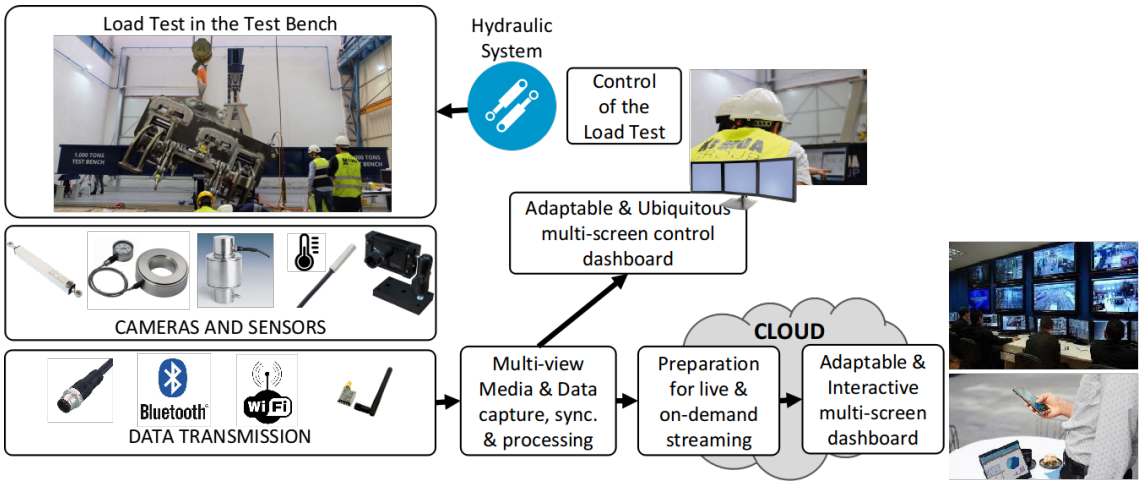


Figure 5.3: Dynamic multiscreen dashboards for operators in Industry 4.0

a specific hydraulic system, with its own HMI, while the outcome information of the test is displayed through different interfaces in machine-specific displays. A team of distributed operators is visually controlling the load test and following the different displays, interacting with the engineer to fine-tune the test. As the test are run in international locations, the engineer has to physically travel to the sites. This is a case of horizontal integration of Industry 4.0, in which emerging internet-based systems allow a deeper integration between industrial clients and providers, empowering Operators and Engineers to work collaboratively even in remote international locations, and avoiding the need of costly physical travel and ad-hoc adaptations of user interfaces. Figure 5.2 and Figure 5.3 show the pilot tested in this field. In this context, a more versatile and smart solution was desired, providing:

- An ubiquitous, adaptable and very flexible multi-screen dashboard to monitor multiple camera views and the outcome data in real time, in order to have all the information when controlling the load test through the hydraulic system.
- A live adaptable and flexible multi-screen visualization solution for engineers around the world to follow the load test remotely, as they were on the premises, combining graphics and media with all the relevant information in an interactive and customizable multi-device interface,
- An on-demand inspection solution of the load test, personalizing the view and verifying all the process and the adequate performance of the test, and the generation of an added value advanced and interactive report as a communication tool for engineers to learn more about the tools that are being measured.

This provides an adaptable and ubiquitous multi-screen control dashboard, where different video sources and heterogeneous outcome data are captured and prepared to be exposed through a Web-based HTML-5 application. All information is shown in any kind of Web-based device, such as a laptop, a tablet or a smartphone. The pieces of information and user interface layout is dynamically configured according to the target device and enables the interaction of the operator. Different devices can be easily associated, such as multiple laptops and mobiles being used by the same operator, and the user interface dynamically adapts to that multi-screen configuration, showing parts of the application on each screen, and enabling the operator to move components from

one device to another. Moreover, the operator can move from one place to another in the premises, dissociating big screens and having all the relevant information in the mobile, while associating new big screens in the new location. Current deployment is well accepted by operators and engineers as it opens more efficient ways to get information and communicate the tests.

In a dashboard like the one described in this section, an multi-screen user interface adaptation model such as the one presented in Chapter 4 could be very valuable since it could automatically distribute the components among the connected devices, providing a responsive layout in each of them. Furthermore, if the model would be enriched through a continuous learning processes, the system would know which configuration is the best for each scenario.

5.3 Crisis environments

Technology can also contribute greatly to crisis environments, especially by enhancing the interconnectedness between the authorities and the public services and by facilitating the rapid exchange of information. Crises take on a variety of shapes and forms —natural or health disasters, terroristic and criminal acts, technology malfunctions, and large-scale accidents— at local, regional, national, and global levels. Crisis management plans, created and implemented at the organizational level, typically involve public service and institutional authorities overseeing emergency response.

In this context, media technologies can facilitate cooperation within response organizations or the response network, but also can enable wider cooperation such as citizen collaboration. To achieve these ends, crisis practitioners and crisis managers are calling for multi-screen technologies that allow to manage multiple sources of information that take into account the perceptions, needs, and practices of multiple users.

5.3.1 Media technologies and multi-screen media services for crisis environments

This section focuses on media technologies that ease a more efficient crisis management, making the exchange of multiple sources of information possible between authorities and public services.

In this context it is interesting to note that solutions that allow the distribution of multiple video signals in any environment, by means of low delay adaptive streaming protocols are important to empower the crisis management. Moreover, we believe that, again, broader combination of computer graphics, vision and other media technologies can be very important to extend the information provided by the video signals and offer to the target viewer the sources that are relevant to make the corresponding decisions. In addition, support from knowledge-based and intelligent systems are also essential in this field.

Figure 5.4 shows the main challenges to manage a crisis environment backed by media technologies.

The first challenge is to cope with multiple video and data signals coming from any environment. This includes, security cameras installed in buildings, TV signals, drones, agents in mobility, data monitored by sensors, social media or even any valuable data found on the Internet. In this context video management capabilities are fully required to enable real-time monitoring of the crisis.

The second challenge is to provide an adaptable, flexible and easy deployment multi-screen media service that allows to centralise and manage several information sources not only based on video signals but also on data extracted from the Internet and authorities' internal data systems. And all of these avoiding to use proprietary softwares and expensive hardware depending solutions. This solutions need to address operators using different screens, such as video-walls, laptops or personal devices. All these scrto be able to perform a forensic analysis and look over the ocurrenceseens must provide a consistent overview of data and information taking into account the role of the target visualiser.

The third challenge is to learn from operators interaction to make even faster the deployment of the needed resources and be able to manage the crisis efficiently from the beginning. Additionally, artificial intelligence techniques over video signals such as object detection and tracking or face recognition could be very helpful to create alarms when a relevant event is identified.

From the mentioned challenges the second one is closely related with the work presented in Chapter 3 and Chapter 4 and it leads us to think about how Dynamic multi-screen dashboards for crisis environments would be.

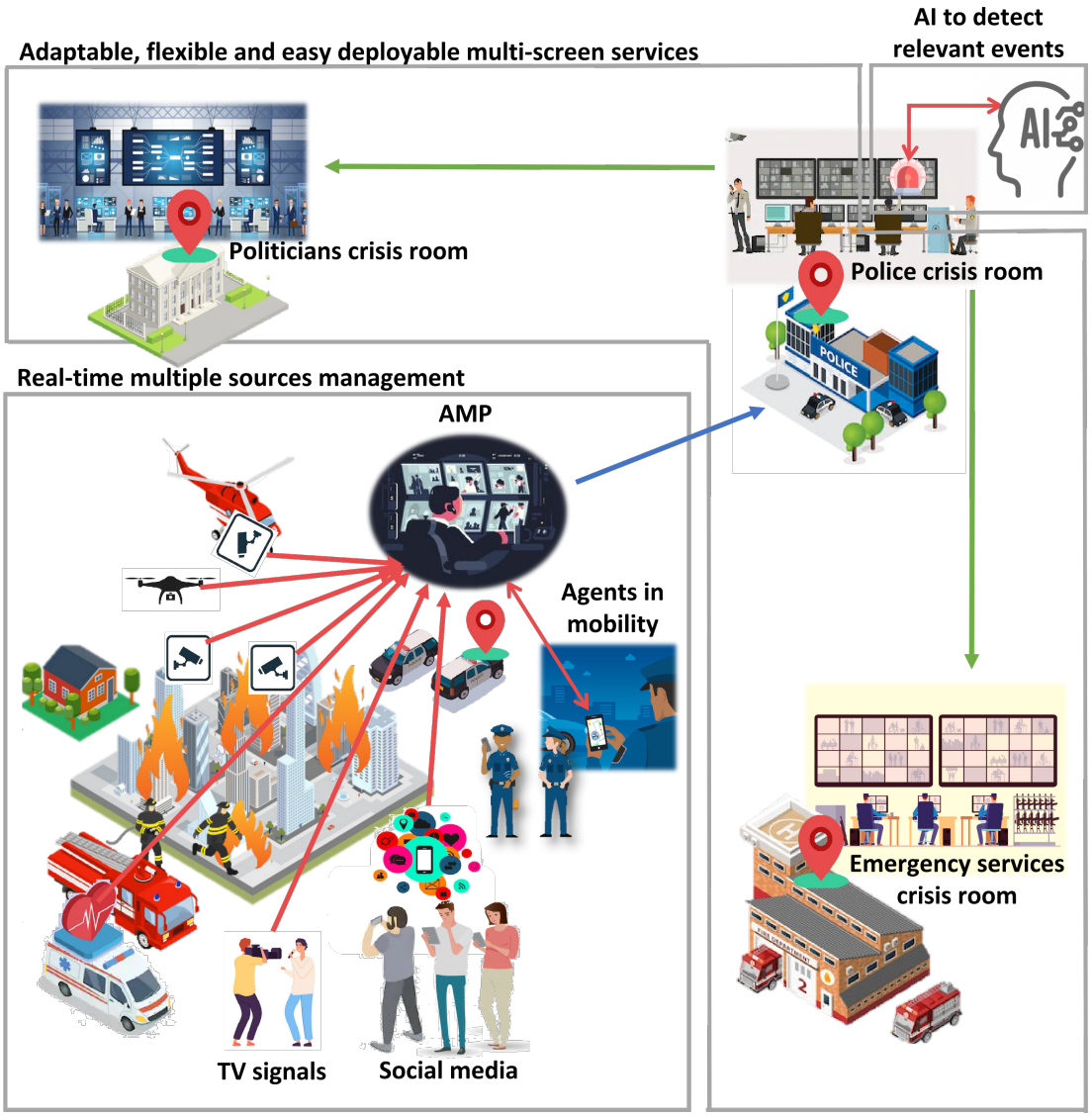


Figure 5.4: Example of a crisis environment

Table 5.2: Analysis of the needs of crisis managers and TV viewers in broadcasting in terms of dynamic multi-screen visualization and interaction, emphasizing the common aspects

Crisis managers needs	**TV spectator needs**
Ubiquitous multi-source information in multi-screen format	Ubiquitous media experience in TV screen, mobile devices, tablets, PCs.
Harmonization of heterogeneous data sources (security cameras, tv signals, agents in mobility, sensor data, social media, news, etc.)	Harmonization of TV broadcast signal with various Internet based information sources
Dynamic synchronisation of video streams, sensor data, action planning or social media to be able to perform a forensic analysis and look over the ocurrences.	Dynamic synchronisation of social network inputs, internet updates, multiple cameras input.
Personalised experience, relevant to the particular crisis manager, authorities and emergency services.	Personalised experience, relevant to user tastes and interests, interaction preferences, devices.
Automatic localized information according to specific specific problem location.	Automatic localized information according to environment (e.g. in a car, in TV proximity, in a mobile)

5.3.2 **Dynamic multi-screen dashboards for crisis environments**

Nowadays different data and video system approaches are used depending on the requirements of each scenario. Therefore, solutions are based on video-walls depending on expensive hardware that runs proprietary software solutions, with highly rigid and fixed distribution of information in the screens, and few interaction possibilities.

In this context we see another interesting opportunity to migrate recent technology advances from other areas with intensive interaction based on dynamic, mobile, multiple screens toward crisis monitoring scenarios, which remove the hardware and proprietary software dependencies and provide a wider range of communication and information sharing possibilities.

Table 5.2 shows the proximity of the broadcast field to this context, in terms of user needs and specifications: we refer to the research advances in the dynamic adaptation of multiple screen visualization and interaction for professional TV experiences, and its synergies with other media (Internet, etc.).

Figure 5.5: Example of a pilot done in a crisis environment

A current research example involves the development of a pilot that allows to monitor an sport event such as the Final Four, which is the final part of the Basketball Euroleague. In this pilot, several information sources are managed by an administrator that has direct view of each source and decides which components to see in each connected screen both in situ and remotely. Furthermore, he can decide the layout template to represent the information in each case. In addition, a team of content managers is be able to operate all the range of content sources, including several video flows, TV signals, investigation information, reserved data, performed action register, public cooperation information, suspects' images, social network entries or even any valuable information found on the Internet. Figure 5.5 shows the pilot tested in this environment.

A similar approach was also implemented to monitor a football match of the Spanish league that took place in San Mames stadium, with the goal of testing a pilot that allows to monitor some matches of the UEFA Euro 2020 that will take place in the same stadium.

Again, these solutions provide an adaptable and ubiquitous multi-screen control dashboard, where different many heterogeneous sources are centralised and prepared

to be exposed through a Web-based HTML-5 application. All information can be shown in any kind of Web-based device, such as a laptop, a tablet or a smartphone. Information can be easily controlled, operated and shared through multiple screens being used by one or multiple users simultaneously and the user interface dynamically adapts to that multi-screen configuration, showing parts of the application on each screen, and enabling the crisis manager to move components from one device to another and share the information in accordance with the role of each agent. Moreover, the system allows agents in mobility receiving or providing relevant information by means of the mobile phone. Current deployment is being well accepted by authorities and engineers as it opens more efficient ways to get information and communicate the tests.

Once again, in a solution like the one described in this section, a multi-screen user interface adaptation model such as the one presented in Chapter 4 could be very valuable since it could automatically distribute the components among the connected devices, providing a responsive layout to each of them. Furthermore, if the model would be enriched through a continuous learning processes, the system would know which configuration is the best for each scenario.

5.4 Common approach

Once the previous fields have been analysed, the common features required by all the cases to succeed (including the broadcast field) are highlighted:

- Web-based, HTML-5 and Web3D compliant technologies to obtain a fully interoperable service.
- Multi-source content management to integrate heterogeneous data into the experience. Data fusion is based on standards.
- Detection and identification of available screens or devices to distribute the content making the most of the resources.
- Information flow between screens to obtain a seamless experience.
- Dynamic time-based alignment and synchronisation of data sources to obtain a coherent visualisation of the content across all the connected screens.

- Dynamic adaptation model, to be able to distribute the content across the available screens and produce responsive and interactive visualization based on user profile.

All the aforementioned proves that there are environments with the necessity of this kind of solutions and in which the methodology is applicable. An effort would be required to characterise the components, devices and layouts in each case in order to generate an specific model in each situation, but the methodology would avoid the definition of an extensive set of rules.

Part IV

Conclusions

CHAPTER

6

Conclusions

6.1 Conclusions

This research work has demonstrated the feasibility to provide a methodology which provides as an outcome and adaptation model for the user interface of multi-device media services. Through a two stage methodology based on obtaining a model from a previous experience a general model has been formalised easily adaptable to many different use cases and scenarios and ready for any technological update.

On the first stage a large-scale pilot deployment of a hybrid broadcast-Internet multi-device service for a live TV programme has been designed and implemented. First some hypotheses have been formulated regarding the mentioned large-scale pilot and then answers have been obtained based on the results of the pilot, which has been used by more than a thousand of users. This pilot has been useful to validate the Web-based distributed architecture for multi-device applications proposed in [Zorrilla et al.15a] and extract some valuable lessons and conclusions to improve several dimensions of COPE based services among which stand out:

- The interest of broadcasters to provide innovative multi-device services within some constraints and as long as it does not imply complexity.

- The need to improve the usability of multi-device User Interface, whose design is the key in guiding the user across all the connected devices.

In order to face these challenges and go a step further in the development of hybrid broadcast-Internet multi-device services for a TV show, an automatic adaptation model that takes into account several user interface elements, design factors and context information is needed. This would provide as an outcome a near-optimal multi-screen experience. Therefore, to make a leap towards the automatisation of the adaptation model, a second stage has been carried out in which the lessons learned from the previous deployment has been used to adjust the initial hypothesis and to define a methodology that allows to generate adaptation models general enough for every use case and technological update. In this second stage, first, a methodology has been defined with different steps to gather the needed information to identify the elements and design factors considered as relevant in the adaptation process of the multi-device user interface. With that information a formal model has been defined, including the characterisation of involved user interface elements, the adaptation process and a evaluation model. Then, implementation examples of the model have been developed in order to validate it in different terms.

Regarding the characterisation of the user interface elements, three different researches have been carried out, once for each identified element i.e. components, devices and layouts.

- Characterisation of the components: an analysis of the content of TV programmes has been done in order to see which contents they show and their relevance. For that, different broadcast emissions have been analysed taking into account different countries and programme types. The analysis has provided as a result a large list of different elements that have been classified into 8 components types according to their common properties. This has been the typification method for the content of the service.

- Characterisation of the devices: devices have been characterised through the information made available by the browser, more in detail, through the User Agent. A comparative empirical analysis has been done with three different approaches

of Web-based device type detection which base their knowledge on different learning systems and statistical models. Therefore, the typification method for devices has been the use of classification algorithms.

- Characterisation of layouts: layouts have been analysed in a slightly different way. First, a set of user tests has been carried out to identify the relevant parameters that suggest the use of a layout template or another. An then, the most representative layout templates have been chosen to implement a model based on the identified parameters.

Regarding the adaptation process, a two-step implementation has been developed following the rationale behind divide-and-conquer strategy. The mentioned two steps have been:

- Assignation: selection of the optimal assignment of components to devices in terms of affinity criteria.
- Representation: search of the optimal layout for each assignment.

This implementation has been completed with an event-driven system to include context information.

Finally, me model has been validated through its evaluation model. A battery of use cases that involve subsets of the characterised components and devices and the parametrisation of their properties and the quality of the resulting user interface have been compared to the optimal. The obtained solutions meet the expectations of broadcasters and researchers and the use cases tested offser a quality of approximately 90% of the optimal. Furthermore, it has been proved that the implemented solution is much more efficient than the exploration of the entire combination both in terms of processing times and power consumption. Moreover, it has been shown that the provided model is general enough to be extended to any type of content, device or context and that it is ready for technological changes as well as continuous adaptive learning processes.

Finally, it has been checked that the methodology proposed which is oriented to hybrid broadcast-Internet scenarios can also be valuable for completely different fields such as industry or crisis environments.

In summary, this research provides progress beyond the-state-of-the-art for the optimisation of user interface of multi-device media services, presenting a methodology that provides as an outcome adaptation models that take into account several user interface elements and design factors to dynamically distribute and adapt the content to multiple devices being used simultaneously, in real-time and in any context. The provided methodology eases developers the generation of COPE based seamless and self-adaptive applications, with simple parametrisations being the only task to accomplish.

6.2 Future Work

During research activities, the literature review, the design and implementation of the model and the analysis of the results, different aspects were identified to complement or extend the research presented in this thesis:

- The exhaustive analysis of each parameter, intuitively or reasonably assigned in the implementation of model defined in Chapter 4, requires exploring many different research lines.

- Specific requirements or limitations could be added for each use case, such as the possibility of creating duplicated components during the *assignation*, as well as not showing specific non-critical components depending on the context. The model could easily assimilate these types of requirements on its implementation by treating it as another component of the same type or removing a specific one.

- New devices and contents could be considered such as AR headsets and contents.

- An extensive user evaluation could be performed to validate the model from the point of view of the TV viewer. Performing such an evaluation is probably premature, as multi-device broadcast/broadband applications are still uncommon and few users are familiar with them. When these types of applications are commonplace, it will be easier to select a user base to validate the presented methodology.

- The role of the context should be analysed more in detail and parameters of the three defined parts i.e. user context, physical context and system context, could be taken into account and see the way they could affect to the adaptation process.
- The analysis of the context could lead to learning processes that allow for modifying or refeeding the model with context information: the interaction of the viewers in general, personalisation for each viewer, analysing how the environmental factors impact the adaptation preferences, etc. In this way a complete context service could be built on top of our system to boost sophisticated smart environments.
- More fields of applications could be explored apart from the ones presented in Chapter 5. For instance, education could be a field in which multi-device environments will be used. Nowadays tablets and laptops are usually found at schools and students visualize the lessons and do their exercises by using them. Therefore, the work presented in this thesis could contribute to generate a service in which the teacher can manage different contents and share them with the students in a more sophisticated way.

Part V

Appendix

APPENDIX

A

Other Publications

List of other publications:

A.1 Web-based Video-Assisted Point Cloud Annotation for ADAS validation

Title: Web-based Video-Assisted Point Cloud Annotation for ADAS validation
Authors: Mujika, Andoni and Fanlo, Ana Dominguez and Tamayo, Iñigo and Senderos, Orti and Barandiaran, Javier and Aranjuelo, Nerea and Nieto, Marcos and Otaegui, Oihana
Proceedings: The 24th International Conference on 3D Web Technology
Pages: 1–9
Organization: ACM
Year: 2019
DOI: `https://doi.org/10.1145/3329714.3338128`
Abstract: *This paper introduces a web application for point cloud annotation that is used in the advanced driver assistance systems field. Apart from the point cloud viewer, the web tool has an object viewer and a timeline to define the attributes of the annotations and a video viewer to validate the point cloud annotations with the corresponding*

video images. The paper also describes several strategies we followed to obtain annotations correctly and quickly: (i) memory management and rendering of large-scale point clouds, (ii) coherent combination of video images and annotations, (iii) content synchronization in all parts of the application and (iv) automatic annotation before and during the annotation task of the user.

APPENDIX

B

Curriculum Vitae

Ana Dominguez is with the Department of Digital Media, Vicomtech. She received her Telecommunication Engineering degree in 2013 and her Telecommunication advanced degree in 2014, both from Tecnun, Universidad de Navarra (Spain). She focuses on the research lines related to multimedia services and interactive media technologies. She developed at Vicomtech her End of Degree Project developing web components based on Web Components standard for their subsequent integration on multi-device multimedia applications. Since 2015 she is working at Vicomtech, where she designs and develops projects of the Digital Television and Multimedia Services area.

In 2019 she has been an associate professor in Deusto Bussiness School in the field of Media Technologies. More in detail, she has given lectures in the double degree MBA (Master in Bussiness Administration) & Computer Engineering as well as in the Communication degree.

Part VI

Bibliography

Bibliography

[aja] AJAX definition. http://adaptivepath.org/ideas/ajax-new-approach-web-applications/. [Online; accessed 12-September-2017]. 20

[Altaboli and Lin11] Ahamed Altaboli and Yingzi Lin. Investigating effects of screen layout elements on interface and screen design aesthetics. *Advances in Human-Computer Interaction*, 2011:5, 2011. 6

[ana] Google Analytics Website. https://analytics.google.com/. [Online; accessed 12-September-2017]. 46

[Armstrong et al.14] Mike Armstrong, Matthew Brooks, Anthony Churnside, Michael Evans, Frank Melchior, and Matthew Shotton. Object-based broadcasting-curation, responsiveness and user experience. 2014. 28, 66

[ATSa] Advanced Television Systems Commitee (ATSC) Website. http://atsc.org/standards/atsc-standards/. [Online; accessed 12-September-2017]. 21

[ATSb] ATSC 3.0 standard. http://atsc.org/standards/atsc-3-0-standards/. [Online; accessed 12-September-2017]. 21

[aws] Amazon Web Services (AWS) Website. https://aws.amazon.com/. [Online; accessed 12-September-2017]. 42

[Bauerly and Liu08] Michael Bauerly and Yili Liu. Effects of symmetry and number of compositional elements on interface and design aesthetics. *Intl. Journal of Human–Computer Interaction*, 24(3):275–287, 2008. 6

[Baumann and Hasenpusch16] Sabine Baumann and Tim Hasenpusch. Multi-platform television and business models: A babylonian clutter of definitions and concepts. *Westminster Papers in Communication and Culture*, 11(1):85–102, 2016. 4

[bbc] Designing for second screens: The Autumnwatch Companion. `http://www.bbc.co.uk/blogs/researchanddevelopment/2011/04/the-autumnwatch-companion---de.shtml`. [Online; accessed 12-September-2017]. 28, 66

[Bishop06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. 100

[bro] Browscap database. `https://browscap.org/`. [Online; accessed 17-January-2019]. 94, 98

[Brudy et al.19] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 562. ACM, 2019. 28

[cat] Categorizr. `https://www.npmjs.com/package/categorizr`. [Online; accessed 17-January-2019]. 91

[cea] CE-HTML description. `https://ec.europa.eu/eip/ageing/standards/ict-and-communication/smart-tv/ce-html_en`. [Online; accessed 12-September-2017]. 20

[Cesar et al.08] Pablo Cesar, Dick CA Bulterman, and AJ Jansen. Usages of the secondary screen in an interactive television environment: Control, enrich, share, and transfer television content. In *Changing television environments*, pages 168–177. Springer, 2008. 5

[Chawla et al.02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 104

[Chen and Guestrin16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016. 104

[Chernock et al.16] Rich Chernock, David Gómez-Barquero, Jerry Whitaker, Sung-Ik Park, and Yiyan Wu. Atsc 3.0 next generation digital tv standard—an overview and preview of the issue. *IEEE Transactions on Broadcasting*, 62(1):154–158, 2016. 21

[Chollet et al.15] François Chollet et al. Keras. `https://keras.io`, 2015. [Online; accessed 17-January-2019]. 97

[chr] Chrome Website. `https://www.google.com/chrome/browser/desktop/index.html`. [Online; accessed 12-September-2017]. 55

[Claudy12] Lynn Claudy. The broadcast empire strikes back. *IEEE Spectrum*, 49(12), 2012. 3, 4, 34

[css] Cascading Style Sheets standard. `https://www.w3.org/Style/CSS/Overview.en.html`. [Online; accessed 12-September-2017]. 20

[cus] Custom elements specitication. `https://w3c.github.io/webcomponents/spec/custom/`. [Online; accessed 12-September-2017]. 25, 55

[D21] MEDIASCAPE project deliverable “Usage scenarios and requirements”. `http://mediascapeproject.eu/files/D2.1.pdf`. [Online; accessed 12-September-2017]. 37

[DAE] OIPF-DAE specification. `http://www.oipf.tv/docs/OIPF-T1-R2_Specification-Volume-5-Declarative-Application-Environment-v2_3-2014-01-24.pdf`. [Online; accessed 12-September-2017]. 19

[det] detectizr.js. `https://github.com/barisaydinoglu/Detectizr`. [Online; accessed 17-January-2019]. 91

[dom] Document Object Model standard. `https://www.w3.org/DOM/`. [Online; accessed 12-September-2017]. 20

[Domínguez et al.17] A Domínguez, M Agirre, J Flórez, A Lafuente, I Tamayo, and M Zorrilla. Deployment of a hybrid broadcast-internet multi-device service for a live tv programme. *IEEE Transactions on Broadcasting*, 2017. 62, 65, 79, 89, 118, 124, 125, 127, 129

[Dominguez et al.18] Ana Dominguez, Iñigo Tamayo, Mikel Zorrilla, Julian Florez, and Alberto Lafuente. Componentizing a hybrid broadcast-internet multi-device media service. In *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6. IEEE, 2018. 62, 68, 84, 119

[Dominguez et al.19] Ana Dominguez, Iñigo Tamayo, Mikel Zorrilla, Julian Florez, Alberto Lafuente, and Stefano Masneri. Methods for device characterisation in media services. In *Proceedings of ACM TVX conference (TVX'19)*, pages 1–6. ACM, 2019. 62, 90

[Dong et al.16] Tao Dong, Elizabeth F Churchill, and Jeffrey Nichols. Understanding the challenges of designing and developing multi-device experiences. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, pages 62–72. ACM, 2016. 28

[Drira et al.07] Amine Drira, Henri Pierreval, and Sonia Hajri-Gabouj. Facility layout problems: A survey. *Annual reviews in control*, 31(2):255–267, 2007. 76

[DVB] ETSI Website - DVB standards. `http://www.etsi.org/technologies-clusters/technologies/broadcast/dvb`. [Online; accessed 12-September-2017]. 20

[Eastman and Ferguson12] Susan Tyler Eastman and Douglas A Ferguson. *Media programming: Strategies and practices*. Cengage Learning, 2012. 3

[ebu] EBU Website. `https://www.ebu.ch/home`. [Online; accessed 12-September-2017]. 85

[eria] ERICSSON CONSUMERLAB, TV and Media 2016 presentation. `https://www.ericsson.com/assets/local/networked-society/consumerlab/reports/tv-media-2016-presentation-ericsson-consumerlab.pdf`. [Online; accessed 12-September-2017]. 18

[erib] ERICSSON CONSUMERLAB, TV and Media 2017 presentation. https://www.ericsson.com/assets/local/careers/media/ericsson_consumerlab_tv_media_report.pdf. [Online; accessed 7-November-2019]. 18

[fal] Browsers false positives. http://www.stucox.com/blog/you-cant-detect-a-touchscreen/. [Online; accessed 17-January-2019]. 90

[fea] Feature.js. http://featurejs.com/. [Online; accessed 17-January-2019]. 90

[Frosini and Paternò14] Luca Frosini and Fabio Paternò. User interface distribution in multi-device and multi-user environments with dynamically migrating engines. In *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems*, pages 55–64. ACM, 2014. 29

[Funk et al.16] Markus Funk, Thomas Kosch, Romina Kettner, Oliver Korn, and Albrecht Schmidt. motioneap: An overview of 4 years of combining industrial assembly with augmented reality for industry 4.0. In *Proceedings of the 16th international conference on knowledge technologies and datadriven business*, page 4, 2016. 132

[Geerts et al.08] David Geerts, Pablo Cesar, and Dick Bulterman. The implications of program genres for the design of social television systems. In *Proceedings of the 1st international conference on Designing interactive user experiences for TV and video*, pages 71–80. ACM, 2008. 84

[Gers et al.99] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999. 105

[Goodfellow et al.16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 97

[Grubert et al.16] Jens Grubert, Matthias Kranz, and Aaron Quigley. Challenges in mobile multi-device ecosystems. *mUX: The Journal of Mobile User Experience*, 5(1):5, 2016. 28

[Hart and Staveland88] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988. 123

[hbba] HbbTV 1.0 specification. http://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.01.01_60/ts_102796v010101p.pdf. [Online; accessed 12-September-2017]. 19

[hbbb] HbbTV 1.5 specification. https://www.hbbtv.org/wp-content/uploads/2015/07/HbbTV-specification-1-5.pdf. [Online; accessed 12-September-2017]. 19

[hbbc] HbbTV 2.0 specification. https://www.hbbtv.org/wp-content/uploads/2015/07/HbbTV_specification_2_0.pdf. [Online; accessed 12-September-2017]. 19

[hbbd] Hybrid broadcast broadband TV (HbbTV) Website. http://hbbtv.org/. [Online; accessed 12-September-2017]. 4, 19

[Horak et al.19] Tom Horak, Andreas Mathisen, Clemens N Klokmose, Raimund Dachselt, and Niklas Elmqvist. Vistribute: Distributing interactive visualizations in dynamic multi-device setups. In *Proceedings of the ACM Conference on Human Factors in Computing Systems. ACM, New York, NY, USA.(conditionally accepted)*, 2019. 29

[Hosmer Jr et al.13] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013. 96

[htma] HTML Imports specitication. https://w3c.github.io/webcomponents/spec/imports/. [Online; accessed 12-September-2017]. 26, 55

[htmb] HTML Templates specitication. https://www.w3.org/TR/html-templates/. [Online; accessed 12-September-2017]. 26, 55

[htmc] Hypertext Markup Language standard. https://www.w3.org/html/. [Online; accessed 12-September-2017]. 20

[Husmann et al.17] Maria Husmann, Daniel Huguenin, Matthias Geel, and Moira C Norrie. Orchestrating multi-device presentations with omnipresent. In *Proceedings of the 6th ACM International Symposium on Pervasive Displays*, page 3. ACM, 2017. 29

[Hussain et al.18] Jamil Hussain, Anees Ul Hassan, Hafiz Syed Muhammad Bilal, Rahman Ali, Muhammad Afzal, Shujaat Hussain, Jaehun Bang, Oresti Banos, and Sungyoung Lee. Model-based adaptive user interface based on context and user experience evaluation. *Journal on Multimodal User Interfaces*, 12(1):1–16, 2018. 116

[iec] International Electrotechnical Commission (IEC) Website. `http://www.iec.ch/`. [Online; accessed 12-September-2017]. 21

[ipt] DVB-IPTV. `https://www.dvb.org/resources/public/factsheets/DVB-IPTV_Factsheet.pdf`. [Online; accessed 12-September-2017]. 4

[iso] International Organization for Standardization (ISO) Website. `https://www.iso.org/home.html`. [Online; accessed 12-September-2017]. 21

[Jokela et al.15] Tero Jokela, Jarno Ojala, and Thomas Olsson. A diary study on combining multiple information devices in everyday activities and tasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3903–3912. ACM, 2015. 28

[js] JavaScript Web API of W3C. `https://www.w3.org/standards/webdesign/script`. [Online; accessed 12-September-2017]. 20

[jsd] jsDelivr. `https://www.jsdelivr.com/`. [Online; accessed 17-January-2019]. 91

[Kingma and Ba14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 97

[Klokmose et al.15] Clemens N Klokmose, James R Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. Webstrates: shareable dynamic media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 280–290. ACM, 2015. 28

[Lee et al.97] Dik L Lee, Huei Chuang, and Kent Seamons. Document ranking and the vector-space model. *IEEE software*, 14(2):67–75, 1997. 94

[log] Logistic regression profiler code on GitHub. `https://github.com/tv-vicomtech/deviceCharacterisationMethods/tree/master/logRegProfiler`. [Online; accessed 21-March-2019]. 96

[Maisonneuve et al.09] Julien Maisonneuve, Muriel Deschanel, Juergen Heiles, Wei Li, Hong Liu, Randy Sharpe, and Yiyan Wu. An overview of iptv standards development. *IEEE Transactions on Broadcasting*, 55(2):315–328, 2009. 4

[McGill et al.15] Mark McGill, John H Williamson, and Stephen A Brewster. A review of collocated multi-user tv. *Personal and Ubiquitous Computing*, 19(5-6):743–759, 2015. 3

[McNamara and Kirakowski06] Niamh McNamara and Jurek Kirakowski. Functionality, usability, and user experience: three areas of concern. *interactions*, 13(6):26–28, 2006. 6

[Med] MediaScape European project Website. `https://cordis.europa.eu/project/rcn/110876/es`. [Online; accessed 12-September-2017]. 4

[Merkel11] Klaus Merkel. Hybrid broadcast broadband tv, the new way to a comprehensive tv experience. In *Electronic Media Technology (CEMT), 2011 14th ITG Conference on*, pages 1–4. IEEE, 2011. 19

[mob] mobile-detect.js. `http://hgoebl.github.io/mobile-detect.js/`. [Online; accessed 17-January-2019]. 91

[moda] Model implementation in Github. `https://github.com/tv-vicomtech/adaptationModel`. [Online; accessed 24-June-2019]. 119

[modb] Modernizr. `https://modernizr.com/`. [Online; accessed 17-January-2019]. 90

[mota] Motion corporation Website. `http://www.motioncorporation.com`. [Online; accessed 12-September-2017]. 40

[motb] Shared Motion - Multi-device timing for the web. `https://webtiming.github.io/timingsrc/doc/shared_motion.html`. [Online; accessed 12-September-2017]. 40

[Nagel15] Wolfram Nagel. *Multiscreen UX Design: Developing for a Multitude of Devices*. Morgan Kaufmann, 2015. 89, 98

[Nebeling and Dey16] Michael Nebeling and Anind K Dey. Xdbrowser: User-defined cross-device web page designs. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5494–5505. ACM, 2016. 28

[Nebeling17] Michael Nebeling. Xdbrowser 2.0: Semi-automatic generation of cross-device interfaces. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 4574–4584. ACM, 2017. 28, 29

[nem] NEM (Networked & Electronic Media) Connected TV position paper (December 2012). `https://nem-initiative.org/wp-content/uploads/2013/12/NEM-PP-015.pdf`. [Online; accessed 12-September-2017]. 5, 9

[neu] Neural profiler code on GitHub. `https://github.com/tv-vicomtech/deviceCharacterisationMethods/tree/master/neuralProfiler`. [Online; accessed 21-March-2019]. 97

[nie] NIELSEN, "The nielsen total audience report: Q3 2018". `https://www.nielsen.com/us/en/insights/report/2019/q3-2018-total-audience-report/`. [Online; accessed 7-November-2019]. 18

[Obrist et al.15] Marianna Obrist, Pablo Cesar, and Santosh Basapur. Forward to the theme issue on interactive experiences for television and online video, 2015. 3

[oip] Open IPTV Forum (OIPF) Website. `http://www.oipf.tv/`. [Online; accessed 12-September-2017]. 19

[opa] HbbTV OpApp specification. `https://www.etsi.org/deliver/etsi_ts/103600_103699/103606/01.01.01_60/ts_103606v010101p.pdf`. [Online; accessed 7-November-2019]. 21

[ope] Opera Website. `https://www.google.com/chrome/browser/desktop/index.html`. [Online; accessed 12-September-2017]. 56

[Oulasvirta17] Antti Oulasvirta. User interface design with combinatorial optimization. *Computer*, 50(1):40–47, 2017. 27

[Park et al.18] Seonwook Park, Christoph Gebhardt, Roman Rädle, Anna Maria Feit, Hana Vrzakova, Niraj Ramesh Dayama, Hui-Shyong Yeo, Clemens N Klokmose, Aaron Quigley, Antti Oulasvirta, et al. Adam: Adapting multi-user interfaces for collaborative environments in real-time. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 184. ACM, 2018. 29

[Paternò and Santoro12] Fabio Paternò and Carmen Santoro. A logical framework for multi-device user interfaces. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 45–50. ACM, 2012. 40

[Rädle et al.14] Roman Rädle, Hans-Christian Jetter, Nicolai Marquardt, Harald Reiterer, and Yvonne Rogers. Huddlelamp: Spatially-aware mobile displays for ad-hoc around-the-table collaboration. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, pages 45–54. ACM, 2014. 29

[Romero et al.16] David Romero, Peter Bernus, Ovidiu Noran, Johan Stahre, and Åsa Fast-Berglund. The operator 4.0: human cyber-physical systems & adaptive automation towards human-automation symbiosis work systems. In *IFIP international conference on advances in production management systems*, pages 677–686. Springer, 2016. 132

[Santosa and Wigdor13] Stephanie Santosa and Daniel Wigdor. A field study of multi-device workflows in distributed workspaces. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 63–72. ACM, 2013. 28

[Sarkis et al.18] Mira Sarkis, Cyril Concolato, and Jean-Claude Dufourd. A multi-screen refactoring system for video-centric web applications. *Multimedia Tools and Applications*, 77(2):1943–1970, 2018. 28, 66

[Savage15] Taylor Savage. Componentizing the web. *Communications of the ACM*, 58(11):55–61, 2015. 9, 66

[Schreiner et al.15] Mario Schreiner, Roman Rädle, Hans-Christian Jetter, and Harald Reiterer. Connichiwa: a framework for cross-device web applications. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2163–2168. ACM, 2015. 29

[Sears93] Andrew Sears. Layout appropriateness: A metric for evaluating user interface widget layout. *IEEE Transactions on Software Engineering*, 19(7):707–719, 1993. 27, 72

[sha] Shadow DOM specitication. `https://w3c.github.io/webcomponents/spec/shadow/`. [Online; accessed 12-September-2017]. 26, 55

[Singh and Sharma06] Surya P Singh and Renduchintala RK Sharma. A review of different approaches to the facility layout problems. *The International Journal of Advanced Manufacturing Technology*, 30(5-6):425–433, 2006. 76

[Soares et al.09] Luiz Fernando Gomes Soares, Romualdo MR Costa, Marcio Ferreira Moreno, and Marcelo F Moreno. Multiple exhibition devices in dtv systems. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 281–290. ACM, 2009. 18, 19

[Soares et al.10] Luiz Fernando Gomes Soares, Marcio Ferreira Moreno, Carlos De Salles Soares Neto, and Marcelo Ferreira Moreno. Ginga-ncl: declarative middleware for multimedia iptv services. *IEEE Communications Magazine*, 48(6):74–81, 2010. 19

[Sodagar11] Iraj Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia*, 18(4):62–67, 2011. 21

[staa] Audiences in France. `https://en.wikipedia.org/wiki/Television_in_France`. [Online; accessed 12-September-2017]. 84

[stab] Audiences in Germany. `https://www.statista.com/statistics/380528/tv-channels-audience-market-share-germany/`. [Online; accessed 12-September-2017]. 84

[stac] Audiences in Italy. `https://en.wikipedia.org/wiki/Television_in_Italy#Most_important_all_national_free-to-view_channels`. [Online; accessed 12-September-2017]. 84

[stad] Audiences in Spain. `https://www.statista.com/statistics/447203/leading-television-channels-in-spain-by-audience-share/`. [Online; accessed 12-September-2017]. 84

[stae] Audiences in UK. https://www.statista.com/statistics/269983/leading-tv-broadcasters-in-the-uk-by-audience-share/. [Online; accessed 12-September-2017]. 84

[staf] Statista WebSite. https://www.statista.com/statistics/461561/smart-tv-shipments-worldwide-by-region/. [Online; accessed 7-November-2019]. 18

[Techrunch13] Techrunch. Google Believes Web Components Are The Future Of Web Development. http://tcrn.ch/2a1ZZpN, 2013. 26

[Todi et al.16] Kashyap Todi, Daryl Weir, and Antti Oulasvirta. Sketchplore: Sketch and explore with a layout optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, pages 543–555. ACM, 2016. 27

[uag] UA dataset on GitHub. https://github.com/tv-vicomtech/deviceCharacterisationMethods/tree/master/UA. [Online; accessed 21-March-2019]. 91, 98, 99

[uap] ua-parser-js. https://github.com/faisalman/ua-parser-js. [Online; accessed 17-January-2019]. 91

[van Deventer et al.16] M Oskar van Deventer, Hans Stokking, Matt Hammond, Jean Le Feuvre, and Pablo Cesar. Standards for multi-stream and multi-device media synchronization. *IEEE Communications Magazine*, 54(3):16–21, 2016. 40

[Vatavu and Mancas15] Radu-Daniel Vatavu and Matei Mancas. Evaluating visual attention for multi-screen television: measures, toolkit, and experimental findings. *Personal and Ubiquitous Computing*, 19(5-6):781–801, 2015. 5

[vim] Vicomtech's Vimeo Website. https://vimeo.com/184664494. [Online; accessed 12-September-2017]. xiii, 38

[vsm] VSM profiler code on GitHub. https://github.com/tv-vicomtech/deviceCharacterisationMethods/tree/master/vsmProfiler. [Online; accessed 21-March-2019]. 95

[w3c] World Wide Web Consortium (W3C) Website. https://www.w3.org/. [Online; accessed 12-September-2017]. 19

[W3C Editor's Draft14] W3C Editor's Draft. HTML 5.1 Nightly. A vocabulary and associated APIs for HTML and XHTML. http://www.w3.org/html/wg/drafts/html/master/embedded-content.html#the-source-element-when-used-with-the-picture-element, 2014. 24

[W3C Recommendation10] W3C Recommendation. Mobile Web Application Best Practices. http://www.w3.org/TR/mwabp/, 2010. 25

[W3C Recommendation12] W3C Recommendation. Media Queries. http://www.w3.org/TR/css3-mediaqueries/, 2012. 24

[W3C Working Draf 14] W3C Working Draf . CSS Flexible Box Layout Module Level 1. http://www.w3.org/TR/css3-flexbox/, 2014. 25

[W3C Working Draft14a] W3C Working Draft. CSS Grid Layout Module Level 1. http://www.w3.org/TR/css3-grid-layout/, 2014. 25

[W3C Working Draft14b] W3C Working Draft. CSS Regions Module Level 1. http://www.w3.org/TR/css3-flexbox/, 2014. 25

[Wäljas et al.10] Minna Wäljas, Katarina Segerståhl, Kaisa Väänänen-Vainio-Mattila, and Harri Oinas-Kukkonen. Cross-platform service user experience: a field study and an initial framework. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 219–228. ACM, 2010. 28

[wcp] Web Components polyfill on Github. https://github.com/WebComponents/webcomponentsjs. [Online; accessed 12-September-2017]. 56

[web] Web Components W3C specification. http://w3c.github.io/webcomponents/. [Online; accessed 12-September-2017]. 25, 28, 41, 66

[Werbos75] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University, 1975. 97

[Yang and Wigdor14] Jishuo Yang and Daniel Wigdor. Panelrama: enabling easy specification of cross-device web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2783–2792. ACM, 2014. 29

[Zhang et al.14] Wenjun Zhang, Yiyan Wu, Namho Hur, Tetsuomi Ikeda, and Pingjian Xia. Fobtv: Worldwide efforts in developing next-generation broadcasting system. *IEEE Transactions on Broadcasting*, 60(2):154–159, 2014. 18

[Ziegler13] Christoph Ziegler. Second screen for hbbtv—automatic application launch and app-to-app communication enabling novel tv programme related second-screen scenarios. In *Consumer Electronics¿ Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on*, pages 1–5. IEEE, 2013. 40

[Zorrilla et al.13] Mikel Zorrilla, Iñigo Tamayo, Angel Martin, and Igor G Olaizola. Cloud session maintenance to synchronise hbbtv applications and home network devices. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2013 IEEE International Symposium on*, pages 1–6. IEEE, 2013. 40, 62

[Zorrilla et al.14] Mikel Zorrilla, Angel Martin, Inigo Tamayo, Sean O'Halpin, and Dominique Hazael-Massieux. Reaching devices around an hbbtv television. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2014 IEEE International Symposium on*, pages 1–7. IEEE, 2014. 40

[Zorrilla et al.15a] Mikel Zorrilla, Njål Borch, François Daoust, Alexander Erk, Julián Flórez, and Alberto Lafuente. A web-based distributed architecture for multi-device adaptation in media applications. *Personal and Ubiquitous Computing*, 19(5-6):803–820, 2015. xiii, 4, 5, 6, 7, 27, 28, 37, 38, 40, 58, 62, 79, 147

[Zorrilla et al.15b] Mikel Zorrilla, Iñigo Tamayo, Angel Martin, and Ana Dominguez. User interface adaptation for multi-device web-based media applications. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2015 IEEE International Symposium on*, pages 1–7. IEEE, 2015. 40, 105

[Zorrilla16] Mikel Zorrilla. *Interoperable technologies for multi-device media services*. PhD thesis, UPV/EHU University, 2016. 4, 7, 21