A Web-based distributed architecture for multi-device adaptation in media applications

Mikel Zorrilla[1], Njål Borch[2], François Daoust[3],
Alexander Erk[4], Julián Flórez[1], Alberto Lafuente[5]

[1]**Vicomtech-IK4**, Mikeletegi pasealekua 57, San Sebastian-Donostia (Spain)
[2]**Norut**, P.O. Box 6434 Tromsø Science Park, Tromsø (Norway)
[3]**W3C/ERCIM**, Route des Lucioles 2004, Sophia Antipolis (France)
[4]**Institut für Rundfunktechnik**, Floriansmuehlstrasse 60, Munich (Germany)
[5]**UPV/EHU**, Lardizabal Pasealekua 1, San Sebastian-Donostia (Spain)

Mikel Zorrilla. Email: mzorrilla@vicomtech.org,
Phone: +34 943 30 92 30, Fax: +34 943 30 93 93

Abstract. HTML5 is driving a strong trend towards interoperable Web-based applications, enabling a wider range of devices to run this kind of applications. The key challenge of next generation media applications is to federate cooperative devices to provide multi-device experiences overcoming current second screen solutions within the Connected TV industry. There is a gap on the experience of users, since they perceive devices as isolated pieces of applications when they would prefer to have a single experience through multiple devices at the same time. This paper proposes a unified methodology and a common specification over Web Components for the adaptation of a single application, seamlessly running different instances on one or more devices simultaneously, according to the multi-device context of the user and the specific features of the devices. The solution presented in this paper extends current Web standards towards an interoperable architecture, and offers broadcasters and media application developers the possibility to easily design applications that will automatically provide a unique consistent experience across the connected devices. The architectural design is targeted to be included in the roadmap of the standards.

Keywords: Multi-device adaptation, responsive adaptive user interface, ubiquitous computing, pervasive computing, media-driven Web application, broadcasting

1 Introduction

We are currently witnessing a strong tide to powerful Web-based applications. This trend is also driving the evolution of HTML5, making a wider range of devices capable of running applications that gain features previously available only through native SDKs [1].

Once the self-capacity and interoperability are being addressed by HTML5, the key challenge for next generation applications is to federate cooperative devices to provide users coherent multi-device experiences. This is a natural step in the adaption of the market and society to the growing behaviour of users accessing services from several devices simultaneously [2]. However, most of the currently existing applications are running on devices independently or at best loosely coupled, sharing a backend on an event-driven model. Therefore, a gap on the experience of the users is produced, as they perceive devices as isolated pieces of applications when they could have a single experience through multiple devices at the same time.

Despite the continuous proliferation of smart devices, the TV set is still the prominent device to consume media services. Nonetheless, televisions are increasingly being combined with tablets, smartphones or laptops, all connected to the Internet. The growing interest in second screen solutions within the Connected TV industry [3] clearly shows that users expect a more consistent experience across different devices and their applications [4-5].

In order to provide multi-device applications, broadcasters and media application developers currently need to implement, distribute and maintain a set of rather complex technical solutions tailored to each of the specific target platforms. For a second screen application including a TV and a mobile device, [6] and [7] present an event-driven approach. They propose a solution with two different applications, one based on HbbTV [8] on the TV and an HTML-based application for the mobile, as well as a server to handle the communication between applications. The need to implement customised applications for each platform, and the necessity to consider a coherent multi-device visualisation during the development phase, induces the development itself, test and maintenance of service-related applications to be inherently complicated.

Current solutions illustrate a fragmented market with vendor-specific, and often application-specific approaches. For Connected TV service provision (HbbTV, YouView, Android based platforms, proprietary technologies such as Samsung, LG, Philips, Sony, etc.) completely different developments have to be created and maintained in order to be compatible with all approaches and brands. So far, these multi-device applications are mainly restricted to a specific TV set and a second screen device (e.g. an Apple TV with an iPhone or iPad, or a Samsung TV with Samsung mobiles) or are fully disconnected, such as using Twitter hashtags.

Despite the clear opportunities to catalyse new business models, the lack of standards prevents the creation of seamlessly connected, intuitively converged and conveniently continuous experiences across a heterogeneous ecosystem of devices. The priority for future research is shifting towards fully manageable, context-sensitive and controllable or self-regulating multi-device applications. Moreover, considering the distributed nature of the devices that access the services, some additional capabilities are necessary, such as autonomous information exchange or triggering actions.

In order to tackle the contextualised limitations, several scientific challenges need to be overcome, such as the device and service discovery, the cross-platform user authentication, the multi-device context and content synchronisation, and the cross-device application adaptation. The work presented in this paper is part of the research

activities that are being performed in the MediaScape European project[1] that addresses the aforementioned challenges. However, this paper is focused on a deep analysis of the challenges for multi-device adaptation and provides a solution for the adaptation domain that enables openness, expressiveness and autonomous behaviour.

Facing the multi-device adaptation challenge means to evolve from developing different applications for specific target devices, controlling one by one their functionality, their interfaces and creating specific event-driven mechanisms for inter-device communication, to a single application where developers describe the complete functionality, interface and the behaviour of the application for a multi-device environment once. The seamless translation of a single application into a multi-device execution while still providing a well fitted portion of the application on each device, is the multi-device adaptation challenge we aim at addressing in the paper.

The main contribution of the work described in this article is the design and implementation of a distributed adaptation architecture for media-driven multi-device applications. We provide a distributed architecture where the application developers create a Web application in terms of logic components once and add properties to define the multi-device behaviour of the application on a high abstraction level. Thus, during the development phase, the application developers do not need to be aware of a particular multi-device context or define views for a specific set of devices (TVs, radios, mobiles, laptops, etc.). Developers have to define global hints to allow the architecture decide the best configuration for visualisation through a dynamic set of involved devices. The libraries and tools comprised in our approach are responsible for analysing and managing the multi-device context of the application and the properties defined in the application to: a) decide which logic components are presented on each device on a specific moment, b) create a responsive layout, seamlessly adapted to each device, and c) generate a seamless inter-device communication channel. The user would then be able to smoothly move parts of the application from one device to another in an intuitive manner, and the application would self-adapt to each device.

The methodology followed to achieve the expected contributions is reflected across the different sections of this paper. In Section 2, the Hybrid TV ecosystem is analysed in order to foresee the experience that next generation multi-device media applications could offer to users. The related work is described in Section 3, including an analysis of broadcast-related specifications to add companion devices to the TV, specifications for the description of ubiquitous media presentations, and Web mechanisms to enable adaptive application. In Section 4, the paper focuses on the definition of the adaptation challenges, exploring different scenarios extracted as subsets of the general overview depicted in Section 2. Section 5 presents the core contribution of the paper. The section presents a general architectural vision, followed by the definition of specific adaptation design objectives. The section also provides details about the architectural design of the proposed distributed adaptation solution and its advancement beyond the presented related work. Section 6 presents a validation of the achievement

[1] http://mediascapeproject.eu

of the design objectives by means of a proof-of-concept implementation of the proposed architecture. Finally, conclusions are stated on Section 7.

2 The Hybrid TV ecosystem

This section provides pivotal conclusions, extracted from market surveys and reports, to be the base for the definition of a general overview of the Hybrid TV and multi-device experiences.

Hundreds of apps in the various market places aid consumers in controlling the first screen, discovering new content, enhancing their viewing experience and enabling a shared, social experience. These Hybrid TV and multi-device applications are completely oriented to a target device and lack a real interoperability between devices. As mentioned before and addressed in the following section, there are limitations on the standards and technologies in the current state of the art to allow broadcasters and Web application developers to create this kind of application with universal compatibility in a sustainable way.

Due to the success of the Smart TV market there is an increasing demand of multi-device applications. As many HCI researchers increasingly point out, Mark Weiser's vision of ubiquitous computing [9] will be applied based on multiple devices (tablets, smartphone, TVs, PCs, etc.). The real power of the concept comes from the interaction between devices and sharing experiences with other users.

In order to gain leverage from ubiquitous computing to multi-device applications, it is essential to advance in four areas that span both application developer needs and user expectations. The areas are as follows:

1. security, addressing authentication on multiple devices [10] and privacy of data transferred during the shared experiences;
2. discovery and pairing, associating the available resources: users, devices and services [6-7, 11-13];
3. synchronisation, providing hybrid broadcast-Internet media synchronisation [14] and deploying a shared context available immediately to all participants to provide session awareness and shared timing information for media synchronisation [15-17]; and
4. adaptation, spanning autonomous creation of adaptive interfaces suitable to each concurrent viewer, and automatic interface arrangement tracking dynamic application logic.

From all the research areas required to deliver a secure, connected, interoperable and continuous experience over a wide range of devices, this paper focuses on the adaptation domain.

To fully understand the scope of our research and its challenges, the following list summarises the considered contextual factors regarding the trend of media consumption habits of users together with market trends. This is reinforced and aligned with the general overview foreseen by [3] and the accumulated experience from first level broadcasters and technology providers: BBC (British Broadcasting Corporation), BR

(Bayerischer Rundfunk), IRT (Institut für Rundfunktechnik), NEC Europe (Nippon Electric Company), Web standardisation bodies like W3C (World Wide Web Consortium) and research institutes like Norut and Vicomtech-IK4 (Visual Interaction & Communication Technologies) [18]:

1. Broadcast capable devices are highly relevant for media applications. TV and radio are still the most common devices to consume media services. In 2014 TV dominates the daily media consumption, where most of the content was delivered to the TV set in a traditional manner: over broadcast, cable, satellite or telecommunication connection [5, 19]. At the same time, radio reaches all people age 12+ during the week [20]. Radio and TV are in a good position to capture the attention of the audience and drive their interests to specific contents or services.
2. Broadcast capable devices are increasingly connected to Internet. Global Smart TV shipments grew 55% year-over-year where around 50 percent of Smart TV owners across the USA and major European markets are currently using their TV's Internet capabilities [21]. Internet radio devices are still immature, evolving to be able to receive and play streamed media from internet radio stations and contents from the home network but often lack a real integration of broadband and broadcast services.
3. Ubiquitous computing centred in the home is a relevant scenario for multi-device media services for end users, broadcasters and application developers. It also brings challenges of ubiquitous computing related to mobility of the users through different contexts and networks, different rooms, multi-member households, going out from home, continuing the experience when arriving at home, etc.
4. Users have other connected devices while watching TV in a home scenario. The consumption habits have changed completely with many TV viewers using tablets, smartphones or laptops while watching TV. According to a study of Ericsson ConsumerLab [22] social networking sites and forums are used by 59% of people while watching TV, 49% will use apps or browse the Internet to find out more about the content they are watching, while almost 1 in 3 will discuss what they are currently viewing over social networks or chats.
5. Users are often using two devices simultaneously, accessing to related information in one device to complete the content being watched in the other. Fuelled by the deep penetration rate of smartphones, tablets and laptops worldwide [23], 35% of first screen time is second screened, of which 1/3 is related to the main content [6]. Concerning second screen companion experiences, 63% of the consumers accessing synchronised content on the second screen say it makes them feel a better experience in the shows they are watching [24].
6. Social experiences relate to a media application. The relation of broadcasted contents and social networks is two way. On the one hand, broadcasted contents are the main driver of social activity at prime time engaging audience to share immediate reactions [25]. On the other hand, the social media has the ability to engage TV audiences in real-time and in the days that follow. Regarding second screen companion experiences, 14% of second screen time is related to social activity

about TV program [26], chat about a live broadcasted programme, suggesting content, share impressions, etc.

Once the focus of our work has been described, **Fig. 1** presents a diagram of the general overview of the hybrid TV and multi-device experiences. On the one hand, the TV and the radio devices have the broadcast capability to access media content. Moreover, broadcast-related specifications, such as HbbTV, allow the hybrid application signalling through the broadcast channel. On the other hand, the media devices at home, such as tablets, smartphones and laptops, are connected to the Internet. This includes the connected TVs and radios that have the hybrid broadcast-Internet capability creating a bridge between the two channels. Internet brings a way to access applications, content and delivery of application triggers. Accordingly, **Fig. 1** shows all the media devices at home inter-connected between them, offering the user a single experience through multiple displays. On the same way, the figure represents the home-centric *on the move* scenarios, where a user arrives at home with a smartphone and the application adapts to the home context. Finally, the icons on the right side show other homes and users *on the move*, sharing a multi-user experience.

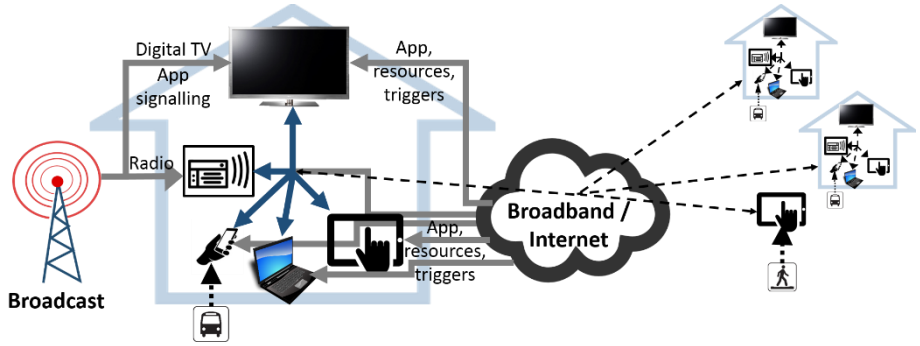

Fig. 1. Diagram of the general overview of the Hybrid TV and multi-device experiences

3 Related Work

This section provides a summary of the previous work in the field of adaptation for media-driven multi-device applications. It is focused on how the multi-device behaviour of the application is defined as well as how the adaptation is done for a set of devices.

Broadcast-related specifications to add companion devices to the TV

There are broadcast-related standards 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) [27] 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 [28]. 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. [27] 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 [8]. In February 2015, the 2.0 specification of HbbTV has been 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[2].

These broadcast-related standards, even though they allow mechanisms to launch applications on a companion screen or provide cooperative multi-device application, the application developer needs to create the application completely oriented to a 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.

Specifications for the description of ubiquitous media applications

Other works deal with the presentation of media applications, describing the different objects and their behaviour, such as SMIL [29], MPEG-4 [30] and MPEG-7 [31].

SMIL (Synchonized Multimedia Integration Language) is a W3C Recommendation that specifies an XML format to describe media presentations. It defines a declarative description for timing information to present media, animations, transitions between elements, etc. SMIL 3.0[3] defines the *MultiWindowLayout* module, which contains elements and attributes to describe multiple sets of regions, where multimedia content is presented. However, SMIL does not allow to specify an association between a set of regions and specific devices.

MPEG-4 Part 11 (Scene description and application engine), also known as BIFS (Binary Format for Scenes), defines a spatio-temporal representation of media objects

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

[3] SMIL 3.0. December 2008. http://www.w3.org/TR/SMIL/

as well as their behaviour in response to interaction. MPEG-4 Part 20 or LASeR (Lightweight Application Scene Representation) is a specification designed for representing and delivering rich-media services to resource-constrained devices such as mobile devices.

MPEG-7 is a multimedia content description standard to provide complementary functionality to the previous MPEG standards, representing information about the content in order to understand its behaviour and see the possibility of interaction with each of the objects and the relation among them.

However, there is a lack of implementations and solutions on top of SMIL, MPEG-4 and MPEG-7 specifications.

Adaptive Web applications and responsive design

There are also 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 [32]. 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*) [33]. These simple filters can be applied to CSS styles in order to change the properties based on characteristics of the end-device. Even for 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 [34] 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[4] boosts the dynamic grid layouts for drag-and-drop actions based on jQuery. CSS3 Flexbox [35] 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 [36] allows elements moving through different regions where CSS Regions library[5] expands the browser support with broader feature coverage. Grid-layout [37] defines a two-dimensional grid-based layout system, optimised for user interface design. The XML XUL[6] language has been created by Mozilla to design user interfaces. Kontx[7] is the proposal of Yahoo to develop TV apps. EnyoJS[8] is a framework to develop HTML5 apps focused on layouts and panels design. Bootstrap[9] 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.

W3C defined Web Components [38] to create a Web application in terms of functional components. In comparison with SMIL, it does not have a temporal description of the application but it is a promising emerging standard being widely included in the roadmap of the Web browsers. Web Components include: a) creation of custom elements that can be easily reused in different Web applications including its behaviour; b) import capability of HTML trees to recycle them in other Web applications; c) declaration of DOM (Document Object Model) subtrees as templates ready to be instantiated in the final DOM; and d) management of Shadow DOM, encapsulating and managing DOM trees for the final DOM composition. This high-level block definition from tags to subtrees empowers the recycling possibilities.

There are libraries, such as Polymer[10], that facilitate the development on top of Web Components specification. Furthermore, due to the still limited browser support of this technology, Polymer overcomes the lack of native support of the browser with polyfills[11], providing universal browser compatibility. The combination of Web Components and *media queries* boost the responsive design of Web applications and they are core technologies, among others, of the next generation of Google apps [39].

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

---

4 Gridster.js website. http://gridster.net/

5 CSS Regions library, April 2014. https://github.com/FremyCompany/css-regions-polyfill

6 Mozilla XUL. https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL

7 Kontx. https://developer.yahoo.com/connectedtv/kontxapiref/

8 EnyoJS. http://enyojs.com/

9 Bootstrap. http://getbootstrap.com/

10 Polymer library's website: https://www.polymer-project.org/

11 A polyfill is a downloadable code which provides facilities that are not built natively in a Web browser

Overview and outlook

It can be concluded that the described broadcast-related standards provide solutions to create a multi-device experience around a TV. Those standards are limited to a specific combination of devices, such as a TV and a smartphone. Therefore, the application developers need to define the visualisation of the application on the first and second screen at the development stage.

In order to dynamically make decisions on how to split the functionality of a media application across devices, there are specifications with a spatio-temporal description of media presentation, such as SMIL or MPEG-4 Part 11. However, these specifications are not widely deployed.

This differs from the approach of Web Components, which enables reusability and expressiveness on top of HTML5 technologies. Moreover, broadcast-related standards such as HbbTV, are evolving towards HTML5, facilitating the broadcast-Internet convergence. In comparison with SMIL, Web Components lack spatio-temporal data to share timing information on a multi-device context, but Web technologies provide many mechanisms to synchronise state between components [15-17].

None of the standards that define interface adaptation and responsive design address applications that span multiple devices. Solutions are designed for the adaptation of an application to a single device, rather than providing the appropriate mechanisms to split the various parts of an application across different devices.

A more versatile solution is required in order to allow an application to self-adapt automatically to the multi-device context of the user at run-time. For that purpose, the application should be defined in terms of logic components and generic multi-device behaviour. Web Components provide that separation and reusability through components, while taking advantage of widely spread Web mechanisms, thus enabling a complete convergence with the broadcast-related standards, such as HbbTV.

However, the multi-device paradigm creates new challenges to be tackled and requirements to be defined. On the one hand, the components should share a state across the devices for the spatio-temporal and context synchronisation. On the other hand, the Web mechanisms for adaptive Web interfaces, as well as the responsive Web design patterns, need to be extended to the multi-device aspect.

4 Multi-device adaptation challenges

This section presents the multi-device adaptation challenges to be addressed in the proposed architecture. Those are the outcome of the analysis of multi-device scenarios and use cases extracted as subsets of the overview diagram (**Fig. 1**) described in Section 2. A more detailed description of the multi-device scenarios and use cases can be found in [18].

Scenario A - Hybrid Live TV Programme: As **Fig. 2** presents, some of the adaptation challenges can be analysed on a hybrid live TV programme where a broadcaster provides a live sports event. In addition to the 'standard' TV signal, the broadcaster is preparing an application with extra content, such as streaming cameras to follow top

athletes or specific spots, maps with geolocated data (e.g. taken from GPS) or real-time statistics.

From a user perspective, users watching sports events on TV can use their mobile devices to find services related to the programme. Some parts can be overlaid on the TV and shared by different households, while other parts can be placed on their mobile devices with more personalised information.

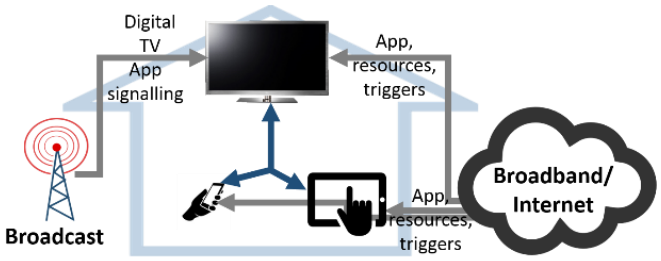

Fig. 2. Diagram for the Hybrid TV Programme

Scenario B - Multiuser quiz: Fig. 3 focuses on a sharing experience between two users, playing a quiz related to a TV programme while on the move. One of them is playing it using the smartphone through 3G, without following the live programme but with all the needed information to play. When arriving home, the TV seamlessly comes into action, providing live media content while the interactive parts remain on the smartphone. Later on, the user moves to the bedroom, only with the smartphone, but connected through Wi-Fi to the Internet. The application will provide a streaming live video together with the quiz. This scenario covers the companion experiences related to a TV programme, where the rhythm of the application can be synchronised with a live TV programme or with a non-linear experience having an independent timeline.

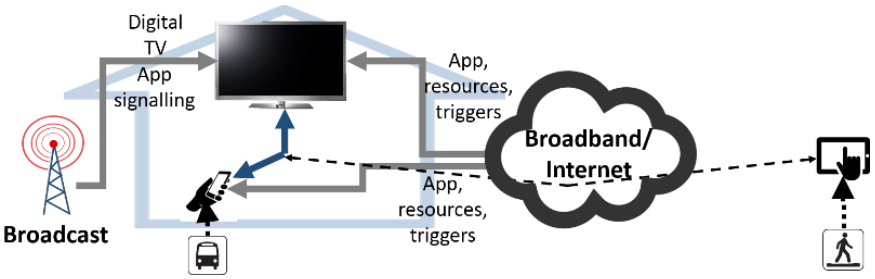

Fig. 3. Diagram for the Multiuser quiz

Scenario C - Hybrid Radio through a connected device: The scenario represented in **Fig. 4** shows a user controlling a radio through a laptop and adding bookmarks to contents for later access. In the same way it enables the user to transfer contents between devices, such as moving the audio from one radio to another or moving the controls from the laptop to a smartphone.

Fig. 4. Diagram for the Hybrid Radio through a connected device

Following the process described in [18], where scenarios A, B and C are deeply described together with different use cases extracted from those scenarios, we identified a set of challenges to be considered in the design of a multi-device adaptation architecture. [18] also provides an analysis of the scenarios across the main stakeholders involved in the scenarios: end users, broadcasters and application developers. In particular, four adaptation challenges have been defined:

1. **Multi-device adaptation**: The application needs to decide which pieces of content are suitable to be presented on each device, depending the contextual information of the multi-device dimension. Thus, it can provide the user a single experience through one or more devices at the same time. Multi-device context management is required, together with the knowledge of the capabilities of the different devices, so the application has contextual information to take decisions.
2. **User interface adaptation**: Once the multi-device adaptation decisions have been taken, the application has to be shown in a responsive way and adapted to the features of the device (screen size, interaction way, etc.).
3. **Personal and shared device adaptation**: The role of the device in multi-device experiences should be considered to decide which pieces of information are suitable for each device.
4. **Inter-components communication**: The components require a shared context, allowing all components in the application to share necessary state. The shared context must be distributed to all components, regardless of which device they are executing on. This will enable, for instance, a controller component to play or pause a video player, no matter if both components are in the same device or not, or to have a common timeline for different components across multiple devices.

Table 1 explores the particular singularities of the defined adaptation challenges in each scenario. **Multi-device adaptation** and **user interface adaptation** are the main challenges that appear in all situations. This is also true for the **inter-components communication** challenge, expressing the need for a flexible state exchange between logic components in order to guarantee the first two challenges. The **personal and shared device adaptation** challenge has been extracted from the **Hybrid Live TV Programme** scenario, as an important particular case of **multi-device adaptation**.

The performed analysis shows that the adaptation challenges are common to the different scenarios proposed in this paper. The election of these scenarios has not been arbitrary. On the one hand, the scenarios are *realistic* since the different stakeholders identified them as familiar. On the other hand, they provide a *complete* view of the ecosystem, as combining the different features depicted by **Fig. 2**, **Fig. 3** and

Fig. 4, the outcome is the general schema shown in **Fig. 1**. Moreover, the scenarios are *non-redundant*, being the three of them necessary to cover the general outline.

Table 1. Analysis of the singularity of the adaptation challenges in each scenario

A. Hybrid Live TV Programme
Multi-device adaptation
• Present the application or its parts depending on the multi-device dimension; TV or TV and mobiles. • Decide the amount of textual information presented on a TV, usually limited. • Avoid parts of the application that may not be suitable for a TV, because the constrained interaction capabilities of the remote control. • Re-structure parts of the application when mobiles are associated providing a richer interaction. • Allow user interaction to move parts of the app from one device to another, such as overlays on TV.
User interface adaptation
• Present pieces of the application on each device following the responsive design pattern. • Provide a good experience in terms of media consumption and interaction.
Personal and shared device adaptation
• Decide the adequate information to present depending the nature of the devices. • Information shared by the different users in front of the same TV. • Personal information on personal mobiles.
Inter-components communication
• Control the application in a single and consistent way through all the devices. • Different household members controlling the same TV.
B. Multiuser quiz
Multi-device adaptation
• Present essential interactive parts when there is only a single mobile device. • Add other components, such as the broadcasted video, when a TV companion joins. • Present streamed video content with high-speed Internet connection.
User interface adaptation
• Present portions of the application on each device. • Focus interactivity on the mobile and in media content on the TV.
Inter-components communication
• Communication between components being managed by a single user through different devices. • Communication through components handled by different users, providing a multi-user experience.
C. Hybrid Radio through a connected device
Multi-device adaptation
• Identify the best device for a specific activity, like the hybrid radio for a live radio programme. • Use devices with high interactivity capability to control the radio.
User interface adaptation
• Present the controls in the most appropriate way depending the device: embedded on a Web page on a laptop or full screen on a smartphone. • Provide a good interaction experience to select where to play the audio and how to control it.
Inter-components communication
• Control the radio from other devices, exchanging state between distributed components.

5 Web-based distributed adaptation architecture

The Web-based adaptation architecture is the main contribution of this paper, enabling the creation of responsive multi-device media applications that overcome the challenges identified in the adaptation domain. This solution is built over the Web Components standard and deployed on top of a distributed architecture, where each device takes its own adaptation decisions according to the shared context of the multi-device dimension. This section explains the architectural design decisions that have been taken and shows the architecture details focusing on the two main modules to solve the adaptation challenges: the multi-device adaptation engine, addressing the **multi-device adaptation** challenge, and the user interface adaptation engine, facing the **user interface adaptation** challenge.

Overview of the architecture

In this Subsection we overview the high-level architecture that has been designed for the development of multi-user and multi-device media centric applications within the MediaScape European project. This architecture aims to solve scientific challenges for such applications, which are wider than the adaptation challenges analysed in this article. Cross-device user authentication, device / user / service discovery and association, and application and content synchronisation are other main research topics together with adaptation in the project.

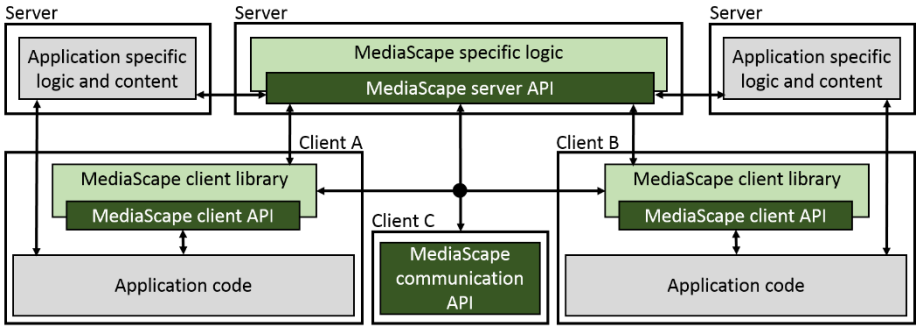

Fig. 5. System overview of the MediaScape architecture

The starting point of the high level architecture is a client/server approach. **Fig. 5** outlines the general architecture of MediaScape. On the top part of the figure, three server boxes are placed. The application developers, creating multi-device media applications with the MediaScape libraries, do not need technological conditions or requirements for the application specific logic and content on the server-side. This means that the application provider can therefore host the application just as other Web content. The MediaScape toolkit logic is hosted on a second server class (depicted in the top centre of **Fig. 5**), providing services to the different MediaScape modules, accessible through an API. The MediaScape services on the server-side are built

as modular and open services, making possible to be hosted by the same application provider or by a third party server.

Applications that are distributed to clients are built on top of standard Web technologies enabling the interoperability with other components of the system. A Web server provides the resources (HTML, JavaScript, and CSS) via HTTP to the application clients. These clients download the required application resources, containing application specific code and MediaScape client libraries. These libraries enable the communication with the required MediaScape services, making them accessible to the application code through a client side API.

Communication between two clients is usually performed through a common application server. Nevertheless direct communication protocols can be used between applications, such as WebRTC or Bluetooth. Hence, the figure shows a connection between client A and client B.

Additionally, a second client type is included in the architecture (client C). This represents clients, that are incapable of running client side components and therefore cannot benefit from the full MediaScape client library (e.g. the hybrid radio presented in the scenarios). These devices may expose their capabilities with standard interfaces, thus other full MediaScape clients could discover and include it into a multi-device application.

Design objectives

This section aims to translate the adaptation challenges exposed in Section 4 into architectural design objectives for adaptation purposes. The required adaptation modules will match the general high-level architecture presented previously.

As an initial design objective, the architecture must provide an **autonomous behaviour** for client devices, enabling a self-organising system in the operational phase. To achieve this, a distributed solution is needed. The alternative centralised design would require a server-side adaptation module orchestrating all the adaptation decisions and delivering what-to-do to client devices. A distributed solution, where the client devices share contextual information and make their own adaptation decisions according to the shared context, follows the Web design trend and removes the need of a dedicated server infrastructure. Distributing the decisions through all client devices removes the possible central server unavailability and assures an autonomous adaptation outcome. The out of date of a client device's shared context, due to any connectivity hazard, will result in a temporary sub-optimal outcome in the worst case.

Moreover, the proposed system has to boost the development of this kind of applications, avoiding broadcasters and application developers to deal with complex and heterogeneous environments, enabling to create multi-device applications by means of a single Web application that simplifies the scientific and technological challenges. In this regard, our approach needs to provide a flexible and modular adaptation architecture, enabling re-use of components and adaptation behaviour from one application to another. A single development environment, with a common set of programming languages based on standard Web technologies, will facilitate the creation of multi-device applications covering a wide range of scenarios. The goal is to avoid ad-hoc

developments for a specific adaptation server and for client devices. Thus, the implementation of an application on top of well-known and existing Web technologies, significantly reduces the learning curve for an application developer. We refer to this feature as **expressiveness**, which is an important design objective.

Finally, the system must follow an open approach to easy the development using, extending or defining standard Web languages, with a potential to be standardised, enabling interoperability with vertically-oriented closed platforms. Accordingly, an **openness** design objective becomes highly relevant. Clear and open specifications help third parties, application developers and broadcasters to create their adaptation logic or re-use existing generic adaptation rules on their multi-device media applications. Moreover, the research activities presented in this paper, as well as the other parts addressed by the MediaScape project, are being performed in communication and collaboration with different W3C Working Groups. The openness design objective contemplates to follow the recommendations of the on-going discussions and feed them with our outcomes to influence the future recommendations and standards.

The implications of the aforementioned objectives on the design decisions and the system architecture can be summarised as follows:

- **Autonomous behaviour**: The entire application will be delivered to each client device. At run-time each device will know how to behave accordingly to a shared context comprising all the devices used by a user simultaneously. A hierarchical component definition will enable each client device to have a general overview of the entire application in order to load specific resources as required. This strategy follows the lazy-load[12] approach, limiting resource usage on clients, servers and network.
- **Expressiveness**: The media-driven multi-device application will be defined on a common set of programming languages on top of standard Web technologies, and over a single developing environment without custom code on the server side to support adaptation.
- **Openness**: Open specifications over current standard Web technologies providing re-usability and development of new adaptation logics, enhancing those standard Web technologies in a way to likely be included in the roadmap of future standards.

As a conclusion, our solution requires a distributed service architecture enabling an autonomous behaviour. It will strongly rely on APIs and service interfaces, and will avoid dependencies from specific, proprietary implementations and/or centralised services, offering expressiveness and openness. The main objective is to open the market of media and service provisioning to all agents, allowing the coexistence of open-source and proprietary solutions through open specifications of module interfaces and data communication protocols.

[12] Lazy-loading is a design pattern to defer initialisation of an object until the point at which it is needed, contributing to the efficiency of the application.

Architectural design of the Web-based distributed adaptation solution

The strategy followed for the adaptation mechanism consists in introducing specific components acting as libraries on top of the Web Components standard prioritising the simplicity for application developers and broadcasters. They just have to include an application container (see *app-container* in **Fig. 6**) and add all the needed application related components inside. The following code guide describes how to create the application:

```
<html>
<link rel="import" href="components/app-container/app-container.html">
<body>
  <app-container>
    <component-1></component-1>
    …
    <component-N></component-N>
  </app-container>
</body>
</html>
```

Fig. 6 shows the adaptation stack diagram hosted by each client. There are some core blocks for adaptation purposes and others that provide base services. The last ones include a capabilities *introspector* to create self-context (*agent-context*), an *app-context* to provide shared state between all components within a session.

Going deeper in the distributed adaptation core blocks, there are three components. The *component-manager* is aware of the hierarchical definition of the application, taking charge of loading the necessary resources following the lazy-load approach. It orchestrates and manages the fired events between the different components. The *adaptation-engine* and the *UI-engine* are in charge of solving the two main adaptation challenges described in section 4: **multi-device adaptation** and **user interface adaptation** respectively. The nature of the devices regarding the **personal and shared device adaptation** challenge is also overcome by the *adaptation-engine* as a particular case of the multi-device adaptation. Finally, the **inter-components communication** is managed by the *component-manager*. It publishes the services provided by the components to the *app-context*, making other components able to invoke these services even if they execute it on another device.

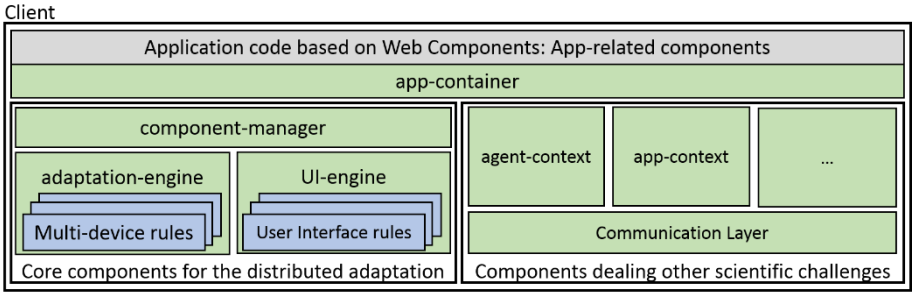

Fig. 6. Client-side block diagram of the Distributed Adaptation Architecture

The adaptation engines apply rules according to the context information and some properties defined in the application code. These properties are aggregated by the application developer as CSS style properties in the following way (both for a specific component and for the main application):

```
<template>
  <style>
    component-1 {
      adaptation-engine-property: value;
      UI-engine-property: value;
    }
  </style>
</template>
```

Fig. 7 presents the workflow of the two-level adaptation engines. The *adaptation-engine* will automatically apply the multi-device rules on each client device when there is an event on the multi-device dimension of the context. According to these rules, each client device will autonomously determine the set of components to show from the complete list included in the application.

The *UI-engine* will dynamically apply the UI rules to organise the layout for the list of components coming from the *adaptation-engine*. The *UI-engine* will add an abstraction level to organise the interface on top of existing responsive design mechanisms such as CSS and *media queries*. Therefore, the *UI-engine* will re-assess the rules for each incoming event on the multi-device dimension of the context, as well as for each fired browser resizing event on the client device, such as transitions from landscape to portrait.

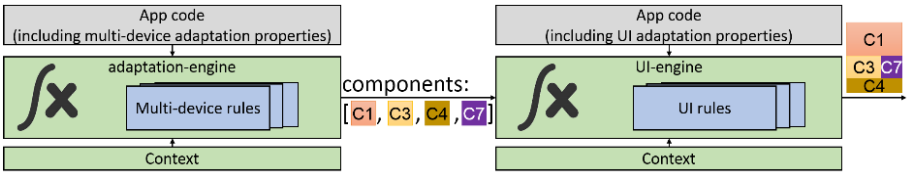

Fig. 7. The workflow and the decisions of the adaptation engines

The following subsections describe the most relevant core adaptation components: *adaptation-engine* and *UI-engine*.

Adaptation engine.

The *adaptation-engine* manages the multi-device behaviour of the application depending on the number of client-devices that a user is using simultaneously and the multi-device rules that the application developer has defined.

For common applications, developers can import existing generic rules on the *adaptation-engine*, and these rules will provide them the default properties that must be specified on the application. Anyway, if developers want to create custom properties, or specific algorithms, the distributed architecture is open to load new rules where custom tags come into the equation.

To overcome the **personal and shared device adaptation** challenge, the developer can define a property on each component establishing if that component is suitable for a shared device or not. This way, the *adaptation-engine* can take correct decision depending on the personal or shared nature from the device context.

More generally, and to overcome the **multi-device adaptation** challenge, the *adaptation-engine* will apply the rules on top of a shared context, where other components - not related directly with the adaptation (see **Fig. 6**) - publish the capabilities and features of each device. Therefore, the application developer can define specific mandatory capabilities for a component (e.g. a component requires geolocation information or broadcast capabilities to be available in a device) or suitable capabilities (like arranging one component in the biggest screen and highest resolution, and a second component in a touch screen to allow a better interaction).

As previously presented, these properties will be added as CSS style properties using an intuitive namespace that the *adaptation-engine* rule will use. Section 6 provides an implementation of a multi-device adaptation rule.

UI engine.

The *UI-engine* dynamically creates a responsive layout organising the output components provided by the *adaptation-engine*. The *UI-engine* applies rules according to the UI properties that the application developer defines for each component and for the application itself. The rules take into account parameters from the context information, such as the features of the device (screen size, capabilities, etc.) and decide, based on templates, the spatial scheme of the components. The layout is generated over CSS properties including *media queries* to integrate responsive design mechanisms.

The *UI-Engine* provides an abstraction layer, adding logic for the organisation of components, benefiting from the context information available. It facilitates the combinatory towards the expressiveness of the UI rules based on templates such as the examples presented in **Fig. 8**. When application developers are facing a single-device user interface, they define CSS templates to organise the items in the layout, usually creating a different template for each target device.

However, when application developers are dealing with multi-device applications, this approximation becomes unapproachable. For instance, for an application with 6 different items, within a multi-device scenario, a single device could show all the components, only 5 of them, 4 of them, etc. Furthermore, when for example 4 items are shown on a device, the combinatory of selecting 4 components from a total of 6 rises 15 different options (see equation 1 where n is the total number of items of the application and k is the number of items shown in the device, in the example $k=4$ and $n=6$). As a result, an application of 6 items has 64 different combinations to create a layout template for each target device [40].

$$f(n,k) = \frac{n \cdot (n-1) \cdot (n-2) \ldots (n-k+1)}{k \cdot (k-1) \ldots 2 \cdot 1} \quad (1)$$

The current implementation of *media queries* in Web Components allows use of *media queries* both for the main application and for each component. *Media-queries* fire events continuously on detected changes in the window size to re-assess visual properties. In our case, the *UI-engine* meets additional conditions since the *adaptation-engine* can decide to add or remove a component from that device (e.g. a new device is connected or disconnected in the session). Under these circumstances, the *UI-engine* re-organises the components in the layout, and each component applies the *media-queries* accordingly to the new assigned position and size.

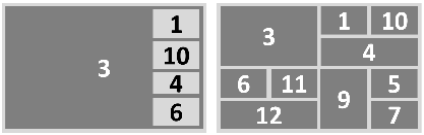

Fig. 8. Different organisations of a layout based on components. On the left a full screen component with picture-in-picture components on top of it is depicted. On the right, the figure presents a fixed layout creating a matrix of components based on their size and aspect ratio.

It is important to highlight that discussions have started in W3C to enable custom *media queries* in CSS based on scripts [41]. The *UI-engine* could take advantage of this feature when it becomes available, as well as fuel the discussion with specific requirements extracted from future steps in our research. Section 6 presents an implementation of a user interface adaptation rule.

6 Validation

This section presents the validation of the defined design objectives over a proof-of-concept implementation of the proposed Web-based distributed adaptation architecture. Our implementation develops all the client-side components described in **Fig. 6** using the Polymer library on top of the Web Components standard. This includes core components for the adaptation (*component-manager*, *adaptation-engine* and *UI-engine*), components dealing with other scientific challenges to provide information to the adaptation engines (*agent-context*, *app-context* and a communication layer to reach the available services) and the *app-container*.

In order to validate the proposed distributed architecture, two sample rules have been developed: one for multi-device adaptation and another one for user interface adaptation. The validation has been done based on the **autonomous behaviour**, **expressiveness** and **openness** design objectives.

Multi-device adaptation rule

We have implemented a multi-device adaptation rule for validation purposes based on the capabilities of each client device provided by the *agent-context* component (see **Fig. 6**). The capabilities are shared across all devices in the session by the *application-context*. However, both context modules are out of the scope of this paper.

The adaptation rule requires the application developer to specify the behaviour of the application using the following custom CSS properties for each component:

- ***adaptation-engine-needs***: a list of mandatory features in the client device for the component, such as the broadcast capability, geolocation information or a camera.
- ***adaptation-engine-best-fit***: a list of desirable features in the client device. For the implemented rule we use *big-screen* considering the resolution and the size of the display and *touchable* regarding the interaction capability of the device.
- ***adaptation-engine-behaviour***: it defines the replication behaviour for each component:
  - ***movable***: the application allows the user to interact with the component and transfer it from one device to another.
  - ***duplicable***: the component can be shown in more than one device at the same time. If it is not duplicable, the component will only be shown in one client.
  - ***required***: the component is indispensable at least in one device, no matter the multi-device context. If it is not required, the component will be shown whenever the required conditions can be satisfied through an established threshold.

The rule is applied autonomously for each client device on run-time. This means each client takes the decisions by itself considering shared context information coming from all clients connected to the session. **Fig. 9** shows the algorithm of the multi-device adaptation rule based on these properties. The movable property has been used to automatically overlay a button in those movable components enabling to move them to the other available devices.

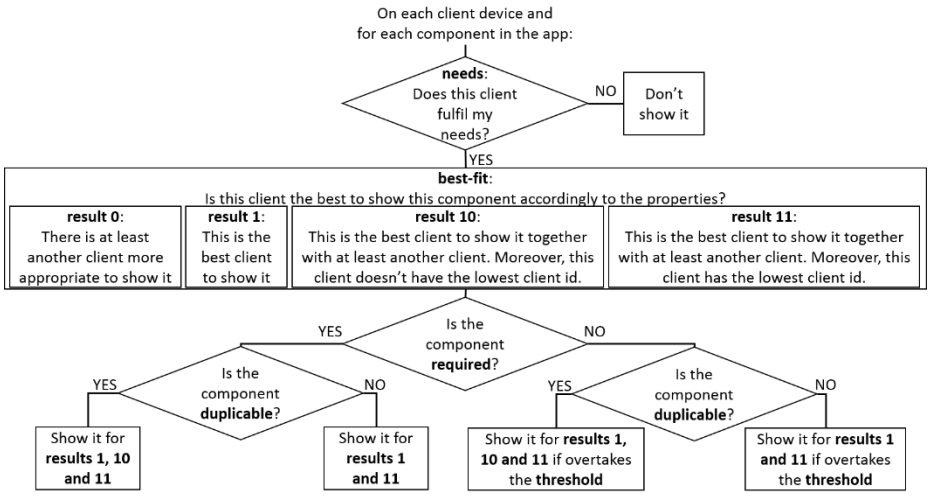

Fig. 9. Multi-device adaptation rule based on the multi-device context of the client device

User interface adaptation rule

We have also implemented a rule for the *UI-engine* for validation purposes. The proof-of-concept implementation includes the two templates depicted in **Fig. 8**: *picture-in-picture* and *fixed* templates. The rule requires the following UI properties for each component:

- ***UI-engine-priority***: the priority of the Web Component inside the application logic. The lower number, the highest priority.
- ***UI-engine-aspect-ratio***: the width and height coefficient that the Web Component needs to maintain its look. It is not mandatory to set an aspect ratio if it is not important for the component visualisation.
- ***UI-engine-size***: the size for a Web Component to show it properly. It is possible to define only the width or the height, setting a specific value in pixels, or defining a *maximum* size property instead a specific value.
- ***UI-engine-template***: the default template to be used. In this case *picture-in-picture* or *fixed*.

On top of these UI properties, the rule determines which template to be used for each client device. For our proof-of-concept implementation, we decided to use the *picture-in-picture* template when the *agent-context* discovers we are on a TV, PC or laptop. However, for smartphones and tablets the *fixed* template will be used. If the *agent-context* is not able to discover the type of device it will use the default template defined in the properties. Depending on this first step, the rule uses the other UI properties (priority, aspect ratio and size) to organise the components inside following the selected template.

For the *picture-in-picture* template, the highest priority component adopts the full screen position while the others are added in order or priority from top to down. We set a specific width and set the height accordingly, obeying the aspect ratio if it is defined or creating a square component.

In the case of the *fixed* template, the rule creates a matrix depending on the window size. Following the priority of the component, the first empty slot with enough space to show it will be assigned, according to the defined size. The engine iterates across the components with a priority-based order, and goes over the layout matrix from left to right and from top to down, filling empty slots with unplaced components. The components without a specific aspect ratio or fixed width and height provide more flexibility to create the layout.

Testing

The two aforementioned rules have been tested successfully in an application using 3 to 6 different Web Components, inspired by the Hybrid Live TV Programme scenario: three video components, the controller for the videos, a component to visualise metadata information and a map with the location of an object in the video.

Fig. 10. A setup of five different devices being used simultaneously during the performed tests

The tests have been accomplished using from one to five devices simultaneously. All devices were running the Google Chrome Web browser. **Fig. 10** shows one of the setups used during the experiments. The figure presents a TV showing the main video component with the front view of a train. The laptops on the left and right sides display other video components, with the left view and the rear-right view of the train, respectively. The tablet shows where the train is located in the map, while the smartphone provides two components: information about the next station (name and estimated arrival time), and the play/pause controls. All the content is synchronised, providing interactivity to the user, being able to pause and resume all views or jump to a specific location in the map influencing all the views. **Fig. 10** shows a screenshot of a video[13] that introduces more details about how the application adapts to the multi-device dimension when devices are connected and disconnected during the session.

The testing activities presented in this paper are exclusively focused on the adaptation domain. However, the proof-of-concept implementation integrates other modules in order to address other scientific challenges involved in multi-device application on a hybrid TV environment, such as discovery of the devices and their features in the session, association between devices, and synchronisation [15-17]. These modules, together with the adaptation activities, are being developed within the MediaScape project. The quantitative performance analysis of the system is outside of the scope of this paper, since the proof-of-concept implementation only seeks to validate that the multi-device adaptation design objectives are achieved, not the performance of all the integrated modules.

Regarding adaptation, the solution demonstrates automatic tracking of dynamic setups as clients come and go. The proof-of-concept implementation confirms the **autonomous behaviour** of each client device due to the distributed design of the adaptation architecture, since each device runs an instance of the same application. Consequently, the same adaptation rules are executed autonomously for each one of the clients when events are raised (e.g. a new device connected/disconnected on the session). It provides a consistent experience across multiple devices, behaving as expected, according to the properties defined by the application developer and the implemented rules.

[13] Video of one of the performed experiments http://vimeo.com/113514286

From a broadcaster or application developer perspective, the performed tests also conclude that in terms of code and application maintenance, the proposed approach provides simpler development through **expressiveness**. For the proof-of-concept implementation a single application was created. The application-related components have been developed following the Web Components specification. The adaptation JavaScript library and the aforementioned rules were also included, being re-usable from on application to another as they are not necessarily application specific. Our approach extends with CSS-style properties the necessary aspects to specify the multi-device adaptation behaviour to apply the rules on top of Web technologies. Alternatively, should our proposal not be used within the current ecosystem, we would have to pre-define how to separate the views of each one of the screens as part of the development phase. For example, by creating separate HbbTV and mobile applications, while using an event-driven server to manage the communication between applications [6-7]. This increases the specific code for the application, hinders reusability and complicates maintenance.

We also validated the **openness** of the approach by exploring further the interoperability with the Connected TV. The proof-of-concept implementation presented in the aforementioned video was running on top a Chrome Web browser in a Cozyswan Mk808b Android 4.2 TV Box. An HbbTV 1.5 device was also tested: Panasonic TX-42ASW654 LED TV. Although this device presents some difficulties to perform some of the HTML5 dependant functions, such as some parts of the CSS Grid Layout polyfill, a basic implementation does however run on this device. The implementation of the video components is responsible for creating the HbbTV video object when they are in an HbbTV device. The adaptation proposal presented in the paper decides where to show each component and how to arrange the layout on each device, but does not address the implementation of each component. Moreover, the multi-device adaptation rule also needs to be slightly modified when an HbbTV device is involved to avoid showing more than one video component in the HbbTV device, as usually only one single video object can be managed by HbbTV devices. The architectural design through independent core adaptation components (as managers and adaptation engines) and adaptation rules (both multi-device adaptation and user interface adaptation rules), grants the openness feature, simplifying the process of adding new rules, such as showing only one video component on an HbbTV application.

In order to evaluate the hybrid broadcast-Internet convergence of the solution during the experiments performed with the HbbTV device, the application was launched on a TV using the application signalling of the HbbTV specification. Other devices, such as smartphones and tablets, were associated afterwards following the experience of the Hybrid Live TV Programme scenario. To integrate the broadcasted video as one of the video components, we extended the multi-device adaptation rule to set broadcasting capabilities to a component in the *adaptation-engine-needs* property. Therefore, the broadcasted video will only be shown on a TV unless a streaming alternative is provided. The broadcast-Internet media synchronisation is out of the scope of this paper but addressed in [14] and [16].

As a summary, these experiments demonstrate our adaptation architecture proposal, giving a proof-of-concept implementation that fulfils all requirements, adapta-

tion challenges and design objectives. The viability of the proposed distributed adaptation architecture has been validated for the Hybrid Live TV Programme scenario. It presents open mechanisms to aggregate new rules with new multi-device adaptation algorithms to make decisions, run new user interface templates, and create custom logic to decide when to use the appropriate template on each device. All these features validate the feasibility of the scenarios presented in Section 4 through the proposed architecture.

7 Conclusions

This paper proposes a Web-based distributed adaptation architecture for multi-device applications driven by media content. This interoperable and standard approach allows broadcasters and media application developers to easily create this kind of applications by extending the Web Components standard. It also provides broadcast-Internet convergence through Web technologies, following the trend of the broadcast-related standards for the delivery of hybrid services, such as HbbTV. Moreover, the proposed approach improves the process that broadcasters and application developers currently need to implement, distribute and maintain over a set of complex technical solutions tailored to each specific target platform.

From the diverse scientific challenges involved in multi-device application creation, our architecture is focused on the adaptation domain. We lay the foundation of some principles of the hybrid TV ecosystem based on media consumption habits and market trends, and on the existing related work. On top of that, we identify the adaptation challenges extracted from the analysis of different scenarios and use cases.

In order to meet the adaptation challenges we propose a distributed adaptation architecture that provides a unified methodology and a common specification over Web Components, extending the current state of the art towards the multi-device dimension within the Hybrid TV environment. The presented architecture enables the development of multi-device applications offering users coherent experiences across multiple devices. The solution enables the following aspects:

- A distributed architecture that allows each client to make autonomous decisions according to a shared context and still provide a consistent view to the user through multiple displays simultaneously.
- A single development environment for broadcasters and application developers that facilitates the creation of multi-device applications without dealing directly with all the adaptation challenges and possible context situations, enabled by the expressiveness of the approach on top of Web technologies.
- The design of generic or custom adaptation rules to create multi-device responsive applications, boosting the re-usability and the development of this kind of applications, due to the openness of the solution built on top of standard technologies.

This paper presents a proof-of-concept implementation of the architecture, including some adaptation testing rules to validate the design objectives of the architecture and its modularity and flexibility. These experiments demonstrate distributed decision

making with autonomously executed adaptation engines, tracking dynamic setups automatically and performing real-time self-organising. The experiments also validate that the proposed scenarios can be achieved by creating adaptation rules on top of our architecture. The results reflect the openness of the proposed approach on top of Web Components and the expressiveness of the properties added for adaptation purposes built over standard technology.

In essence, we propose a distributed architecture for the creation of adaptive multi-device media applications using standard and interoperable Web technologies and extending those in a way to be standardised.

Acknowledgment

The MediaScape project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement 610404. The contribution from the UPV/EHU has been supported by the Basque Council under grant agreement IT395-10. The first author would like to thank Iñigo Tamayo, Angel Martin, Ana Dominguez and Igor G. Olaizola, researchers in Vicomtech-IK4, for supporting the architectural design and implementation presented in the paper.

8 References

1. Anthes, G. (2012). HTML5 leads a web revolution. Communications of the ACM, 55(7), 16-17.
2. Courtois, C., & D'heer, E. (2012, July). Second screen applications and tablet users: constellation, awareness, experience, and interest. In Proceedings of the 10th European conference on Interactive tv and video (pp. 153-156). ACM.
3. Claudy, L. (2012). The broadcast empire strikes back. Spectrum, IEEE, 49(12), 52-58.
4. Cesar, P., Bulterman, D. C., & Jansen, A. J. (2008). Usages of the secondary screen in an interactive television environment: Control, enrich, share, and transfer television content. In Changing television environments (pp. 168-177). Springer Berlin Heidelberg.
5. Millward Brown (March 2014). Marketing in a Multi-screen World. AdReaction 2014.
6. Zorrilla, M., Tamayo, I., Martin, A., & Olaizola, I. G. (2013, June). Cloud session maintenance to synchronise HbbTV applications and home network devices. In Broadband Multimedia Systems and Broadcasting (BMSB), 2013 IEEE International Symposium on (pp. 1-6). IEEE.
7. Ziegler, Christoph. "Second screen for HbbTV—Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios." Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on. IEEE, 2013.
8. Merkel, K. (2011, March). Hybrid broadcast broadband TV, the new way to a comprehensive TV experience. In Electronic Media Technology (CEMT), 2011 14th ITG Conference on (pp. 1-4). IEEE.
9. Weiser, M. (1991). The computer for the 21st century. Scientific american, 265(3), 94-104.
10. TVP-CPA: Cross-Platform Authentication (2014, September). Publication as an EBU Recommendation: EBU Tech 3366. https://tech.ebu.ch/docs/tech/tech3366.pdf

11. Zorrilla, M., Martin, A., Tamayo, I., O'Halpin, S., & Hazael-Massieux, D. (2014, June). Reaching devices around an HbbTV television. In Broadband Multimedia Systems and Broadcasting (BMSB), 2014 IEEE International Symposium on (pp. 1-7). IEEE.
12. Desruelle, H., Lyle, J., Isenberg, S., & Gielen, F. (2012, September). On the challenges of building a web-based ubiquitous application platform. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing (pp. 733-736). ACM.
13. Desruelle, H., Isenberg, S., Botsikas, A., Vergori, P., & Gielen, F. (2014, August). Accessible user interface support for multi-device ubiquitous applications: architectural modifiability considerations. Universal Access in the Information Society, 1-15.
14. Brandenburg, R., & Veenhuizen, A. (2013). Immersive second-screen experiences using hybrid media synchronization. In Media Synchronization Workshop 2013 (MediaSync).
15. W3C Community Group (2015). Multi-Device Timing Community Group. https://www.w3.org/community/webtiming/
16. W3C Editor's Draft (2015, March). Timing Object. http://webtiming.github.io/timingobject/#broadcasting---time-consistent-live-web-productions
17. Arntzen, Ingar M., Njål T. Borch, and Christopher P. Needham. "The media state vector: a unifying concept for multi-device media navigation." Proceedings of the 5th Workshop on Mobile Video. ACM, 2013.
18. MediaScape EU project report (2014, April). Usage Scenarios and Requirements. Deliverable D2.1. http://mediascapeproject.eu/files/D2.1.pdf
19. Nielsen report (2014, Q2). Shifts in viewing: The cross platform report. http://www.nielsen.com/us/en/insights/reports/2014/shifts-in-viewing-the-cross-platform-report-q2-2014.html
20. Nielsen report (2014, Q3). State of the Media: Audio today. http://www.nielsen.com/us/en/insights/reports/2014/state-of-the-media-audio-today-q3-2014.html
21. Strategy Analytics (2014, Q1). 2013 Smart TV Shipments Grew 55 Percent. http://www.strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=5472
22. An Ericsson Consumer Insight Summary Report (2013, August). TV and Media. Identifying the needs of tomorrow's video consumer. Ericsson ConsumerLab. http://www.ericsson.com/res/docs/2013/consumerlab/tv-and-media-consumerlab2013.pdf
23. Gartner report (2014, Q1). Forecast: PCs, Ultramobiles, and Mobile Phones, Worldwide, 2010-2017. http://www.gartner.com/newsroom/id/2645115
24. NAPTE survey (2014, Q1). https://www.natpe.com/press/release/130
25. Nielsen survey (2014, Q1). Sports Fans Amplify The Action Across Screens. http://www.nielsensocial.com/sports-fans-amplify-the-action-across-screens/
26. Nielsen survey (2014, Q2). Who's tweeting about tv? http://www.nielsensocial.com/whos-tweeting-about-tv/
27. Soares, L. F. G., Costa, R. M., Moreno, M. F., & Moreno, M. F. (2009, October). Multiple exhibition devices in DTV systems. In Proceedings of the 17th ACM international conference on Multimedia (pp. 281-290). ACM.
28. Soares, L. F. G., Moreno, M. F., & De Salles Soares Neto, C. (2010). Ginga-NCL: declarative middleware for multimedia IPTV services. Communications Magazine, IEEE, 48(6), 74-81.
29. W3C World-Wide Web Consortium. 2008. Synchronized Multimedia Integration Language – SMIL 3.0 Specification, W3C Recommendation. http://www.w3.org/TR/2008/REC-SMIL3-20081201/

30. Pereira, F. C. (2002). The MPEG-4 book. Prentice Hall Professional. ISBN: 0-13-061621-4.
31. Manjunath, B. S., Salembier, P., & Sikora, T. (Eds.). (2002). Introduction to MPEG-7: multimedia content description interface (Vol. 1). John Wiley & Sons.
32. W3C Editor's draft (2014, November). 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
33. W3C Recommendation (2012, June). Media Queries. http://www.w3.org/TR/css3-mediaqueries/
34. W3C Recommendation (2010, December). Mobile Web Application Best Practices. http://www.w3.org/TR/mwabp/
35. W3C Working Draft (2014, September). CSS Flexible Box Layout Module Level 1. http://www.w3.org/TR/css3-flexbox/
36. W3C Working Draft (2014, October). CSS Regions Module Level 1. http://www.w3.org/TR/css3-regions/
37. W3C Working Draft (2014, May). CSS Grid Layout Module Level 1. http://www.w3.org/TR/css3-grid-layout/
38. W3C Working Group Note (2014, July). Introduction to Web Components. http://w3c.github.io/webcomponents/explainer/
39. Techrunch (2013, May). Google Believes Web Components Are The Future Of Web Development. http://techcrunch.com/2013/05/19/google-believes-web-components-are-the-future-of-web-development/
40. Zorrilla, M., Martin, A., & Tamayo, I. (2015, June). User Interface adaptation for multi-device Web-based media applications. In Broadband Multimedia Systems and Broadcasting (BMSB), 2015 IEEE International Symposium. Pending to be published. IEEE.
41. W3C Editor's Draft (2014, October). Media Queries Level 4. http://dev.w3.org/csswg/mediaqueries4/#script-custom-mq