Journal of Virtual Reality and Broadcasting, Volume 4(2007), no. 17
JVRB EuroITV 2006 Special, no. 7

Semi-Automated Creation of Converged iTV Services: From Macromedia Director Simulations to Services Ready for Broadcast

Emmanuel Tsekleves and John Cosmas

Brunel University
School of Engineering & Design
Uxbridge UB8 3PH, UK
phone, email: Emmanuel.Tsekleves@brunel.ac.uk,
John.Cosmas@brunel.ac.uk

Abstract

While sound and video may capture viewers' attention, interaction can captivate them. This has not been available prior to the advent of Digital Television. In fact, what lies at the heart of the Digital Television revolution is this new type of interactive content, offered in the form of interactive Television (iTV) services.

On top of that, the new world of converged networks has created a demand for a new type of converged services on a range of mobile terminals (Tablet PCs, PDAs and mobile phones). This paper aims at presenting a new approach to service creation that allows for the semi-automatic translation of simulations and rapid prototypes created in the accessible desktop multimedia authoring package Macromedia Director into services ready for broadcast. This is achieved by a series of tools that de-skill and speed-up the process of creating digital TV user interfaces (UI) and applications for mobile terminals.

First presented at the 4th European Interactive TV Conference EuroITV 2006, extended and revised for JVRB

The benefits of rapid prototyping are essential for the production of these new types of services, and are therefore discussed in the first section of this paper. In the following sections, an overview of the operation of content, service, creation and management sub-systems is presented, which illustrates why these tools compose an important and integral part of a system responsible of creating, delivering and managing converged broadcast and telecommunications services. The next section examines a number of metadata languages candidates for describing the iTV services user interface and the schema language adopted in this project. A detailed description of the operation of the two tools is provided to offer an insight of how they can be used to de-skill and speed-up the process of creating digital TV user interfaces and applications for mobile terminals. Finally, representative broadcast oriented and telecommunication oriented converged service components are also introduced, demonstrating how these tools have been used to generate different types of services.

Keywords: Interactive Television, Mobile Terminals, User Interfaces, iTV Services, Macromedia Director

1 Motivation

The next generation multimedia mobile services will provide a variety of multimedia content such as moving pictures, movies, images and Internet pages delivered via converged broadcast (DVB-T, DVB-H, DAB), cellular (UMTS, GPRS), and wireless (WLAN,

urn:nbn:de:0009-6-10862, ISSN 1860-2037

BRAN etc.) networks targeted to a range of different end-user terminal hosting operating systems, Application Program Interfaces (APIs) (e.g. Java MHP, Java MIDP etc.) and player resources (MPEG-2, MPEG-4, H.263, H.264, HTML browser etc.). These services are wanted by consumers to access content in mobile environments and needed by network providers to generate new sources of revenues to support investments in their new networks. It is therefore necessary that "crisp and clear" user requirements be drawn up for the development team in order for them to create the services. Currently this service creation process utilises an iterative software development cycle where once requirements are captured a software based prototype is iteratively developed and validated. This process is expensive because it requires a full software prototype to be developed at the end of each iterative step until a final solution is created. To reduce this cost a new service development methodology has been developed that allows the efficient creation of services through the application of rapid prototyping, simulations and semi-automatic fast application development [SE03, Kra96]. This service development methodology is more efficient because it removes the need for a software-based prototype to be developed in the traditional approach. Alternatively the designer creates and simulates "rapid" prototypes using graphical design tools. These rapid prototypes are then used as a template for attaching pre-implemented functions in a semi-automatic manner thus creating the final service application.

The purpose of service scenarios is to derive "crisp and clear" user requirements by describing in detail and storyboarding the service components, their technical specifications, the situation where they will be consumed and the target audience for each of our distinct service operators-customers. Once the service scenario has been defined, the rapid prototypes can be developed.

The purpose of rapid prototyping of multimedia services is to demonstrate the viability and validity of service proposals without going through the huge effort of building delivery systems and end-user terminals and deploying services. This is a very important step towards service development because, first, it allows service designers to visualise how their proposals would look like on an end-user terminal to obtain a common understanding of the service concept and, second, it enables service designers to test the usefulness of their ideas on relevant user groups well before the completion of any delivery system and end-user terminal technology. A rapid prototype of a scenario can also serve as a testing medium, by involving the user in the earlier stages of the design cycle to establish user acceptance and therefore be modified accordingly. This methodological approach is known as scenario-based requirements gathering. Scenario-based requirements gathering techniques have been used in Human Computer Interaction research as an effective way to capture, analyse and communicate user's needs for technology [CR92, Car00]. Three scenarios have been tested with real users in order to ensure that the selected services really correspond with user needs [PRS02, SDP$^+$04].

The purpose of semi-automatically generating applications is to reduce development times for creating the first and updated versions of the applications for a range of different end-user terminals hosting different operating systems, APIs, network interfaces and player resources. This is very important because the introduction of new services is increasingly becoming a very complex logistical exercise due to the proliferation of different mobile terminal types being introduced into the market with a new model being introduced almost every week from a single terminal manufacturer [Ind06] and due to the proliferation of different service providers, including broadcasters, telecommunication operators and independent providers. Even though the motivations and type of services of these providers are quite different, the tools used to develop the services should be the same.

This paper describes a Rapid Prototyping and Semi-automated Application Generation tools set, which can produce applications very fast from a rapid prototype simulation for all classes of service providers.

2 Service Provision Architecture Overview

This section describes the overview of a service provisioning architecture, which is shown in Figure reffigure01 at sub-system and component level. It consists of the Content Creation, Service Creation and Service Management sub-systems. This overview provides an insight on how the Rapid Prototyping and Semi-automated Generation tools can be incorporated within a typical provisioning architecture.

More precisely, this system architecture has been designed and developed to provide content for services

urn:nbn:de:0009-6-10862, ISSN 1860-2037

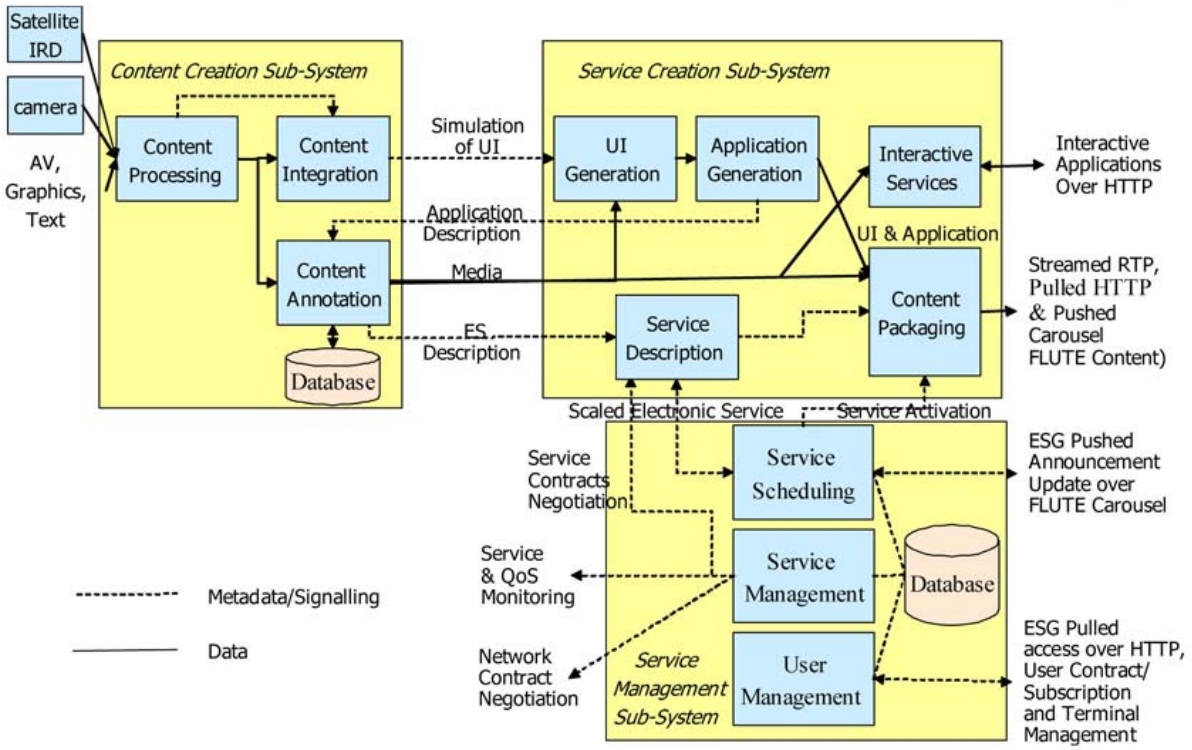

Figure 1: Overall System Architecture – sub-system and component level

in a single base format and generating the graphical components of the end-user terminal's Graphical User Interface (GUI) for delivery into a number of converged (DVB-H/T, UMTS, GPRS) end-user terminals (Tablet PC, PDA, Mobile Phone). The audio visual base format employed in the INSTINCT project was MPEG-2 [ISO95] and the GUI base format was XML with JPEG/PNG. The process of integrating content was achieved using a simulation tool that allows a fast prototype of the service to be generated for testing with the organization commissioning the service and with potential consumers.

More precisely, the Content Creation sub-system consists of the Content Processing, Content Annotation and Content Integration components. The Content Processing component tool provides access to and processing of MPEG encoders, transraters and transcoders (MPEG-2 to MPEG-4). This transrating is being made for the delivery of audio visual content to low-powered terminals such as mobile phones. The Content Integration component aggregates all the User Interface (UI) elements (graphics and buttons) and content elements (video/audio clips, Internet links) that comprise the service's content and user interface look-and-feel. A desktop publishing tool (Macromedia Director) was used to prototype services quickly and iteratively construct and test them on organizations and their customers who were commissioning the service.

The Content Annotation component consists of a Service description tool that generates imports and maintains content descriptions and basic service descriptions. It imports an XML file generated by the Application Generation tool that describes the basic services, contents and associate content and enables the addition of more content service descriptions which are stored into the common database that is shared with Service Description tool.

The Service Creation sub-system consists of the User Interface Generation, Application Generation, Service Description, Content Packaging and Interactive Services components. Once the user interface design process is completed then the UI Generation software component is used to parse the desktop publishing tool file and generate a XML description of the user interface with its graphical components. The XML description could be further translated to HTML, WML and several other formats, depending on the navigation engine resident on the targeted end-user terminals. In order to add functionality to the

user interface graphical widgets, the semi-automatic Application Generation software component, which is shown in Fig. 4, is used to map terminal functions, depicted on the right hand side of the tool, onto UI graphical widgets, depicted on the left hand side of the tool as either a DOM tree or on the centre of a tool as a graphical widget. The different terminal profiles take into account aspects such as operating system (Linux, Windows Mobile), middleware API (MHP, MIDP, JSR272 [RW07]) and device (Tablet PC, PDA, Mobile Phone). The Service Description software component generates scalable service descriptions, which include the session, the schedule, the coding format attributes, and the network area descriptions and delivers these descriptions as a package to the Push Broadcast Portal in the Service Management subsystem [ACC+07].

One or more Service Creation sub-systems deliver service descriptions to a single Service Management sub-system which collates all of them and delivers them as an Electronic Service Guide (ESG) to end-user terminals activating the Content Packaging servers at the appropriate time. The Service Management component is used for setting up contracts between the business entities and then managing them by monitoring the Quality of Service (QoS) for the delivered services. The User Management component is used to establish subscription and on-demand services with end-users and to provide the end-user with a pulled version of the ESG.

3 Metadata Language Comparison

Today it is possible to access information via a wide range of mobile computing devices. Despite the importance these communication devices have gained, too much time is spent in designing UI's for each new device, since there is not a unified standard language. Instead the design of the UI is restricted to specific programming languages, varying from one device to the other (e.g. J2ME for PDA's, Wireless Markup Language for mobile phones, etc). Since our research aims at encompassing several portable devices, from set top boxes to PDAs, tablet PCs and smart phones, a generic language for creating device-independent user interface must be sought.

In addition, a significant aim for a broadcasting related metadata language is the use of a language, which can be economically broadcasted along with the material and media (UI description and graphics) it describes. More precisely, the user interface description language should qualify all of the following criteria:

- *Consistency:* should be consistent amongst different systems.
- *Platform Independency:* the user interface should be designed in a generic way without worrying about the system on which the UI will be employed.
- *Extensibility:* when extra functionality needs to be incorporated the author should be allowed to apply modifications without starting from scratch.
- *Prototyping:* designers should be able to involve the user in the initial design stages and conduct early user-trials.
- *Usability:* the language should be fairly easy to use by non-specialists, minimizing the learning curve.
- *Open Standard:* it should be a non-proprietary language recognised and supported by an international body or consortium.

The major user interface description language candidates for the user interface generation tool are: User Interface Mark-up Language UIML, Scalable Vector Graphics (SVG), Portable Content Format (PCF) and Extensible Markup Language (XML). Let us take and examine each one of them separately.

The UIML is an XML language that allows designers to describe the UI in generic terms and then use a style description to map the UI to various operating systems. In particular, the UIML describes the UI with five different sections: description, structure, data, style and events [APB+99]. Despite the fact that it allows a fairly device independent description of a UI, its main shortcoming is that it must provide a target specific renderer that will convert the UIML metadata generating the user interface [An02]. The fact that the renderers support Java, HTML, WML and VoiceXML, means that the user interface will work only for platform-specific devices. It should be also noted that the company has developed a UIML tool (LiquidUI) in which the Java renderer is currently only compatible with JDK 1.3 [Har04], not taking into account the effort one should put in writing these renderers for every device. In addition it takes quite some

time to learn it, becoming even more complicated by the usage of the different sections (description, data, events, etc.).

Scalable Vector Graphics (SVG) composes another candidate technology for using in our tool. More precisely SVG is an XML-based language for describing two-dimensional graphics and graphical applications in XML. It is a W3C standard [W3C03] and composes a platform and device independent solution to graphic design and display. Being an XML language it integrates with other W3C standards as the Document Object Model (DOM) and XSL. Although SVG seems as a good candidate for the metadata language since it is extensible, consistent, platform independent and an open standard, it was unfortunately not supported by the DVB-MHP when this research project started its work.

The Portable Content Format (PCF) is a standard specified by the DVB project to describe interactive digital television services. PCF is intended to provide the industry with a platform-independent format for the transfer of interactive content between different open and proprietary middleware platforms [BCH+06]. A description captured using the PCF, however is not intended for actual transmission in a digital television network. Rather, it is an intermediate form that will need to be converted to some platform specific representation prior to transmission. There are two main modes of implementing PCF on the terminal side:

a with delivery of the output either directly to the target platforms engine (execution or presentation),

b or possibly via client-server model using a small "micro" browser application sitting on top of an execution engine.

PCF still requires that the party (or parties) handling the conversion will have to implement appropriate conversion infrastructure and software [Pro05]. Furthermore PCF is mainly focused on data exchange between set top box middleware solutions and not middleware platforms for smaller low-power processing terminals such as PDAs and mobile phones. Both SVG and PCF form good solutions that provide added functionality and benefits. They both complement the functionality of our existing tool and is planned to be integrated in a following phase.

Last but not least, XML constitutes a platform independent language created by the W3C Consortium (has also developed HTML), with its main strength the separation of content and presentation. As far as consistency, this can be achieved through the usage of Document Type Definition (DTD). Furthermore through the usage of eXtensible Stylesheet Language (XSL and XSLT) [W3C04] it can be published in different formats. Particularly, in terms of user interface design of iTV services, it can assist in the achievement of a generic UI transformed into the different formats that each device and platform requires quickly, allowing rapid prototyping of the UI design though the creation of a new customized schema.

4 User Interface XML Language Schema

The User Interface XML Schema language was developed for the sole purpose of providing XML descriptions of the User Interface produced by the User Interface Generation Tool.

In general the following functionality is provided:

- User Interface - The root of the schema is comprised by the $< UI >$ element, which has one attribute "Device" (see Figure 2). The "Device" attribute holds information about the screen resolution of the terminal where the UI is presented. (i.e. 1024x768 for a Tablet PC, 260x320 for a mobile phone, etc)

- The $< Node >$ element represents the different nodes/pages of the User Interface. For example a User Interface with 12 pages has 12 $< Node >$ elements each one of them distinguished by a unique "ID" attribute. The $< Node >$ element, in turn, can contain a number of combinations of the following child elements: Button, Graphics, TextArea, TextField, Video and Audio.

- Button - The $< Button >$ element is divided into four categories according to the button type used: Graphic Button, Radio Button, Check Button and Push Button. This element holds information about the button component's screen coordinates (xCord, yCord), image size (Width, Height) and many other parameters. The most important of those are the "ActionID", which is empty but this is responsible for capturing the information that will be filled by the Application Generation Tool that will hold information regarding the functionality/action that takes place

in the back-end when the user clicks on the button. For instance a button component mapped with a link to the video decoder and player APIs to decode and play a video clip, e.g. $< Action >$ VideoPlayer, Play, "Filename" $< /Action >$. The $< Functionality >$ element holds the number of the Node (the next page) the user navigates to.

- Graphics - The $< Graphics >$ element lists the images that compose the User Interface layout and any pictures that are part of the content.
- TextArea - the $< TextArea >$ element holds pure textual information that can be displayed anywhere on the UI. Furthermore like the $< Button >$ element the $< TextArea >$ element can hold functionality information, which enables the user, when the $< TextArea >$ component is clicked, to navigate to a different page. (see figure 2)
- TextField - the $< TextField >$ holds textual information that are displayed only in a Field/Text Field component.
- Video - The $< Video >$ element holds video clip related information, such as duration, screen coordinates of the video clip.
- Audio - The $< Audio >$ element holds audio clip related information such as sample rate (e.g. 22.050 kHz), bit depth (e.g. 16 bits) and duration

5 User Interface Generation Tool

So far the task of developing applications and user interfaces was part of software engineer's job description, who was given the arduous task of hand-writing all the code for the service applications on top of their underlying user interfaces in the Multimedia Home Platform (MHP) [Pro03] for more powerful terminals (Tablet PC's, PDA's, Set -top boxes) and Mobile Information Device Profile (MIDP) [Mic06] for mobile phones.

Previous approaches have been to create digital TV applications and their UI's using the Multimedia and Hypermedia Experts Group (MHEG) or MHP standard. In case of the former, it is part of the ISO/IEC 13522-5 Multimedia and Hypermedia Experts Group (MHEG) [Gro03] international standard used for "Digital Teletext" in UK digital terrestrial television. It is a language for describing multimedia presentations as a collection of objects. More precisely, UIs in the MHEG have to be described as Multimedia presentations using a non XML-based predefined scripting language, thus forcing programmers to deal with the UI as well the coding aspects of the digital TV applications as well. Additionally since it was mainly developed for designing "Digital Teletext" it is only able to produce UIs that are very limited from a design of interactive TV services point of view. Furthermore, employing a non xml-based solution means that the UI cannot be transformed to other formats and be ported over different type of terminals/platforms, other than Set-top boxes. This is further restricted by the user navigation methods defined in the MHEG, which account only for a remote control based mode of user input and interaction [Gro03].

Compared to existing solutions, such as the MHEG, MHP is a second generation system with a significant step forward in functionality, enabling a much greater variety of interactive applications, services and thus interactivity.

Previous approaches to create UIs using MHP have used its Java Abstract Window Toolkit (AWT) or Home Audio Video interoperability (HAVi) components [HAV04]. This approach however is very effort and time demanding, since Java components such as these must be to hand-coded in Java by the programmer, which is not an easy task, especially when you attempt to reproduce attractive and complex user interfaces that require complicated layouts and components. Furthermore, the produced UIs look and feel offers an inflexible, uncreative and unexciting user experience since AWT and HAVi are constrained in creating anything other than simple and fixed shaped UI components. On top of that, building a UI is a highly iterative process, meaning that the code for the UI has to be rewritten many times. The shortcomings of UIs produced in this manner may be less obvious for services delivered to big TV sets via set-top boxes but are not in any way suitable for smaller screens where the "lean forward versus lean backwards" mode of interaction and viewing applies.

Moreover and following the previous discussion it is important to create tools to enable graphics designers, interaction designers and software engineers to work within their own area of expertise. However it is extremely difficult and rare to find a good designer who is also an experienced programmer and vice versa. This is particularly the case for Digital TV that re-

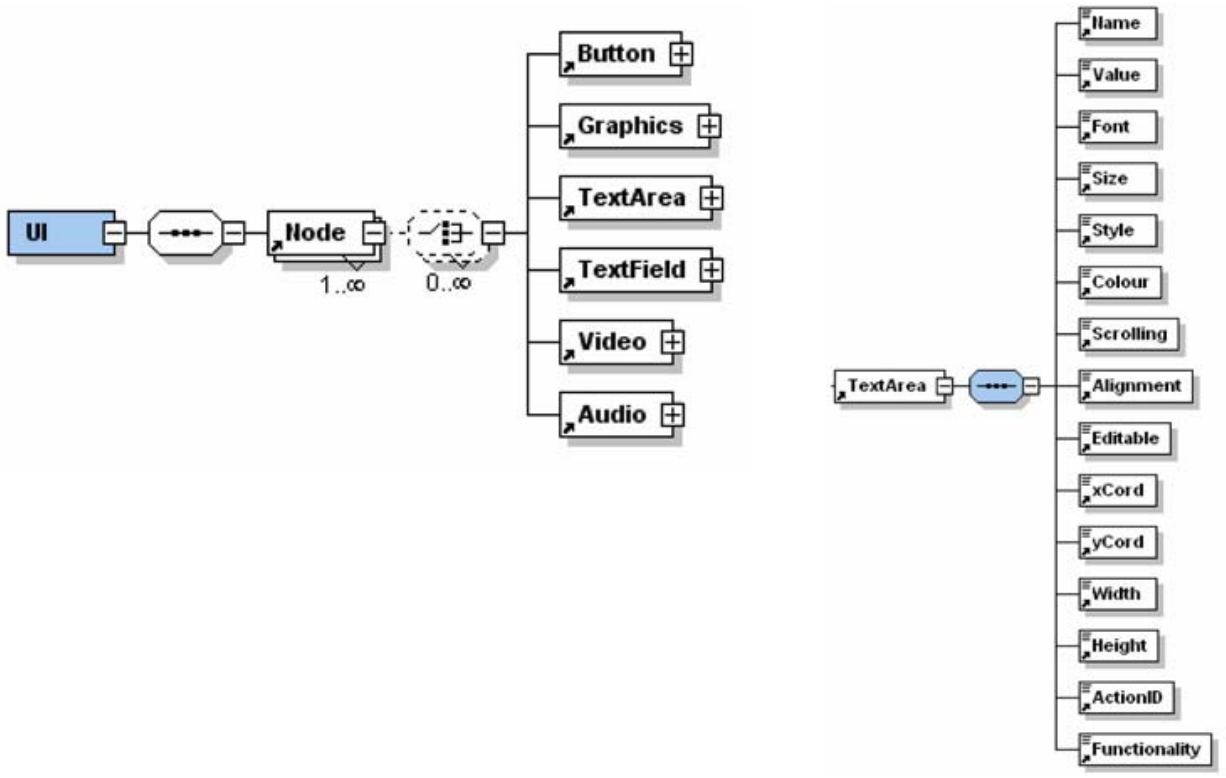

Figure 2: The User Interface XML Schema Language

quires people to work with the Java languages of the MHP and MIDP respectively.

Tools such as Director are very useful and user-friendly since they use graphical means to express graphical concepts (drag-and-drop user interface). Also the addition of an event-based scripting language (Lingo) and custom pre-programmed modules makes Director an ideal tool to employ for the design of fast iTV prototypes, since it maps well to the direct-manipulation graphical user interface style. Therefore by moving some aspects of the user interface implementation from conventional code into an easy-to-use multimedia graphic authoring package, these aspects of user interface design implementation are made available to those who are not conventional programmers.

Despite its ease-of-use and usefulness in providing fast prototypes and simulations of iTV services, Director forms a proprietary tool that does not interface with the open standard solutions of MHP and MIDP employed today in displaying GUIs in Tablet PCs, set-top boxes and mobile devices respectively. Consequently there is a demand for the development of a tool that will create that interface by employing an open-source standard such as XML. This tool should be responsible for converting and outputting the user interface created in Director into a format that is compliant with the MHP specification and can be used in conjunction with MHP Xlets as well as other middleware solutions such as MIDP.

The proposed user interface Generation Tool is part of a three-stage UI production process, where the user interface is divided into a two-tier solution, in which the graphical part of the UI (front-end) is produced independently from the actual applications that lie behind the UI (back-end).

In particular, in the first stage the designer creates the graphical elements (components) of the UI using any commercial graphical design tool. These graphical components are then exported as bitmap graphics in one of the graphic formats currently supported by the MHP (GIF, JPG and PNG). In the next stage these graphics are imported into Macromedia's Director [Sys06] authoring package to create and simulate the user interface layout, content (text, internet pages and snippets of audio and video). These two stages are entirely concerned with the graphical components of the user interface and therefore are responsible for the

front-end.

More precisely, the new tool composes a fully automated package for parsing the different nodes-pages of the UI created in Director that automatically exports the full XML description of the user interface. The tool works by scanning Director's score, a timeline where the author places the graphical components, both horizontically (in terms of frames) and vertically (in terms of channels), so that all the graphics of every UI node/page are identified and automatically output to the target XML file. It is also records the functions used in Director's scripting language (Lingo) to interlink the different pages-nodes.

All this information is exported in a single XML file using a descriptive schema (see Figure 2), categorised and built in such a way that can be then transformed into different formats, such as HTML.

The tool is also responsible for automatically copying all the graphic files and content used in the UI into a folder, thus speeding-up the production and distribution process.

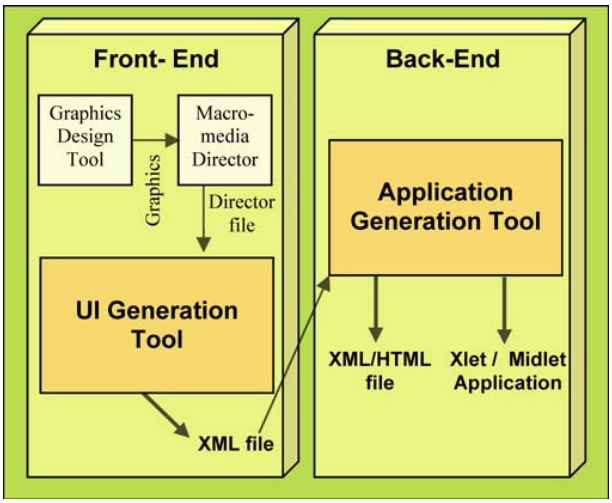

Figure 3: The User Interface XML Schema Language

Therefore, instead of creating a new tool from scratch to perform this task and invest time in training designers to learn how to use it, an already existing successful commercial multimedia authoring package is employed and enhanced.

Last but not least, without having to write a single line of Java code or XML metadata, designers can easily, effectively and quickly produce a graphical user interface (front-end) in a standard format (XML) that can be then manipulated by MHP and MIDP to render the UI on the end-user terminal. Therefore, this approach serves as a User Interface simulator where "what you see is what you get". This approach can be used to rapid prototype and simulate end-user terminal user interfaces and services where human factors specialists and interaction experts can conduct user trials with end-users, utilising the trials' feedback to iteratively create an aesthetically pleasant and user-friendly design.

In the last and more important step, the User Interface Generator Tool semi-automatically performs the most complex task (accounts for the back-end), which is discussed in the next section.

The benefits gained by using this service development methodology increases with relation to the complexity of the service application.

5.1 Limitations

Director is a very versatile package with numerous uses not limited to just building user interfaces. The user interface generation tool accounts for only a small subset of the potential applications of Director, specific to developing user interfaces. In addition, the testing performed on the user interface generation has revealed that parsing of the Director file will fail unless specific set of design and authoring rules are followed prior to the import of the file in the user interface generation tool. Therefore the designer/author of the Director file must ensure his design follows some requirements and guidelines.

More precisely, Director does not distinguish which components are simple graphics and which function as buttons. Thus this needs to be manually specified by the author. This can be done either when the graphical component is being designed in a graphic design package (e.g. Photoshop) by appending the word "Button" to the filename of the component or inside Director by appending the word "Button" to the name of the cast member e.g. the name of the graphic cast member "ClickOk" will become "ClickOkButton".

Furthermore, MHP supports only three graphic types GIF, JPG and PNG. So the UI graphical components should be exported in one of the above formats. PNG 24-bit is recommended since it supports transparency and it is good at retaining vector-based graphics.

Lastly, the tool has been programmed to record some functionality. Namely it records the change of a leaf node (e.g. jumping from one page to another). Supported functions are as follows:

go to frame 45 = goes to the leaf node specified by the frame no

go to "startMarker" = goes to the leaf node specified by the marker name

go "startMarker" = goes to the leaf node specified by the marker name

goToNetPage "www.instinct.org" = opens the web page in a browser

6 Application Generation Tool

After the UI is generated (front-end), it is necessary to code the logic of the application itself (back-end) and then pack the application to be broadcasted and delivered to the terminal.

The Application Generation tool comprises a head-end solution that is responsible for semi-automatically developing MHP (for Laptop PCs, Tablet PCs, PDAs) and MIDP (for mobile phones) applications for Digital TV. Since MHP and MIDP are different standards with different sets of Java APIs, different sets of applications have to be created.

On the terminal side lies a rendering subsystem that displays the front-end and permits Xlet navigation via http links.

The Application Generation tool functions by importing the XML Description of the generated UI (see Figure 4) and presenting it in a hierarchical node by node (page by page) manner both as a XML DOM tree and as a graphical layout (left and centre panel of the screenshot window of Figure 4). The user is also able to further manipulate the Java DOM tree by removing, modifying and adding new components onto the UI without having to revert to the Director simulation and the User Interface Generation tool.

The tool incorporates a number of device profiles (e.g. Tablet PC, PDA) with relevant functional modules, (MHP and MIDP in this case) for different platforms (e.g. Linux, Windows) that the user can import as plug-ins (right panel of Figure 4), This includes playing media files, storing and retrieving media content to storage devices, loading and executing applications and several others. Although only MHP and MIDP functional modules have been written so far, additional middleware solutions based on other Java and C++ APIs can be added very easily, increasing the spectrum of platforms and terminals on which the applications can be consumed.

These functional modules (commands) are classes and methods that are written by the software engineers and imported on to the tool. From this list the designer is able to select the profile for the specified type of device and platform and then, in a drag and drop manner, map the relevant functionality to the particular leaf node of the UI. For instance, a button component can be mapped with a link to call the player API's and play a particular video clip. For this reason a protocol (a specially defined String) has been defined linking the front-end (UI components) to a specific functionality (back-end). This protocol is created when a functional module is drag-and-dropped onto a user interface component and is stored on the XML file that describes the UI under the $< ActionID >$ element. This is then parsed by the resident application on the terminal that performs a search to see if there is a method that corresponds to the contents of this protocol.

For instance if the functional module (command) that has been assigned to a button is responsible for playing an MPEG-4 video clip named "BBC Newsnight" on a MHP enabled terminal such as a set-top box or Tablet PC in a 640x480px resolution then the protocol will be formed as:

```
mhp||playVideoClip|BBC Newsnight.mp4|640x480|
   ↑          ↑              ↑              ↑
API name  method/class   parameter1     parameter2
```

Based on this protocol one can device and write a number of other related functional modules. The following list provides some of these that have been implemented and tested throughout the INSTINCT project:

- Tune Channel: tunes to a TV channel. One parameter that specifies whether to tune to next (+1) or previous (-1) channel or specific channel if number is known (e.g. 6) i.e. mhp||TuneChannel|+1|
- Launch Xlet: launches a specific Xlet application. Parameter: name of Xlet i.e. mhp||launchApp|myXlet|
- Play video or audio clip: plays a specified video clip. Parameters: audio/video clip filename, video clip's resolution. i.e. mhp||playAudioClip|radioOne.mp3|
- Display HTML: launches an Internet browser that displays the specified URL of an HTML page i.e. mhp||launchHTML|http://news.bbc.co.uk/|

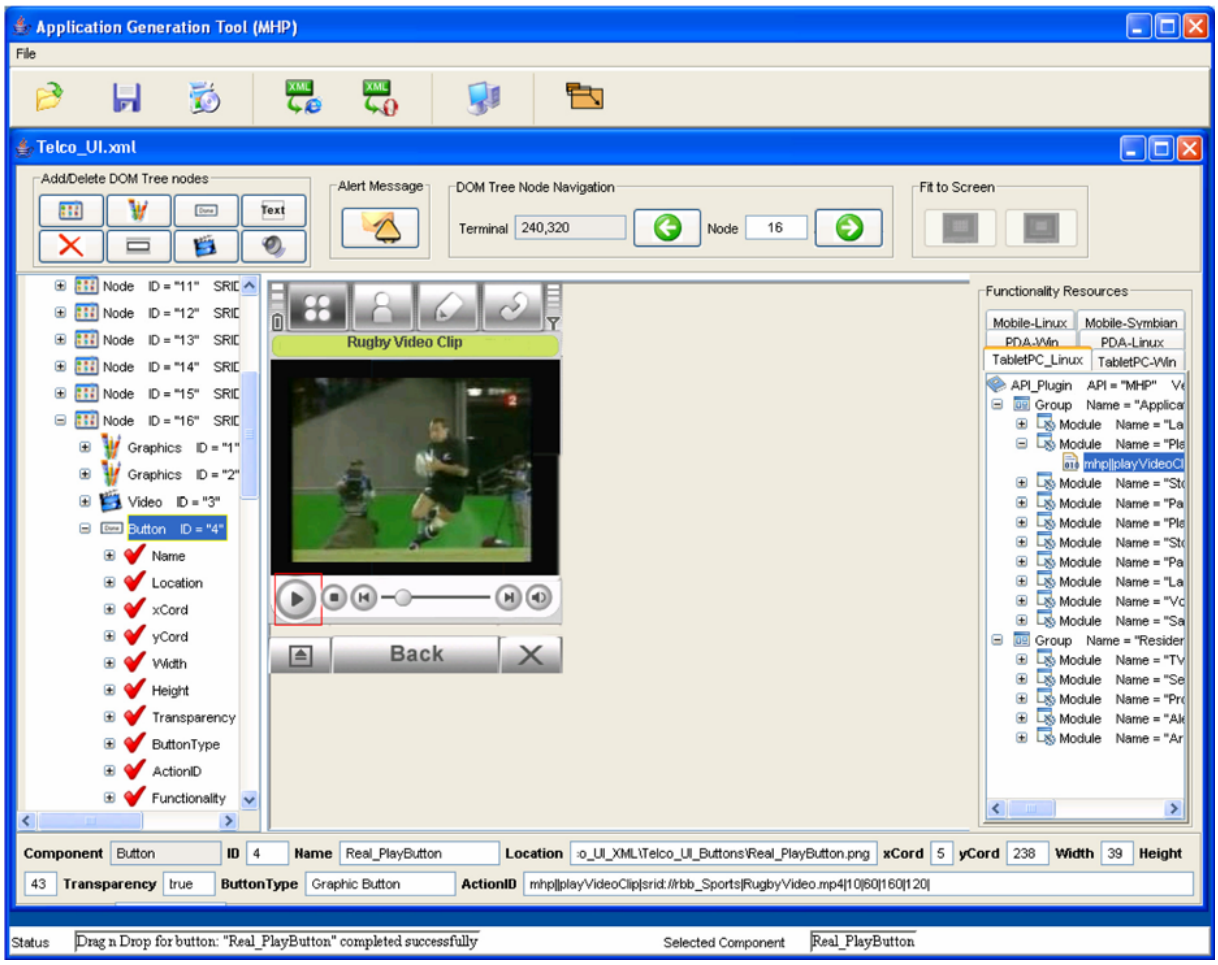

Figure 4: INSTINCT User Interface & Application Generation Model

Once all the required functional modules are linked into the specified UI components, the tool stores these completed strings in the protocol defined above under the $< ActionID >$ element. Then by performing a scan of the assigned commands from these tags it copies the corresponding classes to a number of Java files (files with .java extension) in order to build an Xlet (MHP application) or a Midlet (MIDP application) that will run these commands when run in the end-user terminal.

When the mapping is finished the UI description stored in the XML file can be transformed into HTML using the W3C XSLT language. This transformation serves as a simulation and visualisation of the XML file and evaluation tool of the UI in end-user terminals, which do not have the processing power to run Director for the user trials. In addition such transformations permit the usage of different rendering subsystems on the terminal side, such as an HTML browser-based rendering subsystem as well as using a traditional graphics processing and XML parser to render the UI.

This tool allows services to be simulated on target end-user terminals that have the same look-and-feel as the final service without going through the expensive process of commissioning the service on the broadcast and cellular network delivery platforms.

In addition this approach reinforces platform independency, since it allows the delivery of a common UI with terminal specific functionality. For example the same UI (front-end) can be used on set-top boxes running supporting different middleware solutions (e.g. MHP [Pro03], OpenTV [Cor05], OCAP [MSC05] etc.) and running on different operating systems (e.g. Windows, Linux).

Thus, by reusing code in an easy-to-use, time and code-efficient manner the Application Generation tool can semi-automate the process of writing applications

for a range of terminals and platforms.

7 Design & Application Generation of Prototype and Simulated Services

The user interface generation tool and the application generation tools were used to create both broadcast and telecommunications (Telco) service provider oriented services and applications for a wide range of terminal types during the INSTINCT project and these were demonstrated in the International Broadcast Exhibition (IBC) in 2005 [IBC07].

Figure 5 provides screenshots of the Telco-oriented services. These are GUI prototypes that have been created and simulated in Macromedia Director and then employing the UI and application generation tool have been converted to MIDP applications that run on a mobile phone terminal.

Since the type of terminals (mobile phone, PDA, tablet PC) that can be used for consuming mobile services have a wide range of screen sizes their user interface designs can be quite different. The limited screen size available for mobile phone means that the ESG is designed exclusively onto a separate page, as shown in figure 5a, whereas the more extensive screen size available on a tablet PC means that the ESG can be presented simultaneously with the media services, as shown in figure 6a.

These different designs were made available by the User Interface Generation tool, which provides all the graphics and an XML description of the UI required by the Application (Xlet) to dynamically construct the front-end.

For terminals that have limited resident storage capacity such as mobile phones, the ESG can be interpreted and presented onto the terminal by applications created by the application generation tool that are embedded within the terminal (e.g. ESG manager, service applications). Alternatively for terminals that have a large amount of resident storage capacity, the ESG can be interpreted and presented onto the terminal by applications created by the application generation tool that have been downloaded onto the terminal. Figure 5a shows a typical telecom operator's ESG presented on a mobile phone whilst Figure 6a shows a typical broadcast operator's ESG presented on a tablet PC at the same time as service content because there is sufficient space on the screen.

(a) Telecom service provider's ESG

(b) TV AV Program on a mobile phone

(c) AV Service on a mobile phone

(d) Image Service on a mobile phone

Figure 5: Telco Scenario Service Screenshots

All terminal types have the capability of playing audio/visual content that has been downloaded onto the terminals or streamed either across the DVB-T/H broadcast IP network or through the cellular radio mobile phone unicast IP network. The telecommunication service scenario is motivated to streaming audio/visual services on the IP over DVB-H network, since there is not sufficient storage space on mobile phones or efficient IP multicasting capabilities over cellular radio networks or sufficient space for a large enough battery within the mobile phone to support continuous DVB-T transmissions. A TV program multicast on IP over DVB-H is shown in Figure 5b and a streamed AV service on IP over DVB-H is shown in Figure 5c.

The broadcaster service scenario is motivated to continue to stream TV programs on the stream sections over DVB-T whilst offering additional audio/visual or text based services complementing the "traditional" streamed DVB-T/H broadcast services. These additional services combine downloading or streaming on the IP over DVB-T/H with using the unicast cellular radio network as an interactive "return and delivery channel". A TV programme broadcast on the DVB-T stream section is shown in Figure 6b while an additional audio/visual service streamed on IP over the unicast cellular network is shown in Figure 6c. TV programmes cannot be observed at the same time as additional services because user validation has shown that viewers find this disturbing. The application generation tool specifies the classes that play this AV content whereas the URI is specified on the ESG.

All terminal types have the capability of pulling additional Internet, image or graphical content over the cellular radio mobile phone unicast IP network. The telecommunication service scenario is motivated to pulling content snippets since it does not have the processing power and storage capability to support a full Internet browser. An image service pulled over the IP unicast cellular network is shown in Figure 5d. The broadcast service scenario pushes basic content via the DVB network and uses the IP cellular network to pull special types of content according to user's individual wishes/interests. Figure 6f and Figure 6d shows Text and Internet page services pushed over the IP unicast cellular networks on which there is hypertext URLs to access further media over the unicast network. Again the application generation tool specifies the class that access the web browser.

8 Conclusion

It is clear that with the introduction of a wide range of broadband networks within the mobile domain there will be a large demand for high quality interactive services to wide range of different terminal architectures. The construction and maintenance of such services require significant service provider resources if the services are developed using traditional methods. This paper has proposed a new service development methodology that encompasses a number of tools that aim to reduce this cost by rapid prototyping and semi-automatically generating the applications from a service scenario. In particular, the semi-automatic application generation tool provides a means to cope the logistics of managing the huge variety of screen sizes, applications, middleware APIs, operating systems and network interfaces that are within today's mobile terminals. Furthermore, it has illustrated how these tools are placed within the context of a service provisioning architecture to develop a vast range of different services. Therefore, we believe that these tools represent a significant cost saving in the development of applications/services for consumer converged broadcast and Telco mobile terminals. A second phase is being planned to further enhance the tool by making it DVB-CMBS ESG compliant as well as increasing the terminal/platform range by implementing an SVG and PCF output that will complement the functionality of the existing tool.

9 Acknowledgements

The authors gratefully acknowledge the support for this work that is funded by the EU under the IST program as the project INSTINCT (IP-based Networks, Services and Terminals for Converging systems), IST-2003-507014. The author Emmanuel Tsekleves gratefully acknowledges the financial support offered by the Public Benefit Foundation Alexander. S.Onassis for his PhD programme.

References

[ACC+07] Fabio Allamandri, Sebastien Campion, Angelo Centonza, Alex Chernilov, John P. Cosmas, Annette Duffy, David Garrec, Michel Guiraudou, Kannan Krishnapillai, Thierry Levesque, Bertrand Mazieres, Ronald Mies, Thomas Owens, Michele Re,

Emmanuel Tsekleves, and Lizhi Zheng, *Service Platform for Converged Interactive Broadband Broadcast and Cellular Wireless*, IEEE Transactions on Broadcasting **53** (2007), 200–211, ISSN 0018-9316.

[An02] Mir Farooq Ali and Manuel A. Pérez-Qui nones, *Using task models to generate multi-platform user interfaces while ensuring usability*, Proceedings of ACM CHI 2002 Conference on Human Factors in Computing Systems, 2002, ISBN 1-58113-454-1, pp. 670–671.

[APB+99] Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster, *UIML: An Appliance-Independent XML User Interface Language*, Proceedings of the Eighth International World-Wide Web Conference, 1999, www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html, last visited July 20th, 2007.

[BCH+06] R. Bradbury, R. Cartwright, J. Hunter, S. Perrott, and J. E. Rosser, *Portable content format: a standard for describing an interactive digital television service*, R&D White Paper WHP 134, British Broadcasting Corporation, may 2006, www.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP134.pdf, last visited July 20th, 2007.

[Car00] John Millar Carroll, *Making use : scenario-based design of human-computer interactions*, MIT Press, Cambridge, Mass., 2000, ISBN 0-262-03279-1.

[Cor05] OpenTV Corp., *Open tv*, www.opentv.com/, 2005, last visited July 20th, 2007.

[CR92] John M. Carroll and Mary Beth Rosson, *Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario*, ACM Transactions on Information Systems **10** (1992), no. 2, 181–212, ISSN 1046-8188.

[Gro03] Digital TV Group, *Digital Terrestrial Television MHEG-5 specification, version 1.0.6*, www.dtg.org.uk/, 2003, last visited July 20th, 2007.

[Har04] Harmonia, *LiquidUI tool*, www.harmonia.com/, 2004, last visited November 7th, 2005.

[HAV04] HAVi, *Home Audio Video interoperability*, www.havi.org, 2004, last visited July 20th, 2007.

[IBC07] IBC, *Ibc 2007 conference and exbihition*, www.ibc.org, 2007, last visited July 20th, 2007.

[Ind06] IndiaTimes, *Nokia plans 40 new models this year*, http://infotech.indiatimes.com, 2006, last visited July 20th, 2007.

[ISO95] ISO/IEC, *Information Technology - Generic Coding of Moving Pictures and Associated Audio: Part 2 - Video*, ISO/IEC 13818-2: 1995 (E) Recommendation ITU-T H.262 (1995 E), 1995.

[Kra96] C. J. Kraal, *Developing for interactive services in digital TV*, International Broadcasting Convention, Conference Publication No. 428, 1996, ISBN 0-85296-663-6, pp. 230–235.

[Mic06] Sun Microsystems, *MIDP: Mobile information device profile*, java.sun.com, 2006, last visited July 20th, 2007.

[MSC05] Steven Morris and Anthony Smith-Chaigneau, *Interactive TV standards : a guide to MHP, OCAP, and JavaTV*, Elsevier, Amsterdam, 2005, ISBN 0-240-80666-2.

[Pro03] MHP Project, *MHP: the multimedia home platform*, http://www.mhp.org, 2003, last visited July 20th, 2007.

[Pro05] DVB Project, *A090 - DVB portable content format (PCF) commercial requirements*, www.dvb.org/, 2005, last visited July 20th, 2007.

[PRS02] Jennifer Preece, Yvonne Rogers, and Helen Sharp, *Interaction Design: Beyond Human Computer Intraction*, ch. Chapter 8: Design, Prototyping and Construction, pp. 239–278, Wiley, New York, NY, 2002, ISBN 0-471-49278-7.

[RW07] Antti Rantalahti and Ivan Wong, *Jsr 272: Mobile broadcast service api for handheld terminals*, Tech. report, Java Community Process, 2007, jcp.org/en/jsr/detail?id=272, last visited July 20th, 2007.

[SDP+04] Jean-Luc Sicre, Annette Duffy, Raquel Navarro Prieto, Marcello Otte, John Cosmas, Emmanuel Tsekleves, Michele Re, Veronique Leturcq, and Ronald Mies, *Three user scenarios on the joint usage of mobile telco and TV services for customers on the move*, 12th Wireless World Research Forum Conference WWRF (Toronto, Canada), 2004.

[SE03] Jon Sigurdson and Peter Ericsson, *New services in 3G new business models for streaming and video*, International Journal of Mobile Communications **1** (2003), no. 1-2, 15–34, ISSN 1741-5217.

[Sys06] Adobe Systems, *Macromedia director mx*, www.macromedia.com/software/director, 2006, last visited July 20th, 2007.

[W3C03] W3C, *Scalable vector graphics*, www.w3c.org, 2003, last visited July 20th, 2007.

[W3C04] W3C, *The extensible stylesheet language family (xsl)*, www.w3.org/Style/XSL/, 2004, last visited July 20th, 2007.

(a) Broadcaster service provider's ESG

(b) TV AV Program on a tablet PC

(c) AV Service on a tablet PC

(d) Video & text content on a tablet PC

Figure 6: Appendix

Citation

Emmanuel Tsekleves and John Cosmas, *Semi-Automated Creation of Converged iTV Services: From Macromedia Director Simulations to, Services Ready for Broadcast*, Journal of Virtual Reality and Broadcasting, 4(2007), no. 17, September 2006, urn:nbn:de:0009-6-10862, ISSN 1860-2037.