UIML Based Design of Multimodal Interactive Applications with Strict Synchronization Requirements

Philip Leroux, Vincent Verstraete,
Filip De Turck, Piet Demeester
Ghent University – IBBT
Department of Information Technology (INTEC)
G. Crommenlaan 8 bus 201
9050 Ghent, Belgium
philip.leroux@intec.ugent.be

Kristof Thys, Kris Luyten
Hasselt University - IBBT
Expertise Centre for Digital Media (EDM)
Wetenschapspark 2
3590 Diepenbeek, Belgium
kris.luyten@uhasselt.be

Abstract - As the variety in network service platforms and end user devices grows rapidly, content providers must constantly adapt their production system to support these new technologies. In this paper, we present a middleware platform for deploying highly interactive (television) applications over a diverse collection of networks and end user devices. As the user interface of such interactive applications may vary depending on the capabilities of the different target devices, our middleware uses UIML for the description of generic user interfaces. Our middleware platform also provides a pluggable support for new networks. A factor that highly complicates the design is the need for strict synchronization between an interactive application and video or audio data that is broadcasted. In order to support a maximum of functionality, downloadable application logic is used to provide the interactive services. As a test case, an evaluation setup was built, targeting both set-top boxes and mobile phones.

UIML; MHP; Interactivity; DVB-H; Synchronization; IP Datacast; Middleware Architecture

I. INTRODUCTION

Two trends are clearly visible for the current consumption of digital information. On the one hand, the number and diversity of digital services is rapidly increasing as interactive applications are added to the digital multimedia. On the other hand, more and more telecommunication and broadcast networks are being built up. These networks may have very different characteristics (low vs. high bandwidth, unidirectional vs. bidirectional, reliable vs. best-effort, etc.). This huge variety in available networks and related business models has also led to significant differences in end-user devices. These range from fixed devices such as high-end PCs or set-top boxes to wireless, portable or handheld devices. All these devices have very different characteristics in terms of screen size, network interface, user interaction model, etc. This diversity exists nowadays and is expected to keep growing. This has a great influence on content providers as it will become more and more difficult to continuously adapt their production system to the increasingly complex

world of heterogeneous digital networks and the associated service platforms.

The MCDP project [1] was defined to study the possible architectures for a multimodal multimedia content distribution system that can cope with this complexity today and in the near future. One of the main questions in this project was how to preserve interactivity and the associated synchronization, defined once at the content provider and delivered to a multitude of end-user devices over a variety of networks. The need for strict synchronization is for example found in the digital TV world where interactive applications need to be tightly synchronized with the video data, e.g. during an interactive quiz show. The authors of this paper, affiliated with the Expertise Centre for Digital Media (EDM), focused on the HCI-related challenges, building their own UIML renderers for providing user interface independence. The authors, affiliated with the Department of Information Technology (INTEC), developed a middleware layer for the provisioning of network independence while keeping tight synchronization with the broadcasted video data. In this paper we describe the general architecture that was developed combining the work of both research groups.

Strongly related to our research are the standardization efforts in different domains. One of the most important standards with respect to synchronized interactivity is the MHP specification [2]. In our middleware platform, we were as compliant as possible with this standard. When it comes to generic user interfaces, two approaches are common: model-based user interface development (MBUID [3]) or High Level User Interface Description Languages (HLUID) such as UIML [4]. In this middleware solution, we focused on the integration of UIML in order to describe a device independent user interface.

As a proof of concept, we developed an interactive quiz application. We also adapted our middleware platform for supporting an IP Datacasting system, combining a broadcast network (DVB-H) with a bidirectional unicast network (UMTS).

The rest of the paper is organized as follows. Section II gives an overview of the server-side system architecture of the implemented middleware. Section III introduces the use case that was chosen to illustrate the applicability of the platform and section IV describes the evaluation architecture for this use case. In section V, the results are presented and section VI shortly explains the IP Datacast set-up integration. Finally, section VII states our conclusions and future work.

II. SYSTEM ARCHITECTURE

A. Server-side Middleware Architecture

The functional design of the server-side middleware component is presented in Fig. 1. Content is ingested by the Content Provider and has to conform to the Scalable Presentation Format (SPF [5]), an MPEG-21-based format that consists of four basic parts:

- Multimedia - The video & audio content of the television or radio program.
- App - An interactive application, possibly synchronized with the television program.
- UIML - The user interface of the interactive application, where every page is described with UIML.
- Metadata - The glue to bind these three elements together.

The interactive services are handled by application logic that has to be sent to, and installed on the end user's device. The main advantages of using downloadable applications are the support of a maximum of functionality and the unlimited number of use cases that may be implemented. In this version of the platform, Java application logic is used.

The content that is ingested by the content provider, is saved in a database, and retrieved when the Electronic Programming Guide (EPG) module indicates that the specified television program is to be broadcasted. In this EPG module, the EPG of each channel, generated by the content provider or the service provider, is stored.

When a new program starts, several other functional blocks come into play. The Conversion modules convert the multimedia, applications, events and triggers to a suitable format for the targeted devices, thus providing the user-device independence of both the multimedia content and the interactivity related data. An event is the description of a specific action that can be executed by a downloaded application. A trigger will inform the application logic on the client device when a specific event needs to be executed. There are two types of triggers: Do-It-Now (DIN) event triggers, where the associated event occurs immediately, and Scheduled Event (SE) triggers.

Figure 1: The Server-side middleware architecture.

SE triggers contain a clock value that indicates when the event exactly has to occur. They are sent in advance, and possibly more than once. Based on these SE triggers, tight synchronization with the video data is possible.

By analyzing the information of the EPG, the Scheduler of each channel knows exactly for which television program it must generate interactivity descriptors. First, all the associated interactive applications, the events and their occurrences (SE triggers) are looked up in the database, and descriptors for this information are periodically scheduled. Second, a synchronized time line is built up for the television program, based on the fixed time descriptors associated with the television program. These time descriptors represent discontinuities such as pausing and are also periodically scheduled. Third, intermediate time descriptors are generated. This timeline mechanism is described in section II.D.

The Return Channel Manager collects and processes all the incoming return data and finally the Network Module sends the multimedia, the applications and the generated descriptors over the network to the clients. There are Network Modules implemented for supporting both unicast and broadcast scenarios.

B. Client Architecture

On the end-user device, a middleware framework should be installed that can handle all the extra functionality that is provided through the server-side middleware. The implementation of the client middleware may vary based on the terminal's capabilities, however the general architecture and functionality should be the same on all terminals. As already mentioned, Java application logic is used to provide the interactive services. This requires the dynamic installing and management of Java software components on a device, which is exactly where OSGi comes into play. Our client middleware layer is thus developed on top of the OSGi framework, while using the provided functionality of the OSGi framework in order to install and manage extra Java application logic on the end user's device. The generic client's middleware architecture is shown on Fig. 2.

The automatic initialization, starting, stopping and destroying of the Applications is handled by the Application Manager. The Event Manager provides the dynamic definition of events, and the delegation of these events to the correct Applications. The Return Channel Manager provides an interface for the Applications to the (IP) Unicast Return Channel module (if available). The Display module renders the UIML pages on the screen and the User Input module handles the user input. The Timeline module is used in order to synchronize the Scheduled Events with the broadcasted media content. Finally, a Context Manager module is implemented to store the current context (which channel is currently watched, which applications are running, etc.). The following sections detail the user-device interaction and the synchronized event triggering mechanism.

Figure 2: The client middleware architecture.

C. User-Device Interaction

The User Interface Markup Language (UIML) is used to support user-device interaction on a variety of devices and platforms with a minimum of effort from the content or application provider. UIML is a high-level XML-based user interface description language that facilitates user interface creation and portability. A UIML document consists of four distinctive parts that allow to specify different aspects of the user interface independently:

- *structure*: defines the different parts of the user interface and their hierarchical structure.
- *style*: describes properties of the parts defined in the structure. These properties can be both device specific properties (use device specific capabilities) or generic properties (reusable for a range of devices).
- *content*: separates the content of the interface from the other parts, e.g. to create language independent user interfaces.
- *behavior*: defines a rule-based system with actions that are triggered when a certain condition is met.

In order to generate the actual user interface, the UIML document is linked to a vocabulary. This vocabulary defines how generic user interface parts (button, label) are mapped to widgets from a specific widget set (JButton, JLabel). This mapping is done on each end device by a UIML renderer that exactly knows the characteristics of the device it is running on. This mapping and the generation of the device specific UIML renderers is further described in section II.D.

The main benefit of this approach is its flexibility: since the user interface is not hard-coded on the device and the application logic is invoked remotely, new consumer devices that become available are easily integrated (e.g. new types of mobile phones or set-top boxes). The content provider can also update the user interface without requiring the user to install software updates. This flexibility is exactly what is required by our platform.

D. Synchronized Client-Server Interaction

In order to support full-fledged interactivity, the client framework must be able to dynamically load and unload applications, and to dynamically specify event types to which these applications can react. This allows for the most basic form of interactivity: DIN Event triggering. An occurrence of an event has to be triggered. In the DIN case, such a trigger is generated once on the server, sent over the network and processed by the client, resulting in the event occurrence. This is a best-effort system, but provides no guarantee on the timely occurrence of the event. DIN triggering is sufficient for a wide range of services, and is the degree of interactivity that quite a few interactive television providers support. If,

however, we want timely and possibly frame-accurate occurrence of events to be guaranteed, we must go one step further, and relate the triggers to a timeline that is linked with the multimedia timeline. This can either be the multimedia timeline itself (e.g. the System Time Clock or STC in MPEG-2) or a derived timeline.

A derived timeline that is used in several widespread specifications (e.g. MHP and OCAP) is the Normal Playing Time (NPT). This notion is defined in the DSM-CC extensions to MPEG-2 [6]. The NPT is based on the STC but adds an extra level of abstraction. While the STC always increments by 1, the NPT can be used to reflect pausing, fast forwarding, rewinding and discontinuities in the multimedia stream.

At the client side, the NPT timeline is built and maintained based on NPT descriptors. In a non-broadcasting environment, it is sufficient to provide a descriptor for every change (e.g. resuming a paused NPT). In a broadcasting environment however, it is not known when the user will tune in on the channel, thus intermediate descriptors have to be generated by the server. If not, an NPT will only be started at the next change. The finer the granularity of the intermediate descriptors, the sooner the user will be able to enjoy the associated interactive services.

In order to add synchronized interactivity to our platform, we have chosen to use the NPT, and to relate the SE Triggers to this timeline. The Scheduler generates NPT descriptors, both predefined and intermediate, and Scheduled Event Triggers. These descriptors and triggers are then sent to the different clients over the connected networks. It is important to note that the choice to use the NPT standard is not limiting. When support is needed for another synchronization method, the generated NPT descriptors can dynamically be translated.

III. USE CASE DESCRIPTION

The final goal of our middleware was an interactive, synchronized user experience over several devices and networks. As a proof-of-concept we selected a quiz application. In this quiz format, users are able to participate in an existing television show ("Test the nation") using digital television or a mobile device which allows the participants to test their I.Q. based on questions asked by a quiz master.

In our test case, this quiz application was reused and extended as synchronization of the questions with the television show is added to the picture. When the quiz master asks a specific question, a dynamic overlay interface (rendered at the user's device) is presented to all the viewers who want to participate. Depending on the target device, this interface may contain the possible answers, a repetition of the question, extra related media, etc. The viewer can then answer the questions by pressing on the correct color-button of

their remote control or by selecting the correct answer on the screen. Synchronized with the quiz master giving away the correct answer, all the participants receive an evaluation of their answer at the same time. Note that only the correct answer has to be sent to the end user's terminal, as the evaluation of the user's answer is evaluated by the interactive application logic that has been downloaded to the end devices. The same application logic also keeps track of the user's score and finally the return channel is used in order to give the user feedback on how well he performed in relation with the other participants.

During this use case, targeting two mutual very different end user devices, all the necessary synchronization parameters had only to be defined once as the server-side middleware handled all the device dependencies. Furthermore, the effort of generating different user interfaces for all the supported end-user devices is reduced by using a User Interface Description Language (UIDL). These two extensions make the synchronized define-once-play-everywhere scenario possible.

IV. EVALUATION SETUP

In order to evaluate our platform, we plugged in two very different types of clients. The first client was a set-top box (STB), connected through an IP multicast network over Ethernet and the second type of client was a mid-end mobile phone (GSM) with constrained resources. Interaction with the server is established on this phone via an IP unicast connection over UMTS.

On neither of these platforms, middleware software was available that supports synchronized interactivity nor UIML. Therefore, on the set-top box we developed an OSGi based framework in Java that supports the NPT. We also implemented a similar (but currently non-OSGi) framework on the mobile phone using the J2ME technology, more specifically the MIDP 2.0 Profile. Concurrently, a device independent UIML renderer was developed [7]. Combined with the device-specific widget sets, this allows us to render UIML pages on both the set-top box and the MIDP mobile phone. For both systems, we also integrated an emulator in the evaluation architecture. This allowed us to monitor device-specific influences.

Finally, a multimedia server is used to stream the audio and video content to the set-top box and its emulator. The synchronization between our server-side middleware and the multimedia server is done by means of the Network Time Protocol (NTP) [8] and a common EPG. Our evaluation setup is shown on Fig. 3. Fig. 4 shows how the quiz applications finally looked on both the end-user devices. As you can see, the rendered content is specifically adapted to the hardware parameters of the end device (of which the screen size is definitely one of the most important parameters).

Figure 3: Evaluation setup.

Figure 4: Screenshots of the quiz application, running on the set-top box (a) and on the mobile phone (b).

V. EVALUATION RESULTS

When deploying the interactive quiz on the evaluation architecture, we measured two delays that have a big influence on the synchronization: DIN delay & Rendering time. The results of these measurements are presented in Fig. 5. SE triggers are being sent in advance to the client devices, where they are resolved and added to the NPT timeline. When their trigger point is reached, they can be delegated almost instantly to the application. This is not the case for DIN triggers. When generated at the server, they have to be sent over the network, before being analyzed and resolved. We can make an abstraction of the network delay as the multimedia will also suffer of some network delay. As our graph shows, network delay has a rather limited influence.

Figure 5: Rendering and DIN delays.

The rendering time is the amount of time it takes for the UIML renderer to build and display a page on screen. We have calculated this time span for all the quiz and result UIML pages, and set out the mean value for the four types of clients. As can be seen, rendering may take up to 1-2s, especially on devices with limited resources. Another factor that comes into play is the complexity of the UIML pages. The pages on the set-top box and emulator contained a large background image (see Fig. 4), which is why they took longer to render compared to the mobile phone and its emulator. 1-2 seconds compared to 25-50 frames (on a 50Hz PAL system), so extensive measures (caching, rendering in advance, etc.) will have to be taken in order to support (nearly) frame-accurate interactivity.

VI. IP DATACAST INTEGRATION

The Digital Video Broadcast-Handheld (DVB-H) standard [9] provides an efficient way of carrying multimedia services over digital terrestrial broadcasting networks to handheld terminals. IP Datacast (IPDC) over DVB-H foresees the integration of DVB-H in a hybrid network structure consisting of both a mobile communications network such as UMTS and an additional DVB-H downstream. As our middleware was designed with network and device independence in mind, it forms an ideal basis for the integration of synchronized, interactive services in a IPDC set-up. Only some IPDC specific issues have to be taken care of. The most important issue to be resolved, is related to the synchronization mechanism.

As a standard IPDC protocol stack prescribes the embedding of audio and video into Real-Time Transport Protocol (RTP) [10] packets, a dynamic translation of the NPT descriptors to the RTP timestamp values had to be made. This mapping [11] is done by analyzing the output (i.e. the RTP streams and the related RTCP Sender Reports) of the encoders that are located in a typical IPDC headend. Thus, the most important component that was plugged into the existing middleware framework is a component that analyzes the output of the DVB-H headend's encoders in order to remap our middleware's synchronization mechanism to the RTP timestamp values of the multimedia content that is broadcasted over DVB-H.

Besides this Conversion module, 2 new Network modules are plugged in. The first Network module is a typical broadcast module for sending the broadcast television programs and all the popular interactive services that are broadcasted via the DVB-H network. Note that these interactive services are sent to the user by a data carousel mechanism. The second Network module is a bidirectional unicast module that is responsible for sending additional services to the same clients, but this time over a UMTS connection. This module also provides the interface for the return channel. The client middleware, installed on the DVB-

H terminal, does not differ from the client middleware architecture that was introduced in section II.B.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a middleware framework for the efficient distribution of interactive applications to a variety of end-user devices using heterogeneous network infrastructures. UIML is used for providing user interface independence while preserving the synchronization between applications and the associated multimedia is based on the usage of an event based trigger mechanism. The implementation of a proof-of-concept quiz application illustrated the feasibility of this platform and allowed to evaluate the performance of the platform. In the near future, research will be done on how to reduce this rendering time (caching, pre-rendering, etc.) and adapt it to the requirements of different applications.

To demonstrate the pluggable support of new technologies and protocol stacks, we also integrated the IP Datacast specification into our middleware. Only few extensions to the existing middleware were made in order to support this new set-up. As DVB-H is especially targeting mobile devices, we will also focus more on the client middleware integration on mobile devices. As MIDP 2.0 is not fully capable of adding new application logic to an already installed framework, we will focus on new technologies such as the embedded Rich Client Platform (eRCP) [12].

ACKNOWLEDGEMENTS

The authors would like to thank Nokia Siemens Networks for the definition of the MCDP and the MADUF[13] projects, which allowed us to do this research. Part of the research at EDM and IBCN is funded by EFRO (European Fund for Regional Development), the Flemish Government, IBBT (Interdisciplinary institute for Broadband technology) and the partners in the MCDP and MADUF projects.

REFERENCES

- [1] MCDP, Multimedia Content Distribution Platform, <http://projects.ibbt.be/mcdp/>, 2008.
- [2] J. Piesing: The DVB Multimedia Home platform (MHP) and Related Specifications. *Proc.of the IEEE*, 2007, 94(1).
- [3] F. Paterno, C. Santoro: A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms. *Interacting with Computers*, 15(3) p349-366, 2003.
- [4] M. Abrams, J. Helms: User Interface Markup Language (UIML) Spec. v3.1. Technical report, Oasis UIML TC, 2004.
- [5] D. Van Deursen, F. De Keukelaere, L. Nachtegaele and J. Feyaerts, "A scalable presentation format for multichannel publishing based on mpeg-21 digital items." *Lecture Notes in Computer Science*, vol. 4105, p650-657, 2006.
- [6] ISO/IEC 13818-6 International Standard: Generic Coding of Moving Pictures and Associated Audio: Digital Storage Media Command and Control. Technical report, 1996.
- [7] K. Luyten, K. Thys, J. Vermeulen and K. Coninx: A Generic Approach for Multi-Device User Interface Rendering with UIML. *Computer-Aided Design of User Interfaces*, p175-182, 2006.
- [8] Network time protocol specification, implementation and analysis (v3), RFC 1305. ETSI EN 302304 V1.1.1, 1992.
- [9] Digital Video Broadcasting (DVB), Transmission System for Handheld Terminals (DVB-H), 2004-11, 2004.
- [10] IETF RFC 3550: RTP, A Transport Protocol for Real Time Applications, 2003.
- [11] P. Leroux, V. Verstraete, F. De Turck and P. Demeester: Synchronized Interactive Services for Mobile Devices over IPDC/DVB-H and UMTS. *Proc. IEEE Broadband Convergence Networks*, p 172-183, Munchen, 2007.
- [12] G. Ercan, Embedded Rich Client Platform (eRCP) 1.0 - Release Review, 2006.
- [13] MADUF, Maximizing DVB Usage in Flanders, <http://projects.ibbt.be/maduf/>, 2008.

The Second International Conferences
on Advances in Computer-Human Interactions

ACHI 2009

*1-7 February 2009
Cancun, Mexico*

Editors/Chairs
Sergiu Dascalu
Ivan Poupyrev

CONFERENCE INFORMATION

PAPERS BY SESSION

PAPERS BY AUTHOR

GETTING STARTED

TRADEMARKS

SEARCH

Sponsored by

Published by

IEEE
computer
society

The Second International Conferences on Advances in Computer-Human Interactions (ACHI 2009)

1-7 February 2009
Cancun, Mexico

Editors/Chairs
Sergiu Dascalu
Ivan Poupyrev

Product Number E3529
BWS Part Number CFP09720-000
ISBN 978-0-7695-3529-8

Electronics Engineers, Inc. All rights reserved.