PUBLICATION P7

M. Honkala, P. Cesar, and P. Vuorimaa. A device independent XML user agent for multimedia terminals. In *Proceedings of the 6st IEEE International Symposium on Multimedia Software Engineering, IEEE-MSE2004*, Florida, Miami, December 13–15, 2004, pages 116-123. IEEE.

A Device Independent XML User Agent for Multimedia Terminals

Mikko Honkala
Telecommunications Software and Multimedia Laboratory
Helsinki University of Technology
P. O. Box 5400,
FIN-02015 HUT
Tel. +358-9-451 4794
honkkis@tml.hut.fi

Pablo Cesar
Telecommunications Software and Multimedia Laboratory
Helsinki University of Technology
P. O. Box 5400,
FIN-02015 HUT
Tel. +358-9-451 4794
pcesar@tml.hut.fi

Petri Vuorimaa
Telecommunications Software and Multimedia Laboratory
Helsinki University of Technology
P. O. Box 5400,
FIN-02015 HUT
Tel. +358-9-451 4794
petri.vuorimaa@hut.fi

Abstract

Interactive multimedia software capable of running on different devices, such as smartphones or digital television receivers, will be a consumer expectation in the near future. Supporting each device individually is too expensive. Therefore a common platform is needed. In this paper, a device independent platform for multimedia applications is presented. It is based on XML and Java and takes into account the varying features of the targeted devices. The platform consists of an XML based language profile, capable of supporting networked multimedia applications. The language profile is based on Synchronized Markup Integration Language (SMIL) and XForms. For instance, minimal interaction multimedia presentations can be developed using SMIL, whereas a hybrid SMIL and XForms document could provide real interactive multimedia services that allow connection to a server side process. The platform is written in Java and includes a user interface compatibility layer, which can be run on top of different graphics Java Application Programming Interfaces (APIs) depending on the targeted platform.

1 Introduction

One of the biggest current challenges in software engineering is to develop cross-platform applications. This does not only mean software, which can be run on different Operating Systems, but one that can also be used on different end-user terminals [1]. As Myers has stated [2]: "We are at the dawn of an era where user interfaces are about to break out of the desktop". Not so long ago, the Personal Computer (PC) desktop was the only platform software developers had in mind. Nowadays, the number of targeted devices is increasing (e.g., mobile phones and digital television receivers).

The best alternative for the distribution of device independent multimedia applications is to use higher abstraction level tools. For example, platform independent services can be developed using eXtensible Markup Language (XML) based languages. In this case, the selection of an appropriate language profile to develop multimedia applications becomes an essential issue. This language profile should permit, at least, to handle user interaction, and to synchronise and present multimedia objects. Different multimedia languages, including SMIL, XHTML, XMT (MPEG-4), Flash, and MHEG have been studied extensively in [3][4].

After defining a valid language profile for developing multimedia services, the next step is to provide the needed cross-platform XML user agent (i.e., browser) for it. Each of the new multimedia platforms differs in their physical characteristics (e.g., input methods, processor, screen resolution) and in their graphics libraries. In this paper, all consumer devices are assumed to support Java, which is the most interoperable option available at the moment [5].

Current advances in Java technology are addressing its characteristic drawbacks: performance and size. For example, Project Monty[1], optimised K Virtual Machine (KVM) by applying the techniques used in HotSpot, promises a fast Java VM fitting in less than 1MB. Other relevant products include Savaje[2] and Nokia Communicator[3] hand-held device. The former is a Java based Operating System for advanced mobile phones, while the latter includes support for (Mobile Information Device Profile) MIDP and Personal Profile.

[1] http://www.jlocationservices.com/LBSArticles/Sun.ProjectMontyWhitePaper.pdf
[2] http://www.savaje.com
[3] http://press.nokia.com/PR/200402/935462_5.html

ISBN: 0-7659-2217-3 © 2004 IEEE

Java allows the porting of software to different devices, but even in this environment, the actual Application Program Interfaces (APIs) used for user interface development differ between multimedia terminals. While digital television receivers use Home Audio Video Interoperability (HAVi)[4] Level 2 User Interface, advanced mobile phones directly use Abstract Windowing Toolkit (AWT) as defined in Java 2 Micro Edition Personal Profile, and PC desktops computers normally use Swing.

Developing different versions of applications for each device is not feasible. Hence, in order to cope with different user interface graphics libraries, a generic or abstract graphical user interface library must be built. This implies an amount of effort in the development of the generic toolkit, but then porting only consists of providing once a correct back-end.

Fig. 1 depicts the studied concepts and proposed solutions in this paper. First, at the application layer we study how XML presentation and processing languages can be used to develop interactive applications. Since we are especially interested in interactive multimedia, we propose the use of Synchronized Multimedia Integration Language (SMIL) (i.e., presentation) and XForms (i.e., interaction) hybrid documents. Second, the proposed platform layer includes an interoperable XML user agent implementing the SMIL+XForms language profile. In order to support different device configurations, it has an interface between XML languages and underlying graphics toolkits by implementing a generic graphics user library (i.e., UI Compatibility Layer).

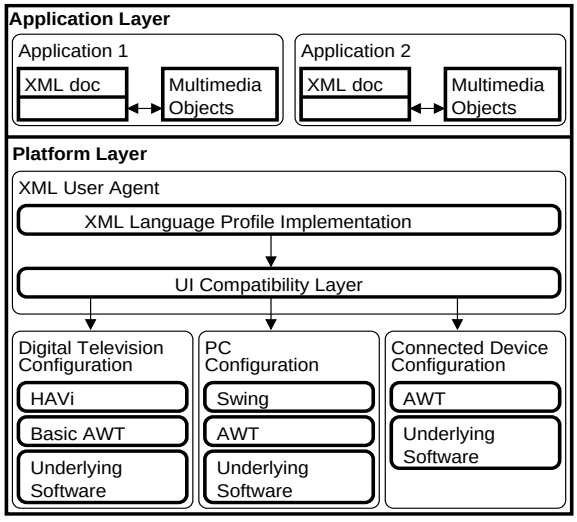

Fig. 1. Application and Platform Layers.

[4] http://www.havi.org

The paper is structured as follows. First, Section 2 identifies the requirements that a multimedia language profile and the underlying platform should meet (e.g., temporal dimension, spatial layout, continuous media support, and interaction). Then, Section 3 presents the experiences of developing an XML user agent for multimedia platforms meeting those requirements and its integration into an ongoing digital television receiver prototype platform. Next, Section 4 shows a case study, an interactive distance education portal that includes synchronised audio, video, and slides implemented completely in a declarative manner (i.e., no scripting is used) running on the platform, thus showing the benefits of the proposed solution. Finally, Section 5 presents the conclusions and future work.

2 Requirements Study

The problem studied in this paper can be divided into two subproblems: definition of a valid language profile for implementing interactive multimedia web applications and characterization of a device-independent platform supporting such language profile. First, the requirements related to the language profile and the platform support, respectively, are studied in more detail. Certain technology solutions are proposed analysing their benefits and limitations. Finally, this Section ends with a summary of the requirements in the form of a table.

2.1 Language Profile

The first task in providing XML based support for multimedia applications is to define a suitable XML language profile. For networked multimedia services there are at least the following requirements:

1. Temporal dimension: synchronization between the multimedia objects of the presentation.
2. Spatial Layout: defines where the multimedia objects in a presentation should be placed.
3. Multimedia Objects Support:
   a) *Continuous Media:* such as video, audio, and animations.
   b) *Discrete Media:* such as text, graphics, and images.
4. User Interaction:
   a) *Links:* selection of a path.
   b) *Validated Entry:* an user input to be validated. For example, the input could be a text, boolean, or a selection of one or more options. Simple calculations should also be supported.
   c) *Submission:* provide user entry to server-side process.

ISBN: 0-7659-2217-3 © 2004 IEEE

All requirements, but Validated Entry and Submission are fullfilled by SMIL 2.0 basic profile. Its main limitation is Spacial Layout, namely lack of flow layout, which could be solved by embedding eXtensible Hypertext Markup Language (XHTML). Other alternative for providing Temporal dimension, which is not further discussed in this paper, would be the use of Timesheets [6]. XForms, on the other hand, fits all of the other user interaction requirements. Since XForms is not intended as a self-standing document type, a host language is needed to provide the document layout. SMIL, for example, can act as host language. Given the previous requirements, it was decided to use SMIL+XForms as the language profile.

2.2 Platform

The requirements identified under platform support can be divided into the following:

5. Different Libraries: Today, Java is available in all the multimedia terminals listed above. The Java environments (e.g., Swing, AWT, and HAVi) are different in each terminal, though.
6. Different Input Mechanisms: the way the user interact with the system depends on the device at hand (e.g., digital television receivers use remote control).
7. Different Output Devices: each multimedia terminal includes a graphical display, but their actual characteristics (e.g., resolution and size) are particular of the device.
8. Different Capabilities: the configuration of each of the devices differ (e.g., processing power and power consumption). For example, devices such as low-end digital television receivers are unlikely to include a video player supporting other formats than MPEG-2. In some cases, video should even be replaced with still images.

In this paper, the platform is implemented in Java, since it is the most interoperable option at the moment. In addition, a UI compatibility layer was implemented to meet the requirements of different libraries and different input mechanism. Possible solutions for supporting different output devices include the use of Cascading Style Sheets (CSS) Media Queries, XSL Transformations (XSLT), or automated content adaptation tools. These approaches require the utilization of declarative languages. An extensive study on the topic can be found in [7]. The Different Capabilities requirement has been researched in [8], although in some simple cases CSS and XSLT are viable alternatives [3].

2.3 Summary

In order to provide a better visual understanding of this section, Table 1 summarises the technological decision taken for each identified requirement.

Table 1 . Requirements List and their Technology Solutions.

Requirements			***Technology***
Language Profile			
	Temporal dimension		SMIL
	Spatial Layout		SMIL (no flow layout)
	Multimedia Object Support		
		Continuous Media	SMIL
		Discrete Media	SMIL
	User Interaction		
		Links	SMIL
		Validated Entry	XForms Controls
		Submission	XForms Controls
Platform			
	Different Libraries		Java + UI Compatibility layer
	Different Input Mechanisms		Java + UI Compatibility layer
	Different Output Devices		CSS, XSL Transformations, or automated content adaptation
	Different Capabilities		CSS, XSL Transformations, or automated content adaptation

3 Implementation

In this Section, the implementation of the language profile and the platform is presented. The first author has done the XForms implementation, and the UI Compatibility Layer (i.e., ComponentFactory) presented in this paper. Additionally, he has actively participated in the W3C XForms Working Group. The XForms implementation was one of the reference implementations that enabled the Working Draft to become a Recommendation. The second author has implemented the digital television platform (i.e., Ubik) presented in this paper. He has also implemented the digital television graphical user interface toolkit, based on the HAVi specifications, completely. The integration of SMIL+XForms is based on the research reported in [9]

ISBN: 0-7659-2217-3 © 2004 IEEE

and [10]. The SMIL implementation used in this work is also based on previous work [11] . It is based on the SMIL 2.0 Basic Profile, but extends it with couple of modules, such as animation. Since SMIL in its basic form does not support CSS layout, some XForms features requiring flow layout were left out of this implementation. These are *repeat*, *switch*, and *group*. These three modules are described in the XForms 1.0 recommendation in chapters 9.1, 9.2, and 9.3, respectively [12]. Otherwise, the whole XForms specification is supported in our SMIL+XForms language profile. In the profile, XForms elements can be inserted into the SMIL host document in any place where content objects (e.g., image, video) are allowed. Some SMIL attributes, such as *@region*, are supported within the XForms elements.

The implementation is based on the X-Smiles browser [13], and therefore the core of the X-Smiles was ported to the different environments. Since all supported graphical user interface libraries are based on AWT in JDK 1.1.8 (or later), this porting meant removing all non-AWT (mostly Swing) and Java 2 dependencies from the browser.

3.1 SMIL implementation

The SMIL player used in the platform is based on the SMIL implementation of the X-Smiles browser. The player is designed so that it is possible to run it in various devices. The implementation architecture is depicted in Fig. 2.

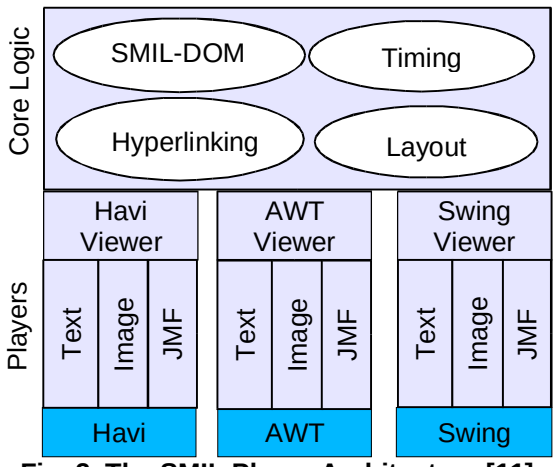

Fig. 2. The SMIL Player Architecture [11].

The player is divided into three layers. The Core Logic is the heart of the SMIL player. It handles timing, decides the layout of the presentation, resolves hyperlinks, etc. When a SMIL document is opened, it is parsed with an XML parser, which constructs a Document Object Model (DOM) tree. Each element in the tree knows its own behaviour, taking care of timing and layout. The Core Logic is completely GUI independent, making it possible to run it in any Java environment with any GUI framework.

The Core Logic layer utilises the Viewer layer to draw media on the screen according to the timing information. Various devices, e.g., desktop computer, digital televisions, and mobile phones use different kind of GUI frameworks. A viewer can be implemented for any possible GUI framework. So far, a viewer has been created for Swing, AWT, and HAVi. [11].

3.2 XForms implementation

The XForms implementation was developed for the X-Smiles browser by one of the authors [14] and it is included in the open source distribution of X-Smiles[5]. The implementation is not described in detail in this paper, but rather we concentrate on those features that enable its usage in different platforms and host languages.

Fig. 3 depicts the architecture of the XForms implementation. There are three layers: the XForms model, Meta UI, and the User Interface. The lowest level, the XForms model, uses Apaches Xalan and Xerces packages for XPath and XML Schema implementations, respectively. It also includes a calculation engine and the implementation of the instance data. The middle layer, Meta UI, has implementations of repeating user interface constructs (i.e., dynamic lists, enumerations, and tables), and switching parts of the user interface on and off dynamically.

Finally, the top layer, user interface, contains both high and low level implementations of all form controls in the XForms specification, such as select and textarea. The high level implementation contains the logic part and maps the form control into a specific user interface widget, which is implemented in the low level part. The high-level part is not aware of the details of the control's graphical representation, but rather receives and sends events from and to the object representing the control.

The lower-level part of the user interface is implemented by creating a factory interface ComponentFactory, along with interfaces for different types of widgets. Then, for each Java component toolkit, such as Swing, an implementation of the ComponentFactory along with the implementations of each of the widgets is provided. This requires quite many classes, but since these classes are just wrappers for the

[5] http://www.x-smiles.org/

ISBN: 0-7659-2217-3 © 2004 IEEE

real widget implementations, there is not that much programming involved.

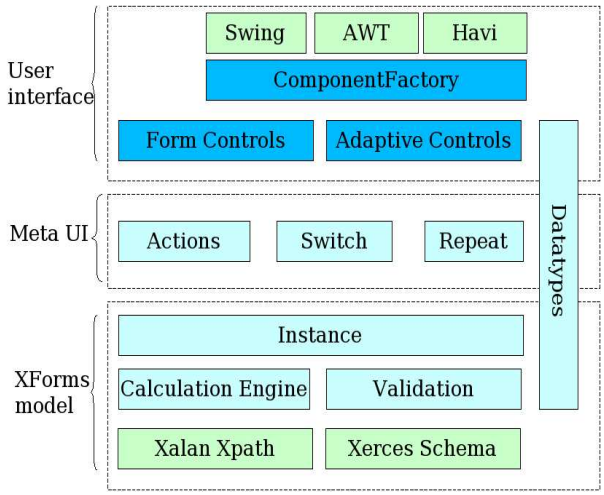

Fig. 3. XForms Implementation Architecture.

Fig. 4. depicts the class relationships between the factory and widget interfaces and their implementing classes. ComponentFactory is an interface that HaviComponentFactory and SwingComponentFactory implement. There is also other interfaces for the widgets themselves, such as XInput and XSelectOne. All of these interfaces are implemented by the corresponding component toolkit wrapper. Each of the implementing classes, such as HaviInput use internally some component from the underlying component toolkit to provide the functionality. For events, AWT event classes and listeners are used where possible, thus removing the need for defining extra event classes.

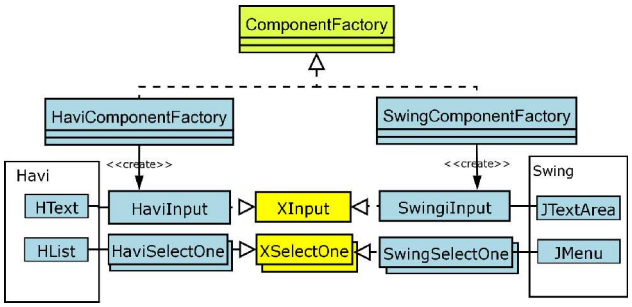

Fig. 4. ComponentFactory is Used to Hide the Concrete Details of the Created Widgets.

As a conclusion, we have a prototype implementation of XForms controls for several back ends: AWT, Swing, and HAVi. Table 2 shows the mapping between the controls and the actual used widgets.

Table 2 . Mapping of the user interface elements.

XForms Control	AWT Widget	Swing Widget	HAVi Widget
select1 & select	List	JList	HListGroup
trigger	Button	JButton	HTextButton
submit	Button	JButton	HTextButton
label	TextField	JTextField	HStaticText
textarea	TextArea	JTextArea	HText
input + xsd:string	TextField	JTextField	HSingleLine Entry
input + xsd:date	TextField	JCalendar[6]	HSingleLine Entry
input + xsd:boolean	CheckBox	JCheckBox	Htoggle Button
secret	SecretField	JPassword Field	Not implemented yet

3.3 Platform Implementation

In previous work, the implementation of a cross-platform SMIL player [11] and its possible uses in a real digital television broadcast environment, called Otadigi[7] [15], have been presented. Even though multimedia presentations can be easily developed as a SMIL document, the main restriction was the lack of real user interaction (i.e., interaction was only provided by the internal links). For that reason, extending support for other XML languages, such as XForms, was essential in order to develop more complex applications.

Ubik is a graphics system framework prototype for digital television receivers [16]. Specifically, our research consists of the study of different user interface tools alternatives for digital television applications. The basic assumption being that there is not a best option, but the selected tool depends on the application at hand (e.g., a 3D graphics game will not be developed using the same tool as a web site). Ubik graphics system, as depicted in Fig. 5, implements the digital television configuration defined in Fig. 1. It provides graphics support for Multimedia Home Platform[8] (MHP) applications, the middleware API defined in Europe as standard for digital television receivers. It includes a basic AWT package and a Java graphical user interface framework following HAVi specifications, called Future TV (FTV) [17]. In addition, the XML User Agent has been integrated in

[6] http://www.toedter.com/en/jcalendar
[7] http://www.otadigi.tv
[8] http://www.mhp.org

ISBN: 0-7659-2217-3 © 2004 IEEE

Ubik, so it now supports as well hybrid SMIL and XForms documents.

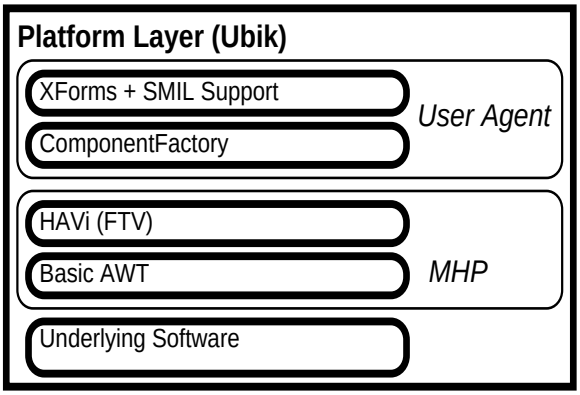

Fig. 5. Ubik Graphics Architecture.

FTV includes, apart from a digital television oriented widgets, all the graphical user interface functionality required for developing interactive applications (e.g., graphical context and events). Fig. 6 shows an UML diagram of some of the widgets included in the package. The visible widgets, which extend AWT's Component Class, are actually the base of all widgets. Navigable components are those, which can be navigated using the remote control. Whereas actionable components are those, which change the state of the application (e.g., buttons). These widgets have been successfully tested in real MHP applications transmitted in Otadigi [18].

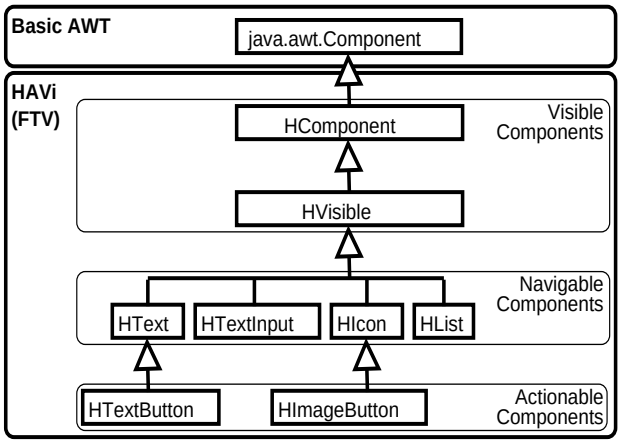

Fig. 6. java.awt.Component and HAVi (FTV) Widgets UML Class Diagram.

4 Case Study

In order to demonstrate the concepts presented in this paper, an interactive Distance Education portal was developed for this paper. The portal contains lectures from the Helsinki University of Technology.

The author of the lectures is either a teacher or an assistant. Note, that the author is not a programmer, so the system should be as easy to author as possible. The user of the portal is a student who wants to reach the lectures ubiquitously using different kind of devices. The portal should contain the lectures in an attractive multimedia form, while still giving the user the possibility to navigate within the lectures in her own pace. Additionally, there should be short "exams" between or within the lectures. Passing the exam is required from the user in order to move to the next part.

Fig. 7 depicts the timeline of a lecture set in the application. First, a welcome screen is splashed. Next, the user is presented with the list of available lectures. The user selects the lecture she wants to attend. The lecture contains video and audio recording from a real lecture. At the same time, the slides of the teacher are shown in a synchronised fashion. The user may jump to different parts of the lecture using an outline that is always present. Some parts may require that the user has successfully responded to a short "exam". The exam has few multiple-choice questions that are quick to answer, but require the user to follow the lecture. Some selections in the outline are activated when the user answers correctly enough to the questions. For example, she has to get 75 % of the answers right.

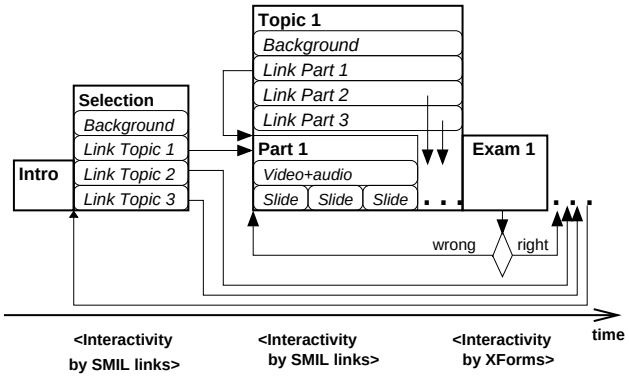

Fig. 7. Timeline of the Distance Education demo.

The requirements of the demo can be mapped to the features of the platform presented in this paper. Timing and multimedia is possible by utilizing SMIL as the host language for the lectures. Interaction and the calculation of the results of the exam in real time can be achieved with XForms. The lectures are easy to author and do not require any programming, since the documents are

ISBN: 0-7659-2217-3 © 2004 IEEE

completely declarative and no scripting is used. Our proposed platform provides device independence since it supports many graphical environments.

The application is developed completely using SMIL and XForms, and it runs on the basic Java Component Toolkits: Swing (PC desktops), AWT (high-end handhelds), and HAVi (digital television receivers). Fig. 8 contains the screenshots of the application. Screenshots a) and b) show a small timed introduction and a lecture selection. The lecture is shown in screenshot c), containing audio and video and a slideset that is synchronised to it. The mid-term exam is depicted in screenshot d).

Applications, such as the one presented here, require advanced timing and sychronization of media, which is not feasible using HTML. Also, the conditional timeline changes based on the selections and answers from the user can be done declaratively, while they usually require programming or scripting. Using our platform, it is possible, also for non-programmers, to create advanced multimedia content and applications. Table 3 shows the approximate number of lines of the application presented on this paper, illustrating the easiness and quickness of developing interactive multimedia applications in our environment.

Table 3 . Number of Lines per Module of the Distance Education Application.

Module	*Number of Lines*
Whole Application	256
Intro	38
Video and Slides	104
Exam	114

5 Conclusions

In this paper, we have presented a device independent XML based user interface model. Interactive services can be developed using hybrid SMIL and XForms documents, where the presentation is provided by the host SMIL language and the interaction is handled by the XForms controls. Furthermore, the graphical support for these XML languages is provided by a general graphical user interface framework, called ComponentFactory API, which includes specific backs-ends such as Swing and HAVi. Thus, the problem of platform independence is solved avoiding the creation of different software for different devices. Moreover, the ComponentFactory can be easily extended in order to support other Java graphics libraries. Since the applications are completely declarative, the presentation layout problem in different devices can be solved by the use of CSS Media Queries and XSL Transformations, or even automated content adaptation tools.

Three main contributions can be found in this paper. The first one is the definition of a valid XML alternative (i.e., SMIL + XForms documents) to develop interactive multimedia applications. The second one is an implementation supporting these languages for a variety of devices. The final one is the inclusion of hybrid XML languages into the Ubik environment, an ongoing digital television prototype platform, which studies the use of different user interface development tools.

The future work includes the generation of a single layout model that would suit SMIL as well as CSS based languages, such as XHTML, which will increase the value of the prototype. First, because the mismatch between the layout models in the XForms and SMIL implementations that we used, there were some advanced XForms functionality that could not be implemented yet. These removed XForms language modules were listed in Section 3. Second, because the integration of XML languages will become easier.

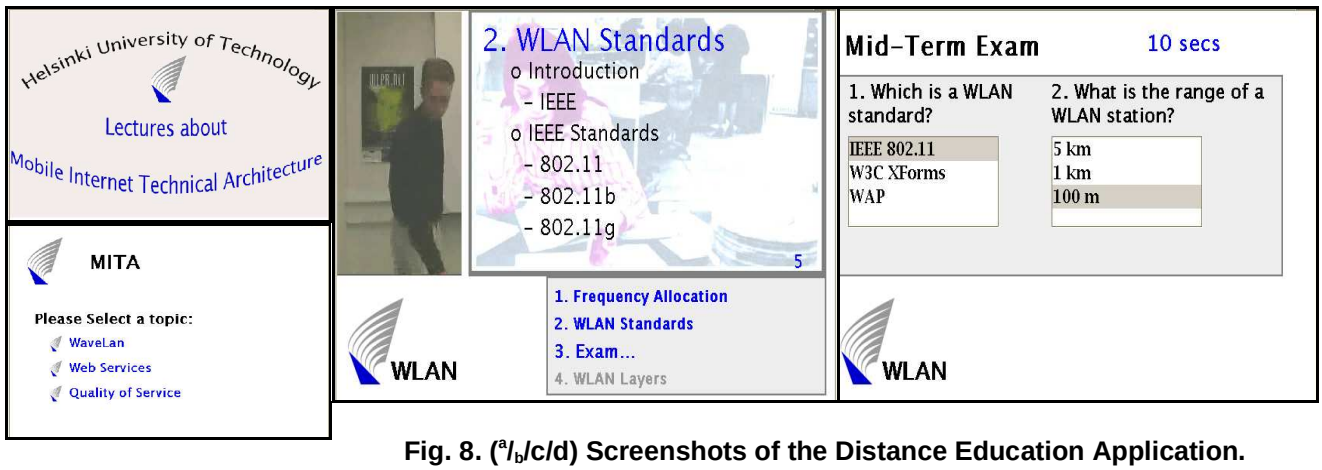

Fig. 8. (^{a}/$_{b}$/c/d) Screenshots of the Distance Education Application.

ISBN: 0-7659-2217-3 © 2004 IEEE

Acknowledgements

The authors Mikko Honkala and Pablo Cesar would like to thank to Nokia Oyj Foundation for providing support during the research. The research was funded by the Go-MM, BROCOM, and XML Devices projects to whose partners the authors would like to express their gratitude. Finally, the authors would like to thank the original developer of the SMIL player, Kari Pihkala.

References

[1] D. R. Olsen, Interacting in chaos, *Interactions*, Volume: 6, Issue: 6, September-October 1999, pp. 42-54.

[2] B. A. Myers, S. E. Hudson, and R. Paush, Past Present, and Future of User Interface Software Tools, *ACM Transactions on Computer-Human Interaction*. Volume: 7, Issue: 1, 2000, pp. 3-28.

[3] K. Pihkala, *Extensions to the SMIL Multimedia Language*, PhD Thesis, Helsinki University of Technology, Finland, 2003.

[4] S. Boll, *ZYX, Towards Flexible Multimedia Document Models for Reuse and Adaptation*, PhD Thesis, University of Vienna, Austria, 2001.

[5] K. Sakamura, A Java-enabled revolution, *IEEE Micro*, Volume: 21, Issue: 4, July-Aug. 2001, pp. 2-3.

[6] W. ten Kate, P. Deunhouwer, and R. Clout, Timesheets – Integrating Timing in XML, in *Proc. WWW2000 Workshop: Multimedia on the Web*, Amsterdam, Netherlands, May 15, 2000.

[7] J. V. Ossenbruggen, L. Hardman, J. Geurts, and L. Rutledge, Towards a Multimedia Formatting Vocabulary, in *Proc. WWW2003*, Budapest, Hungary, May 20-24, 2003, pp. 384-393.

[8] J. Smith, R. Mohan, and C.-S. Li, Scalable Multimedia Delivery for Pervasive Computing, in *Proc. 7th ACM International Conference on Multimedia (Part 1)*, Orlando, Florida, USA, Oct. 30-Nov. 4, 1999, pp. 131-140.

[9] K. Pihkala, M. Honkala, and P. Vuorimaa, Multimedia web forms, in *Proc. Synchronised Multimedia Integration Language European Conference*, *SMIL Europe 2003*, Paris, Feb. 12-14, 2003.

[10] K. Pihkala, M. Honkala, and P. Vuorimaa, A browser framework for hybrid XML documents, in *Proc. 6th IASTED International Conference on Internet and Multimedia Systems and Applications*, *IMSA 2002*, Kauai, Hawaii, USA, August 12-14, 2002, pp. 164-169.

[11] K. Pihkala, P. Cesar, and P. Vuorimaa, Cross-Platform SMIL player, in *Proc. 1st IASTED International Conference On Communications, Internet, and Information Technology*, *CIIT 2002*, St. Thomas, US Virgin Islands, November 18-21, 2002, pp. 48-53.

[12] M. Dubinko et al., XForms 1.0, *W3C Recommendation*, 14 October 2003.

[13] P. Vuorimaa, T. Ropponen, N. von Knorring, and M. Honkala, A Java based XML browser for consumer devices, in *Proc. 17th ACM Symposium on Applied Computing*, Madrid, Spain, March 10-13, 2002.

[14] M. Honkala and P. Vuorimaa, XForms in X-Smiles, *Journal of World Wide Web, Internet and Web Information Systems*, Kluwer, 2001, Vol. 4, No. 3, pp. 151-166.

[15] J.L. Lamadon, P. Cesar, C. Herrero, and P. Vuorimaa, Usages of a SMIL player in a digital television broadcast system, in *Proc. 7th IASTED International Conference on Internet and Multimedia Systems and Applications*, *IMSA2003*, Honolulu, Hawaii, August 13-15, 2003, pp. 579-584.

[16] P. Cesar, J. Vierinen, and P. Vuorimaa, Open graphical framework for interactive TV, in *Proc. 5th International Symposium on Multimedia Software Engineering*, *MSE2003*, Taichung, Taiwan, December 10-12, 2003, pp. 21-28.

[17] P. Cesar and P. Vuorimaa, A graphical user interface framework for digital television, in *Proc. 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, WSCG'2002, 2002, posters, pp. 1-4.

[18] C. Herrero, P. Cesar, and P. Vuorimaa, Delivering MHP applications into a real DVB-T network, Otadigi, in *Proc. 6th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, *TELSIKS2003*, Nis, Serbia and Montenegro, October 1-4, 2003, pp. 231-234.

ISBN: 0-7659-2217-3