Multi-channel Publication of Interactive Media Content for Web Information Systems

Nico Oorts[1], Filip Hendrickx[2], Tom Beckers[3], and Rik Van De Walle[3]

[1] VRT, A. Reyerslaan 52, Brussels, Belgium
nico.oorts@vrt.be
[2] IMEC, Kapeldreef 75, Leuven, Belgium
filip.hendrickx@imec.be
[3] Ghent University-IBBT, Sint-Pietersnieuwstraat 41, Ghent, Belgium
tom.beckers@xmt.be, rik.vandewalle@ugent.be

Abstract. The increasing number of devices for multimedia consumption poses a significant challenge on the device independence of Web Information Systems. Moreover, media content itself becomes increasingly complex, requiring not only an adaptive layout model, but also support for interactivity and synchronisation of resources.
In this paper, we use the XiMPF document model integrated with descriptive languages for layout, synchronisation and interaction. This integration preserves the uniformity which XiMPF handles singular resources and composed multimedia items with. Moreover, it improves the reusability and automatic adaptation of multimedia content. We focus on the introduction of interactivity in this model and present a publication engine which processes XiMPF items and their descriptions, and generates different versions of the content for different platforms.

1 Introduction

The emergence of powerful networked devices such as mobile phones, personal digital assistants and set-top boxes brings about a dramatic diversification of the means of multimedia consumption. Web Information Systems need to be able to select and adapt content in order to target diverse types of client devices.

Our approach allows fine grained adaptation of multimedia content to the capabilities of target devices and user preferences while maintaining maximum publisher control options. It aims to facilitate the reuse of multimedia resources and presentational information across heterogeneous consumption platforms. This is accomplished by a rigorous separation of concerns (content, structure, positioning, styling, synchronisation, behavior/interactivity) in the hierarchically structured XiMPF document model.

2 The XiMPF Document Model

In order to give the publisher maximal control of the output specification and to maximize reuse possibilities, we have continued development of the XiMPF

D. Lowe and M. Gaedke (Eds.): ICWE 2005, LNCS 3579, pp. 394–399, 2005.

document model (see [3, 6, 7] for a detailed introduction and examples). Fashioned after the MPEG-21 Digital Item Declaration (DID) model [5], it defines a semantically richer set of elements to structure and annotate the presentation content.

The XiMPF document model defines a document as a tree aggregating composing items. The nodes of the tree consist of (sets of) description fragments and content resources. The document model makes abstraction of item content: atomic multimedia resources are allowed as alternatives for composite presentation fragments, for example a picture and caption combination can be substituted for a video.

For the aggregation of textual and multimedia content, descriptions of structure, layout, synchronisation and behavior are used. We currently use available W3C technologies (XHTML, CSS, SMIL) supplemented with custom developed description languages where needed (e.g. interactivity). By not limiting the descriptive languages to be used, a flexible and future-proof framework guarantees the ability to cope with new target platforms.

Composite presentation fragments and atomic multimedia resources are treated in a similar fashion. Both can be automatically transformed or adapted to the target platform by the publication engine. The publisher can:

- leave the selection between alternatives to the publication engine - based on the client capabilities and preferences, or
- point to the version he deems fit for a certain presentation.

3 Use Case Description

To demonstrate the validity of the XiMPF model and the feasibility of a web information system in a multimedia, multichannel context, we operated within a specific use case scenario: the development of a new publication infrastructure for the present VRTNieuws.net website [2]. At the moment, the leading Belgian public broadcast company VRT (Vlaamse Radio en Televisie) produces news content for television, radio and internet. The internet publication engine produces only two fundamentally different versions: a full multimedia and a text-only version, both targeted towards a PC browser.

The new engine separates the editing of content from the adaptation and presentation for different platforms. Several versions of the news content (including text content as well as multimedia resources) are automatically derived from the same source material but the publisher retains control over the layout of the document presented to the customer.

In this use case, we generate output adapted to the following platforms:

- a television set with a set-top box and a large, high quality screen;
- a PC system with adequate system resources and a medium sized screen;
- a personal hand-held device (PDA) with wireless network connection.

4 Introducing Interactivity

Adding interactive behavior to media publications usually requires some form of programming. This often results in implementation code being spread across the entire publication. Such entanglement complicates adaptation of the document to support different platforms. It also interferes with reuse of the non interactive parts of the media publication, as these parts cannot be easily separated from the interaction code. In this section, we elaborate on a method to integrate interactive components in the XiMPF model in a way that leaves the separation of concerns intact.

An example is a user controlled slideshow developed for the news site use case. Visually, the slideshow consists of an image and a back and forward button. When the user clicks one of the buttons, the image changes. In a web environment, this requires adding some code to the buttons via the onclick attribute. This code is usually a call to a function in a separate script element or external file. The external code contains a reference to the image element that must change when the user clicks a button. During an initialisation phase, the images are preloaded and stored in an array. The back and forward functions can access this array and show the correct image.

While this is a simple example of interactivity, it already shows that the implementation of the behavior is tightly coupled with the rest of the document. There are links between the buttons, the images, the initialization phase and the rest of the code. Even though a Javascript and a VBScript implementation are conceptually identical, replacing one with the other will not be trivial, as several pieces of code throughout the whole document must be changed. This complicates reuse of the rest of the application (structure, layout and synchronisation).

Publishing this interactive presentation on a fundamentally different platform, like a television with Java based set-top box, will be cumbersome, as manual editing of the whole presentation is inevitable.

Our approach is based on a strict separation of the – possibly platform specific – implementation of the interactive behavior from the rest of the document. This ensures reusability of the rest of the media application. Three pieces of information are required to enable this separation (see figure 1).

1. An *API* defines the generic interface of the interactive behavior. For the slideshow, it defines the parameters for the initialisation phase (a list of images and a reference to the container for the images) and the supported features (back, forward) together with their parameters if present.
2. An *interaction description* in the XiMPF document links the parameters and features from the API with specific XiMPF items. These links are independent from the actual implementation of the interactive behavior.
3. An *implementation* of the API for a certain platform. When targeting a web browser, this implementation will for example define an `onclick` attribute for the back and forward buttons and an `onload` attribute for the `body` element. The attribute values will contain a Javascript function call implementing the desired behavior. The referred functions are also defined in the implementation.

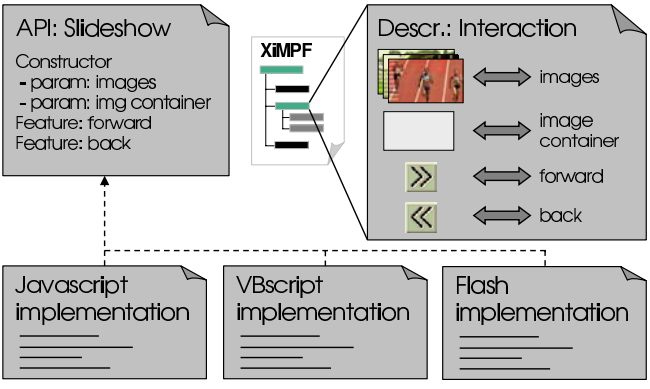

Fig. 1. Interactivity is added to media content via an API defining the behavior, an interaction description in the XiMPF document linking parameters and features with specific XiMPF items and an implementation

To illustrate this method, we will describe the necessary steps to add a feature to an interactive document. This feature will allow a user to push a button to move a slideshow one image forward. To add the 'forward' feature to the slideshow example, one needs to list the feature with its parameters (name and type) (in this case none) in the API. In the XiMPF interaction description, this feature is bound to a GUI item (in this case an image defined in the structure definition that will serve as a button through the inclusion of an onClick method). If needed, items or values are added as parameters to this feature binding in the XiMPF interaction description. The implementation file defines the construction of the code fragments that need to be added to the output document. This construction includes XSLT processing to replace parameters with their actual values. Parameters are replaced with the facet (selected in the implementation, e.g. item ID or item source path) of the parameter item (as declared in the interaction description). The publication engine constructs the code fragments and weaves them into the output publication.

This method of describing the interactive behavior of the slideshow application enables reuse of the XiMPF document, including the interaction description, for different platforms. For example, if we want to use Java or Flash instead of Javascript when publishing the slideshow, we only need to add the corresponding Java or Flash implementation alongside the Javascript implementation. As long as the new implementation adheres to the slideshow API, a publication of the interactive application in the correct format is possible without changes to the XiMPF document.

While the slideshow is a simple example, describing more complex interactive behavior is equally straightforward. Once the API is defined, it can be used in XiMPF documents without any dependency on the actual implementation (complexity).

5 Publication Framework

Our publication infrastructure, based on the Apache Cocoon framework [1], performs the generation of the news site described in our use case and the adaptation of atomic resources. The current version of our architecture (see figure 2) consists of an XML processing pipeline including a XiMPF Transformer, responsible for resolving and adapting XiMPF documents to a specific presentation version, a core engine, which acts as an intelligent broker, a resource and metadata (device characteristics, resource metadata and so forth) database and an adaptation service registry. The adaptation services registered by the service registry are used to process atomic multimedia resources like audio and video.

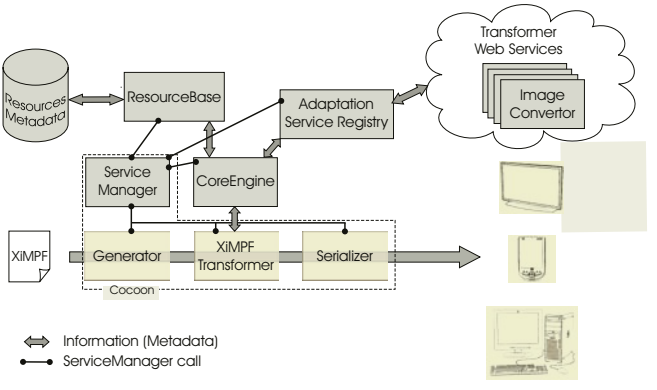

Fig. 2. The publication framework based on Cocoon

6 Related Work

A lot of the standard multimedia document models like SMIL and RIML [8], lack appropriate modeling primitives to enable reuse of multimedia content in different presentations and contexts. For example, SMIL's layout mechanism only allows fixed positioning of media items with regard to each other, and thus forces the author to create separate layouts for devices with highly differing screen sizes. XHTML and CSS take the approach of separating structure and layout but still lack adequate support for true single authoring. Different style sheets are needed to accommodate various platforms. To enable content creation and reuse for complex content applications, a more rigorous separation of presentational aspects is required, together with a high level document model like MPEG-21 DID. However, the practicability of MPEG-21 DID is vague and the model needs extensions for description types and structure which are introduced by the XiMPF model. Model-driven methodologies, like Hera [4] distinguish three design steps: conceptual, navigational and presentation design. Hera's navigation model offers flexible support for form-based interactivity in data-intensive applications. This is complementary to the XiMPF interactivity model, which is more geared towards dynamic content. Note also that Hera does not support synchronisation.

7 Conclusions

Device independent content management is an important challenge for Web Information Systems. This is especially true for dynamic multimedia content. In this context, we have indicated the strengths of the XiMPF document model, namely reuse of both media resources and presentational information at a fine level of granularity, and support for automatic adaptation of both singular and composed media content. The introduction of interactivity in a reusable and implementation independent manner demonstrates the extensibility of XiMPF. We built an extensible publication framework able to adapt media resources and descriptions, and generate specific output documents for different client terminals. We applied it to the news site of the leading Belgian broadcaster VRT, generating adapted news publications including interactive JavaScript, VBScript and Flash components for 3 different platforms.

Acknowledgements

This work is the result of a joint collaboration between VRT, IMEC, Vrije Universiteit Brussel and Universiteit Gent, funded by the Flemish government. The authors want to thank their colleague Peter Soetens for his valuable input regarding the publication framework.

References

1. The apache cocoon project. Available at http://cocoon.apache.org.
2. Vrtnieuws.net. Available at http://www.vrtnieuws.net.
3. T. Beckers, N. Oorts, F. Hendrickx, and R. Van de Walle. Multichannel publication of interactive media documents in a news environment. To be published in Proceedings of the Twelfth International World Wide Web Conference, 2005.
4. G.-J. Houben, F. Frasnicar, P. Barna, and R. Vdovjak. Modeling user input and hypermedia dynamics in Hera. In *Proceedings of the 4th International Conference on Web Engineering*, pages 60–73, Munich, Germany, July 2004. Springer Verlag.
5. ISO/IEC. *MPEG-21 – Part 2: Digital Item Declaration – ISO/IEC FDIS 21000-2.*
6. S. Van Assche, F. Hendrickx, and L. Nachtergaele. Multichannel publication using MPEG-21 DIDL and extensions. In *Proceedings of the Twelfth International World Wide Web Conference*. W3C, 2003. Poster.
7. S. Van Assche, F. Hendrickx, N. Oorts, and L. Nachtergaele. Multi-channel publishing of interactive multimedia presentations. *Computers & Graphics*, 2004(5):193–206.
8. T. Ziegert, M. Lauff, and L. Heuser. Device independent web applications - the author once - display everywhere approach. In *Proceedings of the 4th International Conference on Web Engineering*, pages 244–255, Munich, Germany, July 2004. Springer Verlag.