IPML: Extending SMIL for Distributed Multimedia Presentations

Jun Hu and Loe Feijs

Department of Industrial Design
Eindhoven University of Technology
5600MB Eindhoven, The Netherlands

Abstract. This paper addresses issues of distributing multimedia presentations in an ambient intelligent environment, exams the existing technologies and proposes IPML, a markup language that extends SMIL for distributed settings. It uses a powerful metaphor of play, with which the timing and mapping issues in distributed presentations are easily covered in a natural way.

Ambient Intelligence (AmI) is introduced by Philips Research as a new paradigm in how people interact with technology. It envisions digital environments to be sensitive, adaptive, and responsive to the presence of people, and AmI environments will change the way people use multimedia services [1]. The environments, which include many devices, will play interactive multimedia to engage people in a more immersive experience than just watching television shows. People will interact not only with the environment itself, but also with the interactive multimedia through the environment.

For many years, the research and development of multimedia technologies have increasingly focused on models for distributed applications, but the focus was mainly on the distribution of the media sources. Within the context of AmI, not only are the media sources distributed, the presentation of and the interaction with the media will also be distributed across interface devices. This paper focuses on the design of the structure of multimedia content, believing that the user experience of multimedia in a distributed environment can be enriched by structuring both the media content at the production side and the playback system architecture at the user side in a proper way. The structure should enable both the media presentation and the user interaction to be distributed and synchronized over the networked devices in the environment. The presentation and interaction should be adaptive to the profiles and preferences of the users, and the dynamic configurations of the environment. To structure the media content, the following issues need to be addressed:

1. By what means will the authors compose the content for many different environments? The authors have to be able to specify the following with minimized knowledge of the environments: (a) Desired environment configurations; (b) Interactive content specification for this environment.
2. How can the system playback the interactive media with the cooperation of the user(s) in a way that: (a) makes the best use of the physical environment

H. Zha et al. (Eds.): VSMM 2006, LNCS 4270, pp. 60–70, 2006.
© Springer-Verlag Berlin Heidelberg 2006

to match the desired environment on the fly. (b) enables context dependent presentation and interaction. Here the term "context" means the environment configuration, the application context, the user preferences, and other presentation circumstances. (c) synchronizes the media and interaction in the environment according to the script.

This paper first exams existing open standards for synchronized media and a scripting language called StoryML, then proposes IPML addressing aforementioned issues.

1 When Technologies Meet Requirements

SMIL and MPEG-4 are contemporary technologies in the area of synchronized multimedia [2,3]. SMIL focuses on Internet applications and enables simple authoring of interactive audiovisual presentations, whereas MPEG-4 is a superset of technologies building on the proven success in digital television, interactive graphics applications and also interactive multimedia for the Web. Both were the most versatile open standards available at the moment of time.

But both were challenged by the requirement for distributed interactions. It requires that the technology is first of all able to describe the distribution of the interaction and the media objects over multiple devices. The BIFS in MPEG-4 emphasizes the composition of media objects on one rendering device. It doesn't take multiple devices into account, nor does it have a notation for it.

SMIL 2.0 introduces the MultiWindowLayout module, which contains elements and attributes providing for creation and control of multiple top level windows [4]. This is very promising and comes closer to the requirements of distributed content interaction. Although these top level windows are supposed to be on the same rendering device, they can to some extent, be recognized as software interface components which have the same capability.

To enable multimedia presentations over multiple interface devices, StoryML was proposed [5]. It models the interactive media presentation as an interactive *Story* presented in a desired environment (called a *Theater*). The story consists of several *Storylines* and a definition of possible user *Interaction* during the story. User interaction can result in switching between storylines, or changes within a storyline. *Dialogues* make up the interaction. A dialogue is a linear conversation between the system and the user, which in turn consists of *Feed-forward* objects, and the *Feedback* objects depending on the user's Response. The environment may have several *Interactors*. The *interactors* render the media objects. And finally, the story is rendered in a Theater.

One problem of the StoryML is that it uses a mixed set of terms. "Story" and "storylines" are from narratives, "media objects" are from computer science, whereas *itneractors* are from human computer interaction. Scripting an interactive story requires various types of background knowledge to some extent. It is questionable whether StoryML has succeeded in both keeping the scripting language at a high level and let the script authors only focus on the interactive

content. “Movies did not flourish until the engineers lost control to artists – or more precisely, to the communications craftsmen.” [6]

StoryML uses storytelling as a metaphor for weaving the interactive media objects together to present the content as an “interactive story”. This metaphor made it difficult to apply StoryML to other applications when there are no explicit storylines or narratives. Moreover, StoryML can only deal with linear structure and use only a storyline switching mechanism for interaction.

Instead of StoryML, it is necessary to design a script language that has a more generic metaphor, that supports both linear and nonlinear structures and that can deal with complex synchronization and interaction scenarios.

2 Play

Instead of storytelling, Interactive Play Markup Language (IPML) uses a more powerful metaphor of *play*. A *play* is a common literary form, refering both to the written works of dramatists and to the complete theatrical *performance* of such. Plays are generally performed in a *theater* by *actors*. To better communicate a unified interpretation of the text in question, productions are usually overseen by a *director*, who often puts his or her own unique interpretation on the production, by providing the actors and other stage people with a *script*. A script is a written set of directions that tell each actor what to say (*lines*) or do (*actions*) and when to say or do it (*timing*). If a play is to be performed by the actors without a director and a script from the director, the results would be unpredictable, if not chaotic.

2.1 Timing in a Play

Timing in a play is very important whether it be when an actor delivers a specific line, or when a certain character enters or exits a scene. It is important for the playwright to take all of these into consideration. The following is an example from *Alice in Wonderland* [7, p.14]:

. . .

ALICE Please! Mind what you’re doing!

DUCHESS (*tossing ALICE the baby*). Here . . . you may nurse it if you like. I’ve got to get ready to play croquet with the Queen in the garden. (*She turns at the door.*) Bring in the soup. The house will be going any minute! (*As the DUCHESS speaks, the house starts moving. The COOK snatches up her pot and dashes into the house.*)

COOK (*to the FROG*). Tidy up, and catch us! (*The FROG leaps about, picking up the vegetables, plate, etc.*)

ALICE (*as the FROG works*). She said “in the garden.” Will you please tell me –

FROG There’s no sort of use asking me. I’m not in the mood to talk about gardens.

ALICE I must ask some one. What sort of people live around here?

. . .

A few roles are involved in this part of the play. Their lines and actions are presented by the playwright in a sequential manner, and these lines and actions

are by default to be played in sequence. However, these sequential lines and actions are often not necessarily to happen immediately one after another. For example, it is not clear in the written play how much of time the duchess should take to perform the action "*tossing Alice the baby*" after Alice says "*Mind what your're doing*" and before the duchess says "*Here ... you may nurse it if you like*". The director must supervise the timing of these lines and actions for the actors to ensure the performance is right in rhythm and pace. Furthermore, things may happen in parallel – For example, the house starts moving as the duchess speaks, and Alice talks as the frog works. Parallel behaviors are often described without precise timing for performing. It is up to the directors to decide the exact timing based on their interpretation of the play. For example, the director may interpret "*As the DUCHESS speaks, the house starts moving*" as "*at the moment of the duchess start saying 'The house will be going in any minute', the house starts moving*".

2.2 Mapping: Assigning Roles to Actors

Actors play the roles that are described in the script. One of the important task of the director is to define the cast – assign the roles to actors. This is often done by studying the type of a role and the type of an actor, and finding a good match between them. This is also exactly the problem for distributed presentations: determining which device or component to present certain type of media objects. It can be very hard for a computer to carry out this task, unless these types are indicated in some way otherwise.

In some traditional art of play, these types are even formalized so that a play can be easily performed with a different cast. For example, The character roles in Beijing Opera are divided into four main types according to the sex, age, social status, and profession of the character: male roles (Shēng 生, fig. 1(a)); female roles (Dàn 旦, fig. 1(b)); the roles with painted faces (Jìng 净, fig. 1(c)) who are usually warriors, heroes, statesmen, or even demons; and clown (Chǒu 丑, fig. 1(d)), a comic character that can be recognized at first sight for his special make-up (a patch of white paint on his nose). These types are then divided into more delicate subtypes, for example Dàn is divided into the following subtypes: Qīng Yī(青衣) is a woman with a strict moral code; Huā Dàn (花旦) is a vivacious young woman; Wǔ Dàn (武旦) is a woman with martial skills and Lǎo Dàn (老) is an elderly woman. In a script of Beijing Opera, roles are defined according to these types. An actor of Beijing Opera is often only specialized in very few subtypes. Given the types of the roles and the types of the actors, the task of assigning roles to actors becomes an easy matching game.

2.3 Interactive Play

Plays can be interactive in many ways. The actors may decide their form of speech, gestures and movements according to the responses from the audience. Again with the example of Beijing opera, plays in early days, which sometimes

(a) Sheng(male) (b) Dan(female) (c) Jing (face painted) (d) Chou(clown)

Fig. 1. Role types in Beijing opera

can still be been seen today, may be performed in the street (fig. 2) or in a tea house, where the actors and the audience are mixed – the actors and the audience share the stage. The movements of the actors must be adapted to the locations of the audience, and the close distance between the audience and the actors stimulates the interaction. An example of such interaction is that the characters often strike a pose on the stage, and the audience is supposed to cheer with enthusiasm. The time span of such a pose depends on the reactions of the audience. Although this is often not written in the script, such an interactive behavior is by default incorporated in every play of Beijing opera.

Other interactive plays allow the audience to modify the course of actions in the performance of the play, and even allow the audience to participate in the performance as actors. Thus in these plays the audience has an active role. However, this does not mean that the reader of a novel, the member of audience in the theater are passive: they are quite active, but this activity remains internal.

Fig. 2. 19^{th} century drawing of Beijing opera

The written text of the play is much less than the event of the play. It contains only the dialog (the words that the characters actually say), and some stage directions (the actions performed by the characters). The play as written by the playwright is merely a scenario which guides the director and actors. The phenomenon of theater is experienced in real-time. It is alive and ephemeral – unlike reading a play, experiencing a play in action is of the moment – here today, and gone tomorrow.

To clarify the discussions, the word *performance* is used to refer to the artifact the audience and the participants experience during the course of performing a script by preferred *actors*, monitored and instructed by a *director*. The *script* is the underlying content representation perceived by the authors as a composite

unit, defining the temporal aspects of the performance, and containing the *actions* which are depicted by the *content elements* or the references to these elements. Traditional multimedia systems use a different set of terms which are comparable to the terms above; they are in many cases similar, but should not be confused.

Next the basic structure of the SMIL scripting language [8] is presented. Using the structure of SMIL and taking the extra requirements into account we then propose Interactive Play Markup Language.

3 SMIL

Synchronized Multimedia Integration Language (SMIL) is an XML-based language for writing interactive multimedia presentations [8]. It has easy to use timing modules for synchronizing many different media types in a presentation. SMIL 2.0 has a set of markup modules.

Not attempting to list all the elements in these modules, we show an object-oriented view[1] of some basic elements in Fig. 3(a): Par, and Seq from the timing and synchronization module, Layout, RootLayout, TopLayout and Region from the layout module, Area from the linking module, MediaObject from the media object module, Meta from the metainformation modules and Head, Body from the structure module. Details about the corresponding language elements can be found in SMIL 2.0 specification [8].

The Region element provides the basics for screen placement of visual media objects. The specific region element that refers to the whole presentation is the RootLayout. Common attributes, methods and relations for these two elements are placed in the superclass named the Layout.

SMIL 2.0 introduced a MultiWindowLayout module over SMIL 1.0, with which the top level presentation region can also be declared with the TopLayout element in a manner similar to the SMIL 1.0 root-layout window, except that multiple instances of the TopLayout element may occur within a single Layout element.

Each presentation can have Head and Body elements. In the Head element one can describe common data for the presentation as whole, such as Meta data and Layout. All Region elements are connected to the Head.

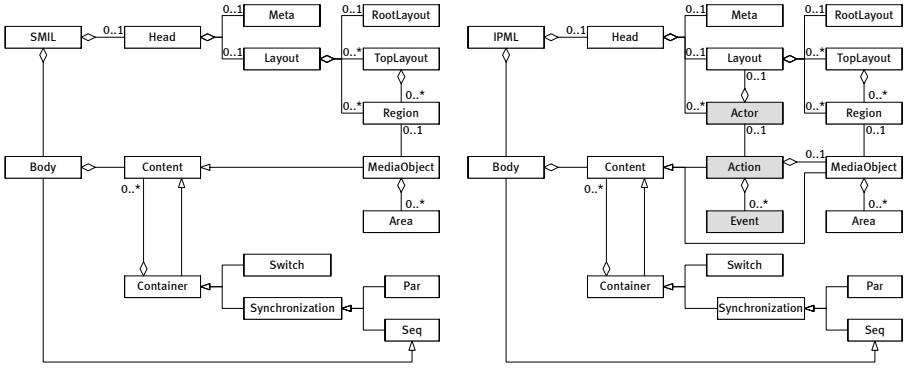
The Mediaobject is the basic building block of a presentation. It can have its own intrinsic duration, for example if it is a video clip or an audio fragment. The media element needs not refer to a complete video file, but may be a part of it. The Content, Container, and Synchronization elements are classes introduced solely for a more detail explanation of the semantics of the Par, Seq, Switch and Mediaobject, and their mutual relations.

Par and Seq are synchronization elements for grouping more than one Content. If the synchronization container is Par, it means that direct subelements can be presented simultaneously. If synchronization container is Seq, it means that direct subelements can be presented only in sequence, one at a time. The Body element is also a Seq container.

[1] This UML model only provides an overview of the selected elements and it is not intended to replace, nor to be equivalent to, the DTD specification from W3C.

The connection between Content and Container viewed as an aggregation has a different meaning for the Synchronization element and for the Switch element. If the Container element is Switch, that means that only one subelement from a set of alternative elements should be chosen at the presentation time depending on the settings of the player.

With the Area element, a spatial portion of a visual object can be selected to trigger the appearance of the link's destination. The Area element also provides for linking from non-spatial portions of the media object's display. It allows breaking up an object into temporal subparts, using attributes begin and end.



(a) SMIL in UML (b) IPML in UML

Fig. 3. SMIL and IPML

4 IPML

SMIL seems to have the ingredients for mapping and timing:

- Its timing and synchronization module provides versatile means to describe the time dependencies, which can be directly used in the IPML design without any change.
- The SMIL linking module enables non-linear structures by linking to another part in the same script or to another script. Although the Area element can only be attached to visual objects, this can be easily solved by lifting this concept up to a level that covers all the elements that need to have a linking mechanism.
- The SMIL layout module seems to be very close to the need of distribution and mapping. The concept of separating mapping and timing issues into two different parts, i.e. Head and Body, makes SMIL very flexible for different layouts – if a presentation need to be presented to a different layout setting, only the layout part need to be adapted and the timing relations remain intact, no matter whether this change happens before the presentation in authoring time, or during the presentation in run time.

It is not yet forgotten in section 1, SMIL was considered not directly applicable out of the shelf for the distributed and interactive storytelling: it does not support a notion of multiple devices. However later we also found that we went one step too far – the StoryML does incorporate the concept of multiple actors, but its linear timing model and narrative structure limited its applicability.

What needs to be done is to pick up SMIL again as the basis for the design, extending it with the metaphor of theater play, bringing in the lessons we learnt from StoryML. Fig. 3(b) shows the final IPML extension (marked gray) to SMIL. The Document Type Definition (DTD) of IPML can be found in [9].

Note that in fig. 3(b), if all gray extensions are removed, the remaining structure is exactly the same as the SMIL structure as illustrated in fig. 3(b). This in an intentional design decision: IPML is designed as an extension of SMIL without overriding any original SMIL components and features, so that the compatibility is maximized. Any SMIL script should be able to be presented by a IPML player without any change. An IPML script can also be presented by a SMIL player, although the extended elements will be silently ignored. The compatibility is important, because it can reduce the cost of designing and implementing a new IPML player – the industry may pick up the IPML design and build an IPML player on top of a existing SMIL player so that most of the technologies and implementations in the SMIL player can be reused.

4.1 Actor

The Head part of an IPML script may contain multiple Actor elements which describe the preferred cast of actors. Each Actor has a type attribute which defines the requirements of what this actor should be able to perform. The type attribute has a value of URI, which points to the definition of the actor type. Such a definition can be specified using for example RDF [10] and its extension OWL. RDF is a language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources. However, by generalizing the concept of a "Web resource", RDF can also be used to represent information about things that can be identified on the Web, even when they cannot be directly retrieved on the Web. OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. The "thing" we need to describe is the type of the actor.

During the performance time, the real actors present to the theater to form a real cast. Each actor then needs to report to the director about what he can perform, i.e. his actor "type". The "type" of a real actor is defined by the actor manufacturers (well, if an actor can be manufactured). The real actor's type can again be described using an RDF or OWL specification. The director then needs to find out which real actor fits the preferred type best. The mapping game becomes a task of reasoning about these two RDF or OWL described "types". First of all the user's preferences should be considered, even if the user prefers a "naughty boy" to perform a "gentleman". Otherwise, a reasoning

process should be conducted by the director, to see whether there is exactly an actor has a type that "equals to" the "gentleman", or to find an "English man" that indeed always "is a" "gentleman", or at least to find a "polite man" that "can be" a "gentleman" and that matches "better than" a "naughty boy", etc. This reasoning process can be supported by a variety of Semantic Web [11] tools, such as Closed World Machine (CWM) [12] and Jena[13] just for example.

4.2 Action

The Action element is similar to the MediaObject element in SMIL. However, Action can be applied to any type of content element which is not explicitly defined using different media objects such as Img, Video and Animation in SMIL. The Action element has an attribute src giving the URI of the content element and its type either implicitly defined by the file name extension in the URI if there is one, or explicitly defined in another attribute type. The type attribute defines the type of a content element as the type attribute of Actor defines the actor type, using a URI referring to a definition.

Action may have an attribute actor to specify the preferred actor to perform it. If it is not specified, the type of the content element may also influence the actor mapping process: the director needs to decide which actor is the best candidate to perform this "type" of action. Again, the user preference should be taken into account first, otherwise a reasoning process should be conducted to find the "gentleman" who can nicely "open the door for the ladies".

In addition, the Action element may have an observe attribute which specifies the interested events. This attribute is designed for an actor to observe the events that are of interest during the the course of performing a specific action. For example, when an actor is performing an action to present a 3D object, it may be interested in the controlling events for rotating the object. This actor can then "observe" these events and react on it. Note that these observed events have no influence on the timing behavior: it will neither start nor stop presenting this 3D object, unless they are included in timing attributes, i.e, begin and end. Events that are not listed will not be passed by the director to the actor during this action, therefore the event propagation overhead can be reduced.

However, some actors may be interested in the events that are not related to certain actions. To accommodate this and not to change the original SMIL structure, we require these Actors to perform an action of the type null, specified using a special URI scheme "null:", which allows events to be "observed" during an action of "doing nothing".

4.3 Event

The third extension of IPML to SMIL is event based linking using the Event element. Event elements in an Action element are similar to Area elements in a visual MediaObject element in SMIL, with the exceptions that it does not require the parent Action element to have a visual content to present, and that the events are not limited to the activation events (clicking on an image, for example) on

visual objects. An Event has an attribute enable to include all interested events during an action, including all possible timing events and user interaction events. Once one of the specified event happens, the linking target, specified using the attribute href is triggered. Similar to the Area element, the Event element may also have begin, end and dur attributes to activate the Event only during a specified interval. Event based linking makes IPML very flexible and powerful in constructing non-linear narratives, especially for the situations where the user interaction decides the narrative directions during the performance.

5 Conclude with Alice in Wonderland

To show what a IPML would look like in practice, we again use the example from *Alice in Wonderland* in section 2.1. Since we can't embed multimedia content elements in this printed paper and we only have printed lines and action instructions, we introduce two exotic URI schemes: "say:" for the lines and "do:" for the action instructions, just for the fun of it:

```
<ipml>
<head>
    <actor id="ALICE"    type="http://alice@wonderland.eu/lovelygirl" />
    <actor id="DUCHESS"  type="http://alice@wonderland.eu/seriouswoman" />
    <actor id="COOK"     type="http://alice@wonderland.eu/cook" />
    <actor id="FROG"     type="http://alice@wonderland.eu/frog" />
    <actor id="HOUSE"    type="http://alice@wonderland.eu/woodenhouse" />
</head>
<body>
    <action actor="ALICE" src="say:Please! Mind what you're doing!" />
    <par>
        <action actor="DUCHESS" src="do:tossing Alice the baby"
                id="DuchessTossingBaby"/>
        <action actor="DUCHESS"
                src="say:Here...you may nurse it if you like, I've got to get
                     ready to play croquet with the Queen in the garden." />
        <action actor="ALICE" src="do:receiving the baby"
                begin="DuchessTossingBaby.babytossed"/>
    </par>
    <action actor="DUCHESS" src="do:turns at the door" />
    <action actor="DUCHESS" src="say:Bring in the soup." />
    <par>
        <action actor="HOUSE" src="do:moving" />
        <seq>
            <par>
                <action actor="DUCHESS" src="say:The house will be going any minute!" />
                <action actor="COOK"    src="do:snatches up her pot and dashes into the
                                             house" />
            </par>
            <action actor="COOK" src="do:turns to the FROG" />
            <action actor="COOK" src="say:Tidy up, and catch us!"
            <par>
                <action actor="FROG"  src="do:leaps about" />
                <action actor="FROG"  src="do:picking up the vegetables, plates, etc." />
                <action actor="ALICE" src="say:She said 'in the garden', will you please
                                           tell me—" />
            </par>
            <action actor="FROG"  src="say:There's no sort of reason asking me, I'm not
                                       in the mood to talk about gardens." />
            <action actor="ALICE" src="say:I must ask some one. What sort of people live
                                       around here?" />
        </seq>
    </par>
</body>
</ipml>
```

References

1. Aarts, E.: Ambient intelligence: a multimedia perspective. IEEE Multimedia **11**(1) (2004) 12–19
2. Battista, S., Casalino, F., Lande, C.: MPEG-4: A multimedia standard for the third millennium, part 1. IEEE MultiMedia **6**(4) (1999) 74–83
3. Battista, S., Casalino, F., Lande, C.: MPEG-4: A multimedia standard for the third millennium, part 2. IEEE MultiMedia **7**(1) (2000) 76–84
4. Rutledge, L.: SMIL 2.0: XML for web multimedia. IEEE Internet Computing **5**(5) (2001) 78–84
5. Hu, J.: StoryML: Enabling distributed interfaces for interactive media. In: The Twelfth International World Wide Web Conference, Budapest, Hungary, WWW (2003)
6. Heckel, P. In: The Elements of Friendly Software Design. Sybex (1991) 4
7. Carroll, L., Chorpenning, C.B.: Alice in Wonderland. Dramatic Publishing Co., Woodstock (1958)
8. Ayars, J., Bulterman, D., Cohen, A., Day, K., Hodge, E., Hoschka, P., Hyche, E., Jourdan, M., Kim, M., Kubota, K., Lanphier, R., Layaïda, N., Michel, T., Newman, D., van Ossenbruggen, J., Rutledge, L., Saccocio, B., Schmitz, P., ten Kate, W., Michel, T.: Synchronized multimedia integration language (SMIL 2.0) - [second edition]. W3C recommendation (2005)
9. Hu, J.: Design of a Distributed Architecture for Enriching Media Experience in Home Theaters. Technische Universiteit Eindhoven (2006)
10. McBride, B.: Rdf primer. W3C recommendation (2004)
11. Berners-Lee, T., Fischetti, M.: Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. Harper San Francisco (1999) Foreword By-Michael L. Dertouzos.
12. Berners-Lee, T., Hawke, S., Connolly, D.: Semantic web tutorial using n3. Turorial (2004)
13. McBride, B.: Jena: Implementing the rdf model and syntax specification. In: Semantic Web Workshop, WWW2001. (2001)