Multimedia Authoring for CoPs

Romain Deltour, Agnès Guerraz, and Cécile Roisin

INRIA Rhône-Alpes
655 avenue de l'Europe
38334 Saint Ismier, France
{Romain.Deltour,Agnes.Guerraz,Cecile.Roisin}@inria.fr

Abstract. One way of providing technological support for CoPs is to help participants to produce, structure and share information. As this information becomes more and more multimedia in nature, the challenge is to build multimedia authoring and publishing tools that meets CoPs requirements. In this paper we analyze these requirements and propose a multimedia authoring model and a generic platform on which specific CoPs-oriented authoring tools can be realized. The main idea is to provide template-based authoring tools while keeping rich composition capabilities and smooth adaptability. It is based on a component-oriented approach integrating homogeneously logical, time and spatial structures. Templates are defined as constraints on these structures.

1 Introduction

In order to support the activities of Communities of Practice, the Palette project [6] will provide tools for document production and for document reuse in heterogeneous applications. The objective is to reduce the current limitations caused by the proliferation of data sources deploying a variety of modalities, information models and encoding syntaxes. This will enhance applicability and performances of document technologies within pedagogically consistent scenarios.

The LimSee3 project aims at defining a document model dedicated to adaptive and evolutive multimedia authoring tools, for different categories of authors and applications, to easily generate documents in standard formats (see the authoring process showed in Fig. 1). Our approach is to focus on the logical structure of the document while keeping some semantics of proven technologies such as SMIL [7]. This provides better modularity, facilitates the definition of document templates, and improves manipulation and reusability of content.

This paper is organized as follows: Sect. 2 presents a scenario example that will be developed throughout the paper and thereby analyzes CoPs requirements for authoring multimedia documents. We then define the main concepts on which multimedia authoring tools are based and we classify existing approaches in the light of these concepts. Section 4 introduces our LimSee3 document model and Sect. 5 shows how it can be used for the development of authoring tools tuned for specific CoPs. Last section presents the current state of our development and our perspectives in the context of the Palette project.

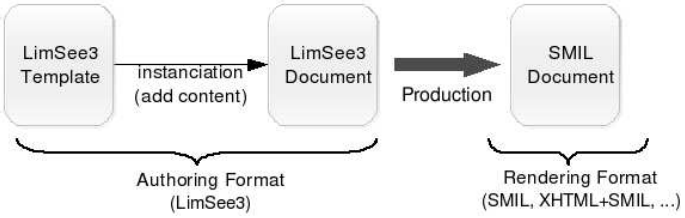

Fig. 1. The authoring process in LimSee3

2 Real-Life Example and Requirements of CoPs

The instrumentation of CoPs heavily relies on communication technologies. In this paper we are concerned with communication through sharing and collaborative authoring of information. We are studying scenarios where experience and knowledge are shared by means of multimedia data, such as annotated video or synchronized slideshow. The key point is that in CoPs, content readers are also content creators but usually have no skills in multimedia authoring. We develop below a concrete scenario of how a particular CoP shares information and then we identify the main requirements of multimedia authoring in such situations.

2.1 Experience sharing between reps

Studies of experiences at companies such as Xerox [8] have demonstrated that CoPs, as the copier repair technicians ("tech reps") CoP, are a very effective way for professionals to share informal or tacit knowledge gained from experience in the field. This sharing of tips, which could not be found in training manuals or classroom settings, was critical to help the tech reps do a better job and was even ultimately fostered by Xerox.

The practice of creating and exchanging stories has two important aspects. First of all, telling stories helps to diagnose the state of a troublesome machine. Reps begin by extracting a history from the users of the machine and with this and the machine as their starting point, they construct their own account. If they cannot tell an adequate story on their own then they seek help from specialists or colleagues (over coffee or lunch).

Brown took example on one service call observed by the ethnograph Orr in [12]. A rep confronted a machine that produced copious raw information in the form of error codes and obligingly crashed when tested. As the error codes and the nature of the crashes did not correspond, the case immediately fell outside the directive training and documentation provided by the organization. Unfortunately, the problem also fell outside the rep's accumulated, improvised experience ; his technical specialist was equally baffled. Solving the problem in situ required constructing a coherent account of the malfunction out of the incoherence of the data and the documentation. To do this, the rep and the specialist embarked on a long story-telling procedure. They explored the machine

or waited for it to crash for collecting data such as logs, screenshots, sound records. The rep and specialist recalled and discussed other occasions on which they had encountered some of the present symptoms via phone calls, webcam records, user feedback... Each story presented an exchangeable account that could be examined and reflected upon to provoke old memories and new insights. Yet more tests and more stories were thereby generated. The story-telling process continued forming a purposeful progression from incoherence to coherence.

Ultimately, these stories generated sufficient interplay among memories, tests, the machine's responses, and the ensuing insights to lead to diagnosis and repair. Through story-telling, these separate experiences converged, leading to a shared diagnosis of previously encountered but unresolved symptoms. Rep and specialist were now in a position to modify previous stories and build a more insightful one. They both increased their own understanding and added to their community's collective knowledge. A story, once in the possession of the community, can then be used – and further modified – in similar diagnostic sessions.

The information units that are exchanged in this particular CoP are multimedia story documents that are composed of sequences of story steps where data elements are heterogeneous and multimedia. The challenges are to enrich information with the synchronization of data elements (for instance a phone call with the corresponding webcam excerpt) and to provide a document structure enabling knowledge sharing and reusability (of experience stories).

2.2 Basic requirements

The cooperative platform to be provided to the CoPs must have the two following basic features: (i) authoring tool of stories dedicated to tech reps ; (ii) access tool to read the existing stories on different devices (desktop PC, PDA, mobile phone...). Looking more closely at the ways in which CoPs participants are producing multimedia information, we can identify some requirements for the authoring and presentation platform:

1. Simple and efficient authoring paradigms – because CoPs members are not (always) computer science technicians.
2. Easy and rapid handling of the authoring tool – because new members can join CoPs.
3. Modular and reusable content – because multimedia information results in a co-construction process between members.
4. Evolutive structuring of documents – because of the dynamic nature of CoPs.
5. Use of standard formats – because CoPs need portability, easy publishing process and platform-independence.

Basically, our approach proposes a template mechanism to cope with requirements 1 and 2, a component-based structuring enabling requirements 3 and 4, and relies on proven standard technologies to ensure the last requirement. Before further stating our authoring model, we present in the next section the main concepts and approaches of multimedia authoring on which this work is based.

3 Multimedia Documents and Multimedia Authoring

In traditional text oriented document systems, the communication mode is characterized by the spatial nature of information layout and the eye's ability to actively browse parts of the display. The reader is active while the rendering itself is passive. This active-passive role is reversed in audio-video communications: active information flows to a passive listener or viewer. As multimedia documents combine time, space and interactivity, the reader is both active and passive. Such documents contain different types of elements such as video, audio, still-picture, text, synthesized image, and so on, some of which having intrinsic duration. Time schedule is also defined by a time structure synchronizing these media elements. Interactivity is provided through hypermedia links that can be used to navigate inside the same document and/or between different documents.

Due to this time dimension, building an authoring tool is a challenging task because the WYSIWYG paradigm, used for classical documents, is not relevant anymore: it is not possible to specify a dynamic behavior and to immediately see its result. Within the past years, numerous researches have presented various ways of authoring multimedia scenarios, focusing on the understanding and the expressive power of synchronization between media components: approaches can be classified in absolute-based [1], constraint-based [9], [11], event-based [14] and hierarchical models [7], [15]. Besides, to cope with the inherent complexity of this kind of authoring, several tools [1], [4], [10] have proposed limited but quite simple solutions for the same objective. Dedicated authoring, template-based authoring and reduced synchronization features are the main techniques to provide reasonable editing facilities. But we can notice that these tools generally also provide scripting facilities to enrich the authoring capabilities and therefore loose in some way their easiness.

Beside timelines, script languages and templates, intermediate approaches have been proposed through "direct manipulation" and multi-views interface paradigms. IBM XMT authoring tool [2] and SMIL tools such as LimSee2 [3] and Grins [5] are good examples. In LimSee2, the time structure of SMIL is represented for instance in a hierarchical timeline as shown in of Fig. 2 (4). Time bars can be moved or resized to finely author the timing scenario. This kind of manipulation has proven very useful to manipulate efficiently the complex structures representing time in multimedia XML documents.

However even if XMT and SMIL are well-established languages, the above-mentioned tools are too complex for most users because they require a deep understanding of the semantics of the language (e.g. the SMIL timing model). Moreover these models generally put the time structure at the heart of the document whereas it does not always reflect exactly the logical structure in the way it is considered by the author. Our approach instead sets this logical dimension as the master structure of the document, which is a tree of modular components each one specifying its own time and spatial structures. Additionally, the document can be constrained by a dedicated template mechanism.

A template document is a kind of reusable document skeleton that provides a starting point to create document instances. Domain specific template systems

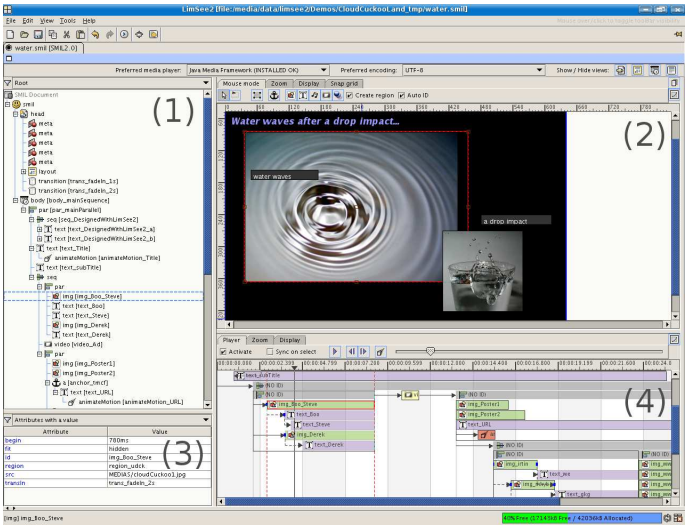

Fig. 2. Multiview authoring in LimSee2

are a user-friendly authoring solution but require hardly extensible dedicated transformation process to output the rendering format. We chose on the contrary to tightly integrate the template syntax in the document: the template is itself a document constrained by schema-like syntax. The continuum between both template and document permits to edit templates as any other document, within the same environment, and enables an evolutive authoring of document instances under the control of templates. There is no need to define a dedicated language to adapt to each different use case.

We believe that the combination of document structuring and template definition will considerably help CoPs in (i) reusability of materials, (ii) optimization of the composition and life cycle of documents, (iii) development and transmission of knowledge, (iv) drawing global communities together effectively.

4 The LimSee3 Authoring Language

4.1 Main Features

In the LimSee3 project, we define a structured authoring language independently of any publication language. Elements of the master structure are components that represent semantically significant objects. For instance a story report document is a list of step components. Each step is composed of several media objects and describes a phase of the story (failure description, machine exploration...). Components can be authored independently, integrated in the document structure, extracted for reusability, constrained by templates or referenced by other components.

The different components of a multimedia document are often tightly related one with another: when they are synchronized or aligned in space, when one contains an interactive link to another, and so on. Our approach, which is close to the one proposed in [13] is for each component to abstract its dependencies to external components by giving them symbolic names that are used in the timing and layout sections. This abstraction layer facilitates the extraction of a component from its context, thus enhancing modularity and reusability.

Finally, the goal was to rely on proven existing technologies, in both contexts of authoring environments and multimedia representation. The timing and positioning models are wholly taken from SMIL. Using XML provides excellent structuring properties and enables the use of many related technologies. Among them are XPath, used to provide fine-grained access to components, and XSLT, used in templates for structural transformation and content generation.

The authoring language is twofold: it consists in a generic document model for the representation of multimedia documents, and it defines a dedicated syntax to represent templates for these documents.

4.2 Document Model

A *document* is no more than a `document` element wrapping the root of the object hierarchy and a `head` element containing metadata. This greatly facilitates the insertion of the content of a document in a tree of objects, or the extraction of a document from a sub-tree of objects.

A *compound object* is a tree structure composed of nested objects. Each compound object is defined by the `object` element with the `type` attribute set to `compound`. It contains a `children` element that lists children objects, a `timing` element that describes its timing scenario and a `layout` element that describes its spatial layout.

The value of the required `localId` attribute uniquely identifies the component in the scope of its parent object, thereby also implicitly defining a global identifier `id` when associated with the `localId` of the ancestors. In Example 1, the first child of object `step1` has the local id `copyLog` and hence is globally identified as `step1.copyLog`.

The timing model, and similarly the positioning model, is taken from SMIL 2.1. The `timing` element defines a SMIL time container. The timing scenario of a component is obtained by composition of the timed inclusions defined by the `timeRef` elements, whose `refId` attributes are set to local ids of children.

```
<document xmlns="http://wam.inrialpes.fr/limsee3/"
  xmlns:smil="http://www.w3.org/2005/SMIL21/">
  <head><!-- some metadata --></head>
  <object localId="step1" type="compound">
    <children>
      <object type="text" localId="copyLog">...</object>
      <object type="image" localId="screenshot">...</object>
      <object type="compound" localId="AnnotatedVid">...</object>
```

```
    </children>
    <timing timeContainer="par">
      <timeRef refId="AnnotatedVid" begin="0s"/>
      <smil:seq begin="0s">
        <timeRef refId="screeshot"/>
        <timeRef refId="copyLog"/></smil:seq></timing>
    <layout height="100" width="100">
      <layoutRef refId="AnnotatedVid" left="0"/>...</layout>
</object></document>
```

Example 1. **A simple story step LimSee3 document**

A *media object* is actually a simple object that wraps a media asset, i.e. an external resource (such as an image, a video, an audio track, a text...) referenced by its URI. It is defined by the `object` element with the `type` attribute set to either `text`, `image`, `audio`, `video` or `animation`. The URI of the wrapped media asset is the value of the `src` attribute. Example 2 shows a text media object with local id `menuItem1` which wraps the media asset identified by the relative URI `./medias/item1.txt`.

Area objects inspired from the SMIL `area` element can be associated with media objects. They are used for instance to structure the content of a media object or to add a timed link to a media object. An area is defined as an `object` element with the `type` attribute set to `area`. For instance, in Example 2 the media object `menuEntry1` has a child area which defines a hyperlink.

Relations of dependency between objects are described independently of their semantics in the document. External dependencies are declared with `ref` elements grouped inside the `related` child element of objects. The value of `refId` of a `ref` element is the id of the related element and the value of `localId` is a symbolic name that is used within the object to refer to the related object. For instance, in Example 2, object `menuItem1` describes a text that links to the object `story.step1`, by first declaring the relation in a `ref` element and then using this external object locally named `target` to set the value of the `href` attribute of the link, using `attribute` and `value-of` elements taken form XSLT.

```
<object localId="menuItem1" type="txt" src="./medias/item1.txt">
  <related><ref localId="target" refId="story.step1"/></related>
  <children><object type="area" localId="link"/></children>
  <timing>
    <attribute name="begin">
      <value-of refName="target" select="@id"/>.begin</attribute>
    <timeRef refId="link">
      <attribute name="href">
        #<value-of refName="target" select="@id"/></attribute>
</timeRef></timing>...</object>
```

Example 2. **A LimSee3 object with external dependency relations**

4.3 Templates

Template nodes aim at guiding and constraining the edition of the document. In order to have better control and easy GUI set up, the language includes two template nodes: media zone and repeatable structure.

A *media zone* is a template node that defines a reserved place for a media object. It is represented by the `zone` element, that accepts a `type` attribute (`text`, `img`, `audio`, `video`, `animation`, `any`, or a list of these types) to define what types of media object can be inserted in this zone. The author can also specify content that will be displayed to invite the user to edit the media zone with the `invite` element (of any media type). For instance Example 3 shows a media zone for an image, with textual invitation. During the authoring process `zone` elements are filled with media objects inserted by the user.

A *repeatable structure*, represented by the `objList` element, is a template node that defines a homogeneous list of objects. Each item of the list matches a model object declared in the `model` child of the list. The cardinality of the list can be specified with the `minOccurs` and `maxOccurs` attributes. Example 3 shows a story template document based on an `objList` named `step-list`, and partially instanciated with three compound objects respecting the `step` model. Thanks to the use of XSLT-like syntax, the timing scenario can be specified independently of the content of children instances.

It is possible to lock parts of a document with the `locked` attribute, to prevent the author from editing anything. This permits for instance to guide more strongly inexperienced users by restricting their access to the only parts of the document that make sense to them.

```
<object localId="story" type="compound">
  <children>
    <objList localId="step-list" maxOccurs="20">
      <model name="step">
        <object type="compound">...</object></model>
      <object type="compound" localId="step1">...</object>
      <object type="compound" localId="step2">...</object>
      <object type="compound" localId="step3">...</object>
  </objList></children>
  <timing>
    <smil:seq begin="1s">
      <for-each
        select="children/objList[@name="step-list"]/object">
        <timeRef>
          <attribute name="refId">
            <value-of select="@localId"/>
    </attribute></timeRef></for-each></smil:seq>
</timing>...</object>
```

Example 3. **A partially instanciated story template**

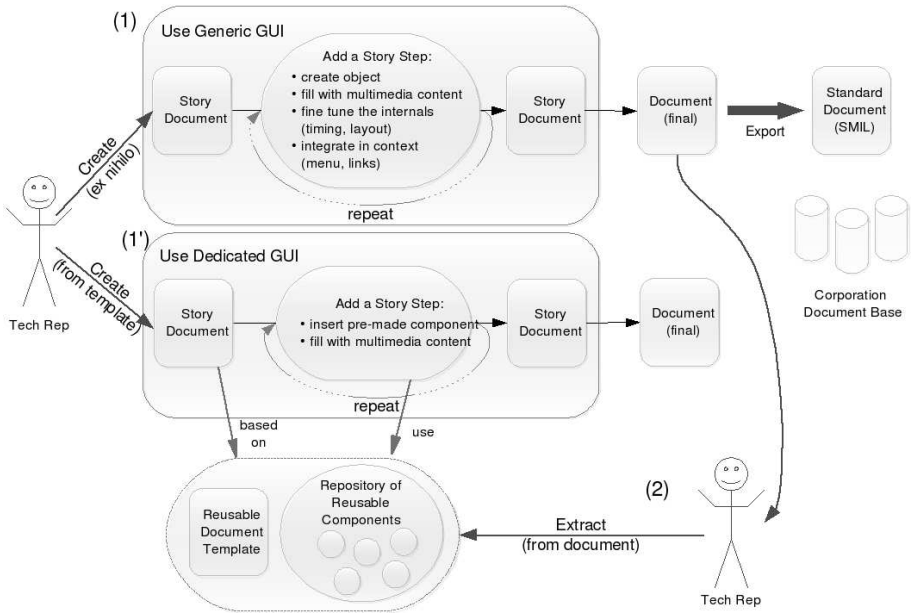

Fig. 3. Authoring with LimSee3

5 Authoring with LimSee3

Figure 3 (2) also shows the creation of a template document from an existing document. The main structure of the document, in this case a sequence of story steps, can be constrained by template nodes such as repeatable structures. Additionally, inter-object relations described in Sect. 4 facilitate the extraction of components from their context so that they can be reused in other documents. In the tech reps CoP, a possible workflow is to first create a story report from scratch (1), then to extract a template document from this report (2), along with a dedicated GUI, to ease the creation of further story reports (1'). This is a typical example of participative design leading to the development of a dedicated tool based on the LimSee3 generic platform.

The LimSee3 model leads to the development of authoring tools that fit the requirements of Sect. 2.2. We are defining a generic platform that permits to manipulate all the elements defined in the model (documents, compound objects, timing and layout details, relations...). It provides features based on the proven authoring paradigms described in Sect. 3 such as multi-views, timeline, structure tree an 2D canvas. In the reps CoP example described in Sect. 2, a tech rep could have used the generic GUI to create the story report ex-nihilo, as shown in Fig. 3 (1), incrementally adding story steps by creating and integrating new objects in the document (resulting in the LimSee3 document of Example 1). Once fully authored, the story report can be persistently added to the base

of documentation maintained by the company, and published on demand to any output format (provided its semantics is included in the document model).

Another approach is to use a domain-specific template with dedicated GUI, as shown in Fig. 3 (1'). For instance, a template for a story report could consist in a repeatable structure of story steps. These steps could be instanciated from existing template components such as an audio zone for phone calls, a text zone for machine logs, ... The constraints of the template would guide the tech rep in the creation of the document, reflected in the GUI by dedicated buttons or menu items such as "add a story step", "insert a phone call record", or a form-based interface for adding titles or comments to multimedia content. In the underneath manipulated model, the tight integration of template nodes in the document ensures a smooth evolution from the template to the final document.

6 Conclusion

The model presented in this paper develops a practice-based approach to multimedia authoring dedicated to communities where collaborative and participative design is of high importance. It improves reusability with template definitions and with the homogeneous structuring of documents. This document model is being implemented as cross-platform java software. In the context of Palette, we will use this model to develop dedicated authoring tools for pedagogical CoPs.

References

1. *Adobe Authorware 7 and Director MX 2004.* http://www.adobe.com/products/.
2. *Authoring in XMT.* http://www.research.ibm.com/mpeg4/Projects/AuthoringXMT/.
3. *LimSee2.* http://wam.inrialpes.fr/software/limsee2/.
4. *MS Producer for PowerPoint.* http://www.microsoft.com/office/powerpoint/producer/.
5. *Oratrix Grins.* http://www.oratrix.com/.
6. *Palette.* http://palette.ercim.org/.
7. *W3C SMIL.* http://www.w3.org/AudioVideo/.
8. J. S. Brown and P. Duguid. Organizational learning and communities-of-practice: Toward a unified view of working, learning and innovation. *Journal Information for Organization Science*, 2(1):40–57, 1991.
9. M. Buchanan and P. Zellweger. Automatic temporal layout mechanisms. In *ACM Multimedia'93.*
10. X. Hua, Z. Wang, and S. Li. LazyCut: Content-aware template based video authoring. In *ACM Multimedia'05.*
11. M. Jourdan, N. Layaïda, C. Roisin, L. Sabry, and L. Tardif. Madeus, an authoring environment for interactive multimedia documents. In *ACM Multimedia'98.*
12. J. Orr. *Sharing Knowledge, Celebrating Identity: War Stories and Community Memory in a Service Culture*, pages 169–189. Sage Publications.
13. H. Silva, R. Rodrigues, L. Soares, and D. M. Saade. NCL 2.0: integrating new concepts to XML modular languages. In *ACM DocEng'04.*
14. P. Sénac, M. Diaz, A. Léger, and P. de Saqui-Sannes. Modeling logical and temporal synchronization in hypermedia. *IEEE J. on Sel. Areas in Comm.*, 14(1), 1996.
15. G. van Rossum, J. Jansen, K. Mullender, and D. Bulterman. CMIFed : a presentation environment for portable hypermedia documents. In *ACM Multimedia'93.*