Improving Formatting Documents by Coupling Formatting Systems

Fateh Boulmaiz
WAM Project, INRIA Rhône-Alpes
Zirst, 655 Avenue de l'Europe
38330 Montbonnot, France
Fateh.Boulmaiz@inrialpes.fr

Cécile Roisin
WAM Project, INRIA Rhône-Alpes
Zirst, 655 Avenue de l'Europe
38330 Montbonnot, France
Cécile.Roisin@inrialpes.fr

Frédéric Bes
WAM Project, INRIA Rhône-Alpes
Zirst, 655 Avenue de l'Europe
38330 Montbonnot, France
Frederic.Bes@inrialpes.fr

ABSTRACT

In this paper, we present a framework for coupling an existing formatting system such as SMIL [7] and Madeus [13] with a formatting control system XEF [10]. This framework allows the coupling process to be performed at two levels: 1) the language level, which is concerned with how to link the control features of XEF and the elements of an existing formatting system, and 2) the formatter level, which deals with the creation of a new formatter by formatter composition.
The overall objective is to provide more powerful and flexible formatting services to cover new needs such adaptive and/or generated presentations.

Categories and Subject Descriptors

1.7.2 [**Document and text processing**]: Document Preparation, Languages and systems, Markup languages, Hypertext/hypermedia.

General Terms

Documentation, Languages.

Keywords

presentation language, language coupling, software coupling.

1. INTRODUCTION

With the increasing complexity of multimedia documents, the emergence of heterogeneous devices (cellular phones, personal digital assistant, desktop computers,...) and the diversification of author and user expectations, several presentation languages emerged to meet these needs. However, these languages often fail to provide satisfactory solutions. Failures occur when there is no solution that satisfies the set of presentation properties or when an unexpected formatting result is computed. The latter situation occurs when flexibility provided by the presentation language can allow multiple formatting solutions.
Recently, some research has focused on the use of constraint techniques to avoid formatting failure, including CSVG [9] and CCSS [8] for spatial formatting and Isis [12] and Madeus [13] for temporal formatting. These systems are useful for instance to compute global properties, but unfortunately remain limited. On one hand, the elaboration of constraint-based formatters is more difficult. Indeed, the flexibility that constraint-based languages offer makes the dependencies between objects, relations and dimensions very complex to integrate into formatters. Moreover, constraint techniques require considerable computing time. On the other hand, actual constraint-based languages do not provide authors any way to control formatting process.
Our approach [10] addresses these formatting problems through the definition of presentation operators that allow the author to better control the formatters. The principle of this approach is based on the following:

- new presentation properties can be added to existing presentation languages to allow the author to express priorities, more abstract properties and fall-back positions. These properties are independent of any presentation language and can thus be used with any language.
  - Priorities allow authors to specify preferences used by the formatter. The formatter will then be able to apply the correct choice among alternatives or to decide to compute the properties again with new values when a formatting failure occurs.
  - Abstract properties allow the specification of a global or partial policy over the formatting system such as the maximum number of simultaneous images in a presentation or the duration of the presentation.
  - Fall-back positions are a set of formatting specifications defined by the author to cover failure situations. Compared to alternatives as defined in SMIL and XSL-FO (through the *switch* element), XEF alternatives permit the selection of an alternative independent of the values of the various predefined or custom test attributes but dependent on the type of failure event.
- a constraint-based formatter that handles these properties.

Our goal is to construct a formatting system by composing documents specified in different presentation languages and software components (formatters) that are mutually dependent. More precisely, we aim to use the XEF system to render existing systems more flexible and more controllable. Achieving this goal will improve multimedia presentations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DocEng'03, November 20–22, 2003, Grenoble, France.
Copyright 2003 ACM 1-58113-724-9/03/0011 ...$5.00.

because authors will be able to more finely control the formatting process and because existing formatters will become more powerful.
To achieve this goal, the task can be divided into two related levels. At the language level, several solutions exist for coupling XML-based languages, such as the use of a selection language and transformation techniques. In Section 3.1, we review briefly how to use XPath to carry out this link. At the formatter level, techniques such as Component-oriented Development [14] have been used to solve similar problems in other contexts. In order to apply these techniques in the context of formatters we need to identify the precise needs in coupling formatters. Section 3.2 gives an outline of these problems and the solution we propose. In order to test the feasibility of our architecture, we have implemented our solution using a SMIL formatter (cf.§.4).

2. RELATED WORK

The first work on compound documents can be traced back to the OpenDoc [4] lead by Apple. Recently, the issue arose again with new technologies like OLE (Object Linking and Embedding), and Bonobo (the Gnome component model) [1]. With the introduction of namespaces and DOM, the W3C has made an important step towards the integration of different markup languages into the same document. With regard to software composition, a W3C activity [6] on component extensions has been proposed, though it is still pending.
Recently, tools such as X-Smiles [3] have emerged that use several formatters to format a compound document. However, none of these technologies is explicitly meant to be used as a general framework for coupling formatting systems. They are targeted at a coarser granularity and the communication between formatters is restricted to the exchange of simple data, *i.e.* data necessary for rendering the formatted document fragment for which each formatter is responsible.

3. ARCHITECTURE OVERVIEW

Figure 1 depicts the different components of our coupling system. A multimedia presentation is specified in a presentation language such SMIL (which we call the *source document*). XEF properties used to control multimedia presentation are described in a separate document (which we call the *control document*). Formatting is performed by a composite formatter which is defined in terms of interconnected formatters (the source language formatter and the XEF formatter). The communication between formatters is carried out through interfaces that must be generic enough for the architecture to be suitable for a variety of formatting systems.
In the remainder of this section, we present the structure of each component. We first introduce the coupling of languages (control document structure) and then the composite formatter. We then describe the communication and cooperation between formatters and conclude with a brief discussion of our design choices.

3.1 Language coupling

We use XPath [5] language to link XEF properties of the control document with elements or relations of source document. XPath provides a powerful mechanism to address parts of an XML document. It is also designed so that it has a natural subset that can be used for testing whether or not a node matches a pattern.
For example, adding an abstract property to a temporal sequence identified by the XPath expression */smil/body/seq* in the source document consists in adding the control rule */smil/body/seq : balancing="proportional (p1)"* which means that all the media defined in the priority *p1* will be balanced according to its level of priority.

3.2 Formatter coupling

3.2.1 Formatter coupling problems

The main challenge in designing our system is to manage effective communication between formatters. By effective communication, we mean that the system should be able to detect and resolve problems relating to *dependencies* between XEF's elements and source language's elements. Such dependencies can involve inconsistencies in the source document. Indeed, adding a XEF control is equivalent to an authoring operation, and it has been shown in other research [13], that incremental authoring may lead to temporal or spatial inconsistency because of temporal/spatial parameters of objects (duration, begin-end dates, space positions) must satisfy invariants related to the time progression and causality. For example, consider two media objects A and B for which it is specified that A *meets* B and B *finishes* A. In the event of failure of the formatter, the application of XEF's control *reply=reduction (B)* can lead to a qualitative inconsistency if the end of B occurs before the end of A. So formatter coupling requires that the coupling system provides facilities to insure that for every modification introduced by one of the two formatters, the overall document remains consistent. Dependencies between elements and relations pose another well known problem, that of *cycles*. To cope with these two related problems we have proposed a solution that handles source nodes by groups of dependent ones, as described in the section below.
Finally the system must deal with the identification of the *stability state* in which the system state is identical to that of the previous processing stage, thus no more processing stage is necessary.

3.2.2 Composite formatter architecture

As shown in Figure 1, the composite formatter is composed of three components. The first component is *source language formatter* processes the source document in two steps. First, the source document is parsed and converted into an internal data structure represented in the form of a DOM tree, called the *raw source DOM*. Next, the raw source DOM is processed and converted into a *formatted source tree*. It should be noted that in this stage, all necessary attribute values for rendering the source document are calculated. The second component is the *XEF formatter* which parses the control document and generates a DOM tree containing XEF nodes and selector nodes. This tree is used by the third component of our architecture, the *XEF/source connector*, to establish the link between formatters. This connector is responsible for two functions. In its first stage, it extracts the source document's nodes specified by the control document. The second stage translates the representation of these nodes from the source language to the XEF language.

Once the nodes are extracted and transformed into a raw

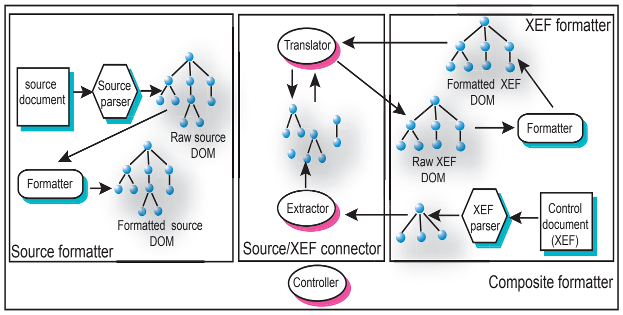

Figure 1: Coupling system overview

Table 1: Summary of obtained results

	example 1	example 2
number of dependencies between nodes	26	68
SMIL formatting duration(ms)	1263	1420
transitive extraction duration(ms)	145	217
total formatting duration (ms)	1908	2218

XEF DOM tree (that contains only XEF nodes), the XEF DOM tree is handled by the XEF engine. The formatted nodes are then forwarded to a translator, where they are translated to source language nodes and they are reinjected into the source document tree. This last process is performed directly thanks to the use of node references maintained by the extractor.
To overcome the above problems we propose to use *transitive extraction*. This consists of the extraction and translation the formatted source nodes for which one or several XEF controls are specified *and* the extraction and translation of all nodes having dependencies on them.

4. EXPERIMENTAL RESULTS

We have implemented in Java a prototype of this architecture for the system LimSee [11], a SMIL authoring tool that provides many advanced edition views (hierarchical, attributes and timeline views). In practice, no modification was carried out on the SMIL formatter.
Table 1 reports some numerical results obtained with our coupling system for 2 medium size SMIL documents (respectively 54 SMIL nodes and 75 SMIL nodes). The author has used XEF to express an abstract property and a fallback specification that limit the duration of the presentation. See [2] for the complete specification of these examples.
The results of this experiment show that XEF properties and the XEF formatter support increased SMIL expressivity while avoiding failure situations in the SMIL formatter (in this case, a document whose duration is too long or a media element that terminates earlier than expected).
As can be seen in the Table, the coupling system succeeds in formatting the controlled documents with about 1/3 of supplementary time compared to the SMIL formatter alone. Moreover the table shows that for these examples, transitive extraction does not augment significantly the computation time even when the number of dependencies increase substantially (see example 2).

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a framework that addresses new challenges for multimedia document presentation by coupling formatting systems. One originality of our work is that it proposes to use existing formatters without any modification to bring to them.
The solution proposed for coupling formatters belongs to the so-called "light-coupling class". Indeed coupling is performed through data (thanks to the XPath expressions computed by the extractor) and the scheduling is a successive execution of the two formatters starting from the source one.
The experimentation has shown that transitive extraction is a good approach for documents in which dependencies remain local (as in SMIL documents). However, we have not tested our system in enough various situations. In particular, we encountered no cases where there is a need to execute several times our formatting process, so we have not yet studied any stability detection process.
Finally, one of our objectives was to propose a solution independent from any source language and source formatter but with the hypothesis that it can provide a DOM access for a formatted tree. We were not able to completely achieve this goal but we have limited it inside a unique component, namely the translator component.
Next steps of this work will be based on a more intensive use of the system to evaluate formatting costs related to the coupling and investigate other scheduling solutions.

6. REFERENCES

[1] Gnome office. http://www.gnome.org/gnome-office/bonobo.shtml.
[2] http://wam.inrialpes.fr/people/boulmaiz/publication.
[3] X-smiles an open xml-browser for exotic devices. http://www.xsmiles.org.
[4] OpenDoc Programmer's Guide. Addison-Wesley Editions, 1996.
[5] Xml path language (xpath), w3c recommendation. http://www.w3.org/TR/xpath.html, Nov 1999.
[6] Component extension (cx) api requirements (1.0). W3C Note, http://www.w3.org/TR/CX, Dec 2001.
[7] Synchronized multimedia integration language (smil 2.0), w3c recommendation. http://www.w3.org/TR/smil20, 2001.
[8] G. Badros and al. Constraint cascading style sheets for the web. In *ACM Conf on User Interface Software and Technology*, Nov 1999.
[9] G. Badros and al. A constraint extension to scalable vector graphics. In *WWW10*, May 2001.
[10] F. Bes and al. A presentation language for controlling the formatting process in multimedia presentations. In *DocEng'02*, Nov 2002.
[11] C.Roisin and al. Editing smil with timelines. *SMILEurope2003*, Feb 2003.
[12] M. Kim and al. Multimedia documents with elastic time. In *Third ACM Int Conf on Multimedia*, San Francisco, Nov 1995.
[13] N. Layaïda and al. Maintaining temporal consistency of multimedia documents using constraint networks. In *Int.Conf on Multimedia Computing and Networking*, pages 124–135, Jan 1996.
[14] C. Szyperski. Component software - beyond object-oriented programming. *ACM Press*, 1998.