Towards Second and Third Generation Web-Based Multimedia

Jacco van Ossenbruggen, Joost Geurts, Frank Cornelissen, Lynda Hardman and Lloyd Rutledge
Centrum voor Wiskunde en Informatica (CWI), Amsterdam
Jacco.van.Ossenbruggen@cwi.nl

ABSTRACT

First generation Web-content encodes information in handwritten (HTML) Web pages. Second generation Web content generates HTML pages on demand, e.g. by filling in templates with content retrieved dynamically from a database or transformation of structured documents using style sheets (e.g. XSLT). Third generation Web pages will make use of rich markup (e.g. XML) along with metadata (e.g. RDF) schemes to make the content not only machine readable but also machine processable — a necessary pre-requisite to the *Semantic Web*.

While text-based content on the Web is already rapidly approaching the third generation, multimedia content is still trying to catch up with second generation techniques. Multimedia document processing has a number of fundamentally different requirements from text which make it more difficult to incorporate within the document processing chain. In particular, multimedia transformation uses different document and presentation abstractions, its formatting rules cannot be based on text-flow, it requires feedback from the formatting back-end and is hard to describe in the functional style of current style languages.

We state the requirements for second generation processing of multimedia and describe how these have been incorporated in our prototype multimedia document transformation environment, *Cuypers*. The system overcomes a number of the restrictions of the text-flow based tool sets by integrating a number of conceptually distinct processing steps in a single runtime execution environment. We describe the need for these different processing steps and describe them in turn (semantic structure, communicative device, qualitative constraints, quantitative constraints, final form presentation), and illustrate our approach by means of an example. We conclude by discussing the models and techniques required for the creation of third generation multimedia content.

Keywords: Multimedia, Transformations, XSLT, XML, SMIL.

WWW10, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

1. INTRODUCTION

The evolution of the Web is sometimes described in terms of first, second and third generation Web content [10]. In the first generation, the Web browser provided its users a uniform interface to a wide variety of information on the Internet. URIs provide a simple but universal naming scheme, and HTTP a simple but fast transfer protocol. In theory, HTML was designed to provide the "glue" between various information resources in the form of hyperlinks, and as a default document format Web servers could resort to when other available formats were not understood by the client. In practice, however, HTML turned out as being the format that was also used to put the bulk of the content on the Web [3]. A major problem of the first generation Web content was the fact that all this HTML content was manually written. This proved to be too inflexible when dealing with content that is stored in existing databases or that is subject to frequent changes. For larger quantities of handwritten documents, keeping up with changing browser technology or updating the "look and feel" proved to be hard.

In the current, second generation Web, the required flexibility is provided by a range of technologies based on automatic generation of HTML content. Approaches vary from filling in HTML templates with content from a database back-end to applying CSS and XSLT style sheets to give the content the appropriate look and feel while storing the content itself in a form free of presentation and browser related details. Current trends on the Web make the flexibility provided by the second generation Web technology even more relevant. The PC-based Web browser is no longer the only device used to access the Web. Content providers need to continuously adapt their content to take into account the significant differences among Web access using PCs and alternative devices ranging from small-screen mobile phones and hand-held computers to set-top boxes for wide-screen, interactive TV [13]. Additional flexibility is required to take into account the different languages, cultural backgrounds, skills and abilities of the wide variety of users that may access their content.

By providing flexibility in terms of the presentation and user interaction, second generation Web technology primarily addresses the needs of human readers. In contrast, third generation Web technology focuses on content that is both human *and* machine processable. Machine-processable content is a pre-requisite for the more intelligent services that constitute the "Semantic Web" as envisioned by Tim Berners-Lee and others [3]. To provide real machine-processable content, next generation Web technology primarily needs

to extend interoperability on the semantic level. Current Web recommendations focus on either syntactic issues or on semantics for a generic domain. Examples of such generic domains that are covered by current W3C specifications include the semantics of presentation (CSS, XSL), interaction (XLink, XForms), privacy (P3P) and content rating (PICS). The third generation, however, needs to provide interoperability in terms of application and domain-dependent semantics. A first step in this direction has already been taken by W3C specifications such as XML and RDF.

New models and tools to improve the support for second and third generation Web currently receive ample attention, both in research and commercial environments. Most of this attention, however, is directed towards text-oriented applications. *Multimedia* content — that is, content that seamlessly integrates multiple media types in a synchronized, interactive presentation — has some characteristics that are fundamentally different from text. These differences mean that the models and tools that are developed for text cannot be readily applied to multimedia. In this article we claim that — while the need for second and third generation multimedia content is similar to that for textual content — the technical requirements to support this need are substantially different.

The structure of the remainder of the article is as follows. First, we analyze the requirements for multimedia presentation generation, focusing on the differences between text and multimedia document transformations. Then we describe the different levels of abstraction that characterize the *Cuypers* multimedia presentation generation system (the system is named after the Dutch architect who designed several famous buildings in Amsterdam, including the Rijksmuseum and Central Station). We discuss the use of these abstraction levels in the context of an example scenario. We conclude with an overview of related work and a description of future work.

2. REQUIREMENTS FOR SECOND GENERATION MULTIMEDIA CONTENT

Many of the advantages of generated Web content over manually authored Web content are commonly known and well described in the research literature [22, 25]. When the content and its underlying structure are stored separately from the details of a specific presentation of that content, tools can be developed to automatically adapt the presentation to the current situation, both in terms of the capabilities of the technical infrastructure and the specific needs of the user. These advantages not only apply to text, but perhaps even more to multimedia. One can argue that adaptation to the available network bandwidth, presentation capabilities of the end-user's platform, preferred language and preferred media types is even more important for complex, interactive multimedia than for content that is mainly text-based. In this section, we explain the differences between the requirements for second generation text and multimedia content.

Standards such as SGML [14] and XML [6] use embedded markup to encode documents in a presentation-independent way in order to increase longevity, reusability and flexibility. The visual appearance of these so-called *structured documents* is defined by the specification of a *style sheet*. Style sheets effectively define a mapping between the abstractions in the document structure and those in the presentation structure. This mapping is defined using a style sheet language. In general, efficient run-time execution and standardization of the style language (in order to be able to interchange documents) is considered more important than the language's flexibility and expressiveness. Additionally, for most applications, the mapping can be described, relatively straightforwardly, in a functional way — standardized style and transformation languages such as DSSSL [15] and XSLT [7] are functional languages.

Style sheets as described above mainly come in two flavors: template-driven and content-driven. Template-driven style sheets first set up the designed layout, and then fill in the content by querying the source document or underlying database. This works well when

1. the underlying structure of the content is sufficiently known to allow effective querying,
2. the structure of the generated presentations is comparable and known in advance and
3. the presentation structure is independent of the results returned by the query.

For example, the type and size of the information returned by each query and the number of items queried for are required to be known in advance.

In contrast, in content-driven style sheets the size and global structure of the generated presentation is not known in advance, so it cannot be defined by a template specification. Instead, the style sheet defines a set of transformation rules that will be applied to the source document. Again, certain aspects of the structure of the underlying content should be known, this time not to allow querying, but to allow the definition of an effective *selector* which determines to which element(s) each rule applies. In content-driven style sheets, the structure and size of the generated presentation varies largely with the structure and size of the underlying source document. For most textual documents, this is not a problem, since most textual elements can be flexibly nested and chained. Strict constraints on the size of the resulting presentations are also rare: for online presentations, scrollbars make the page length irrelevant, and for paged-media, extra pages can always be added.

Today's style languages are, however, not suited for those rare cases that size does matter. It is, for example, extremely hard (if not impossible) to write a style sheet that formats this HTML paper and makes sure it exactly meets the conference's 10-page limit. For many text-based applications, however, these constraints do not apply, and the techniques described above work well. For almost all multimedia applications, however, spatio-temporal positioning is not this flexible. In addition, there are a number of other issues that prevent the use of text-oriented techniques for multimedia document generation:

1. multimedia uses different document and presentation abstractions,
2. multimedia document formatting is not based on text flow,
3. multimedia transformations need feedback from the formatting back-end,
4. multimedia transformations are hard to describe in a functional language.

These four issues are discussed below.

2.1 Multimedia uses different document and presentation abstractions

The separation of the document's structure from its visual appearance is a fundamental and well known abstraction technique, both in database and structured document technology. A less common distinction is that between the specification of the document's visual appearance and its realization in terms of the final-form presentation format. For example, style languages such as XSL (and also DSSSL) define an abstract formatting object model that can be used to define the visual appearance of a presentation in a way that is independent of format-specific details of the final-form presentation. In this way, a single XSL style sheet can be defined for a specific set of documents, and, depending on the available back-ends for the XSL formatting model, the same style sheet can be applied for the generation of an online, PostScript or RTF version.

XSL style sheets essentially map document abstractions onto presentation abstractions. For text, this works well because on both levels, we have a commonly established set of abstractions to work with. On the document level, chapters, sections, headings and titles, bulleted lists, etc., are abstractions that are frequently used across different tools. On the presentation level, the same applies to abstractions such as pages, columns, inline, block-level and floating material, font families, etc. These abstractions are not only applicable across many domains, they have also proven to be extremely stable over time: while the majority of these abstractions originate from the early days of printing, they are still highly applicable to today's online documents. Even for text, however, transformations based on the abstract formatting models of XSL or DSSSL are not (yet) widely used: most tools still transform directly into target formats such as HTML or WML, by-passing the abstract formatting model entirely. While this may be partly because of the relative newness of the XSL specification, another reason is that the advantages in terms of reuse do not always outweigh the disadvantages in terms of decreased flexibility and increased complexity.

For multimedia, we still lack a commonly accepted set of abstractions, both on the document and the presentation level. The relatively slow acceptance of abstract formatting models for text, combined with rapidly changing multimedia technology and the vast range of different multimedia applications and presentation features, will make it very unlikely that this situation is going to change in the near future. This is highly unfortunate, because it means that a major advantage of today's style sheet technology — the definition of style sheets independent from the syntactic details of the target presentation format — cannot be applied to multimedia.

2.2 Multimedia document formatting is not based on text flow

For text, we have an established set of (complicated but) well understood algorithms [19] that can be used to automatically typeset a text according to the requirements of a given layout specification. To keep the style sheet itself as declarative as possible, the components implementing these relatively low level and detailed algorithms are typically part of the style engine's back-end application. These back-end components typically implement kerning, hyphenation, justification, and line and page breaking algorithms. Note that these algorithms are based on the linear structure of the underlying text. Since multimedia documents are not based on such a text flow, these algorithms do not suffice for formatting multimedia documents. For example, in text-based formatting, content that does not fit on a single page or screen is just spread out over multiple pages, or rendered on a scrollable area that is larger than the screen. These solutions are, in general, not applicable to multimedia presentations, where the very concept of a page or scrollbar often does not make sense.

In addition, many document-level and presentation-level abstractions for text are also based on text flow. For example, in a style language such as CSS, a key concept such as *relative positioning* refers to the ability to specify the position of an object relative to its default position in the text flow. Such flow-based models are, in general, not applicable to multimedia documents.

2.3 Multimedia transformations need feedback from the formatting back-end

Most style and transformation languages do not support feedback from the rendering application back into the main transformation process. For example, information about the precise position onto which a specific word is rendered, is only available after the rendering application has fully formatted the document (using the algorithms described above). Consequently this type of information is not available in the transformation process. For text, this limitation hardly ever causes problems: due to the flexibility of the text flow, the system can in most cases adjust the layout to make it meet the given constraints. For multimedia, however, the only way to determine whether a given layout specification can be realized with respect to a given set of constraints is to actually solve this set of constraints. This task is typically not performed by a (high level) transformation engine, but by a (lower level) constraint solver implemented in the back-end. For multimedia, it is thus of crucial importance to allow feedback from the lower levels of the process to the higher levels.

2.4 Multimedia transformations are hard to describe in a functional model

Transforming a presentation-independent structure to a presentation of acceptable quality is, when compared with text-centric presentations, relatively complex for media-centric presentations. These mappings are often best specified using a trial and error strategy, by backtracking over a set of alternative presentation rules, trying out different sets of constraints along the way. In contrast, most textual transformations are relatively straightforward mappings that can be better specified in a set of functional style rules. The more complex transformations that are common in multimedia are more conveniently expressed in a logic-based language with built-in support for backtracking and constraint solving.

The system introduced in the following section addresses these issues.

3. LEVELS OF ABSTRACTION IN CUYPERS

Cuypers is a research prototype system, developed to experiment with the generation of Web-based presentations

as an interface to semi-structured multimedia databases. Cuypers addresses many of the issues discussed in the previous section. First of all, it explores a set of abstractions, both on the document and on the presentation level, that are geared towards interactive, time-based and media centric presentations, rather than presentations that are based on text-flow. Second, it uses a set of easily extensible transformation rules specified in Prolog, exploiting Prolog's built-in support for backtracking. Finally, it facilitates easy feedback between the higher and lower level parts of the transformation process by executing both within the same environment. Instead of a strict separation between the transformation engine and the constraint solver, our system uses a constraint solver embedded in Prolog, so the system is able to backtrack when the transformation process generates a set of insolvable constraints.

Cuypers operates in the context of the environment depicted in Figure reffig:arch below. It assumes a server-side environment containing a multimedia database management system, an intelligent multimedia IR retrieval system, the Cuypers generation engine itself, an off-the-shelf HTTP server, and — optionally — an off-the-shelf streaming media server. At the client-side, a standard Web client suffices. The focus of this paper is the Cuypers generation engine. Given a rhetorical (or other type of semantic) structure and a set of design rules, the system generates a presentation that adheres to the limitations of the target platform and supports the user's preferences.

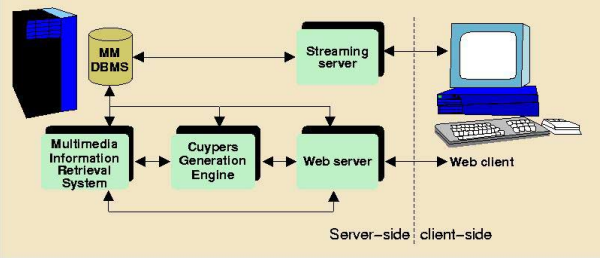

Figure 1: The environment of the Cuypers generation engine

The experience gained from the development of earlier prototypes (e.g. work done by Bailey [22]) however, proved that for most applications, the conceptual gap between an abstract, presentation-independent document structure and a fully-fledged, final-form multimedia presentation is too big to be specified by a single, direct transformation. Instead, we take an incremental approach, and define the total transformation in terms of smaller steps, which each perform a transformation to another level of abstraction. These levels are depicted in Figure 2 and include the semantic, communicative device, qualitative constraint, quantitative constraint and final-form presentation levels, resp.

Below, we describe each abstraction level and why it is needed in the overall process. We take a bottom-up approach and start with the final-form presentation level, which is the level that describes the presentation as it is delivered to the client's browser. This is also the level readers will be most familiar with, since it describes documents on the level of their encoding in for example HTML [28], SMIL [27] or SVG [11]. We subsequently add more abstraction levels,

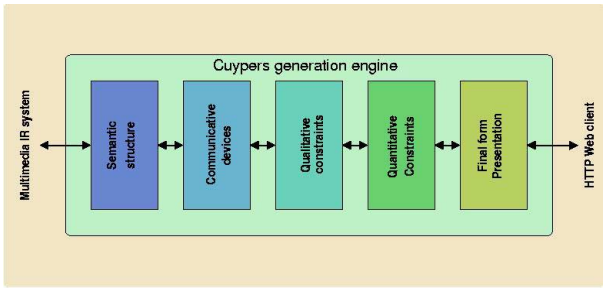

Figure 2: The layers of the Cuypers generation engine.

and end with the highest level, the "semantic level", which completely abstracts from all layout and style related information.

1. **Final-form presentation level** — At the lowest level of abstraction, we define the final-form presentation, which encodes the presentation in a document format that is readily playable by the end user's Web browser or media player. Examples of such formats include, HTML, SVG, and — the focus of our current prototype — SMIL. This level is needed to make sure that the end-user's Web-client remains independent of the abstractions we use internally in the Cuypers system, and to make sure that the end-user can use off-the-shelf Web clients to view the presentations generated by our system.

2. **Quantitative constraints level** — To be able to generate presentations of the same information using different document formats, we need to abstract from the final-form presentation. On this level of abstraction, the desired temporal and spatial layout of the presentation is specified by a set of format-independent constraints, from which the final-form layout can be derived automatically.

   An example of a quantitative constraint is "the x-coordinate of the top-right corner of picture X should be at least 10 pixels smaller than than the x-coordinate of the top-left corner of picture Y". Such constraints provide a first level of abstraction, abstracting from the syntactic details of the final-form presentation format, but also from the presentation's exact numeric layout specifications. While more abstract than the final form presentation, a specification at this level provides sufficient information for the Cuypers system to be able to automatically generate the final-form presentation. An off-the-shelf numeric constraint solver is used to determine whether or not a given layout specification can be realized, and, if so, to generate any numeric layout specifications needed. The use of constraints also gives the system the flexibility to automatically adapt to small differences in screen size between, for example, two different handhelds or mobile phones.

   In practice, larger differences cannot be solved at this level of abstraction. The use of numeric constraints is, for example, not sufficient to cater for the differences between the small screen of a mobile phone versus the

large screen of a desktop web browser. Another drawback is that it is hard to specify higher level requirements such as the fact that certain rules should be applied consistently across the entire layout. In addition, for some final-form formats this level of abstraction is just too low to be useful. For example, for the relatively flat spatial layout of a SMIL 1.0 document, the constraints given above are well suited. The same constraints, however, are too low-level to generate the complex temporal hierarchy that gives a SMIL presentation its adaptive scheduling information. On the implementation level, numeric constraints also have serious drawbacks. For example, when automatic backtracking over alternative layouts is used, a set of quantitative constraints might generate alternative layouts which are all equal, except for one coordinate, whose value only increases or decreases with one pixel (or other unit) for each generated layout.

The discussion above can be summarized by stating that numeric or quantitative constraints are necessary because solving a set of quantitative constraints is the only way to determine whether a specific layout can be realized with respect to a specific requirements. In addition, many final-form formats use numeric information to define the layout presentation. For many other purposes, however, these constraints are too low level and contain too much detail. Qualitative constraints are introduced to solve these problems.

3. **Qualitative constraints level** — An example of a qualitative constraint is "caption X is positioned below picture Y", and backtracking to produce alternatives might involve trying right or above, etc. Some final-form formats allow specification of the document on this level. In these cases, the Cuypers system only generates and solves the associated numeric constraints to check whether the presentation can be realized at all, it subsequently discards the solution of the constraint solver and uses the qualitative constraints directly to generate the final form output.

   In the Cuypers system, qualitative constraints also provide a basis for defining *meta-constraints*. Meta-constraints are necessary to specify more global properties of the resulting document, and are used with Cuypers to ensure consistency across the presentation. For example, to prevent some captions from appearing below figures and others above, a designer could add a meta-constraint specifying that all captions should appear either below or above their figures. Meta-constraints derive their name from the fact that they are implemented as constraints that constrain the set of generated constraints.

   While qualitative constraints solve many of the problems associated with quantitative constraints, they are still not suited for dealing with the relatively large differences in layout, e.g., as in the mobile phone versus the desktop browser example given above. Therefore, another level of abstraction is introduced: the communicative device.

4. **Communicative device level** — The highest level of abstraction describing the presentation's layout makes use of *communicative devices* [23]. These are similar to the patterns of multimedia and hypermedia interface design described by [21] in that they describe the presentation in terms of well known spatial, temporal and hyperlink presentation constructs. An example of a communicative device described in [23] is the *bookshelf*. This device can be effectively used in multimedia presentations to present a sequence of media items, especially when it is important to communicate the *order* of the media items in the sequence. How the bookshelf determines the precise layout of a given presentation in terms of lower level constraints can depend on a number of issues. For example, depending on the cultural background of the user, it may order a sequence of images from left to right, top to bottom or *vice versa*. Also its *overflow strategy*, that is, what to do if there are too many images to fit on the screen, may depend on the preferences of the user and/or author of the document. It may decide to add a "More info" hyperlink to the remaining content in HTML, alternatively, it could split the presentation up in multiple scenes that are sequentially scheduled over time in SMIL.

   Note that communicative devices, including the one described above, typically deal with layout strategies that involve multiple dimensions (including space, time and linking), while the constraints discussed above typically do not cross the boundaries of a single dimension. Constraints using variables along more than one dimension are called cross-dimensional constraints, and have previously been discussed in [22]. The introduction of such constraints would simplify the definition of many communicative devices and is the subject of further research.

   While the communicative device level is a very high-level description of the presentation, we still need a bridge from the domain-dependent semantics as stored in the multimedia information retrieval system to the high-level hypermedia presentation devices. To solve this problem, we introduce one last level of abstraction: the semantic structure level.

5. **Semantic structure level** — This level completely abstracts from the presentation's layout and hyperlink navigation structure and describes the presentation purely in terms of higher-level, "semantic" relationships.

   In the current Cuypers system we focus on the rhetorical aspects of the presentation, because it applies well to the application domains for which we are currently building prototypes (e.g. generating multimedia descriptions of artwork for educational purposes).

   Depending on the target application, however, other types of semantic relations can be used as well. Possible alternatives include abstractions based on the presentation's narrative structure for story-telling applications or abstractions based on an explicit discourse model. From the perspective of the Cuypers architecture, any set of semantic relations can be chosen as long as it meets the following two requirements:

   (a) it should sufficiently abstract from all presentation details so that these can be adequately adapted by the lower levels of the system, and

(b) it should provide sufficient information so that the relations can be used to generate an adequate set of communicative devices that convey the intended semantics to the end user.

The subdivision of the generation process in Cuypers is based on these different levels, with an extensible set of transformation rules to move from one level to another (see the section on implementation below). In practice, however, the transformations work by backtracking up and down different levels, and transformation rules may have access to information from several steps earlier. To explain the different abstraction levels and the associated transformation steps, the next section uses an example scenario to illustrate all levels.

4. EXAMPLE SCENARIO

In this section, we use an example scenario where the user (studying art history) just asked the system to explain the use of the *chiaroscuro* technique (strong contrast of light and dark shading) in the paintings of Rembrandt van Rijn. The system's multimedia information retrieval back-end queried its annotated multimedia database system and retrieved five images of paintings that are annotated as using this technique, the accompanying titles and a general textual description of the term *chiaroscuro*.

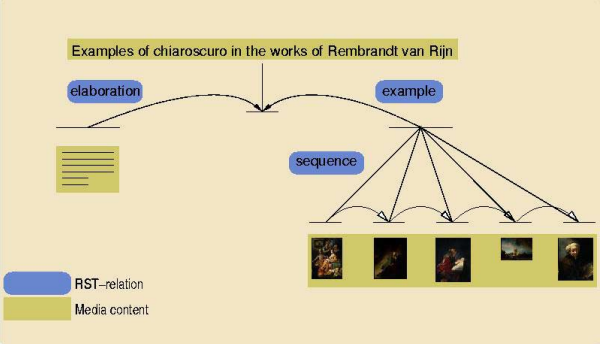

Figure 3: RST tree representation of the input.

4.1 Semantic level: rhetorical structure

A presentation is constructed around the concept "Examples of Chiaroscuro in the works of Rembrandt van Rijn", using the images as *examples* of the the core concept, and the text as an *elaboration* of the core concept. Additionally, to preserve the ordering of the time the picture was made, the five images are shown in a *sequence* relation. The resulting tree structure, using the notation common in Rhetorical Structure Theory [20] is shown in Figure 3 (the titles of the individual paintings have been omitted for brevity). The tree is encoded using a simple XML Schema to represent RST nucleus/satellite relations. The XML associated fragment is shown in Figure 4.

Note that we do *not* expect content authors to directly use the semantic abstractions nor the rhetoric markup during the authoring phase. Automatic generation of these structures, however, requires advanced knowledge of the domain, the organization of the multimedia database, the users and their task, and is the topic of future research. In the current Cuypers prototype, the generation of the rhetorical structure is simply hardwired into the server's multimedia information retrieval system, which is considered to be beyond the scope of this paper. Here, we focus on the Cuypers presentation engine, and assume the RST structure as the input given to the engine.

```
<!DOCTYPE presentation PUBLIC "-//CWI/DTD Rhetorics 1.0//EN" "rhetoric.dtd">
<presentation xmlns="http://www.cwi.nl/~media/ns/cuypers/rhetoric">
  <media id="title" ... refs to content/metadata database .../>
  <media id="img1" ... />
  <media id="img2" ... />
  <media id="img3" ... />
  <media id="chiaroscuro" ... />
  <nsRelation>
    <nucleus>
      <mediaref idref="title"/>
    </nucleus>
    <satellite type="example">
      <mnRelation type="sequence">
        <nucleus>
          <mediaref idref="img1"/>
        </nucleus>
        ... ...
        <nucleus>
          <mediaref idref="img5"/>
        </nucleus>
      </mnRelation>
    </satellite>
    <satellite type="elaboration">
      <mediaref idref="chiaroscuro"/>
    </satellite>
  </nsRelation>
</presentation>
```

Figure 4: XML encoding of the presentation's rhetorical structure.

4.2 High-level presentation semantics: communicative devices

Note that the rhetorical structure given in Figure 3 contains no information about the spatio-temporal layout of the final-form presentation. This information is incrementally added by the Cuypers system, based on general design knowledge, combined with knowledge about the underlying domain (e.g. "17th century painting"), the task and preferences of the end-user and the capabilities of the device that is used to access the Web. In the first step, the input is matched against a set of rules designed to convert the input to a communicative device hierarchy. Note that this is purely a design decision: in practice, designers of a particular application will need to extend the default rule set that comes with the Cuypers system.

In the example above, the rules that match the input RST structure could, for example, map the root nucleus (the label "Chiaroscuro by Rembrandt van Rijn") to the title of the presentation. In addition, the rules determine that the title, elaborative text and the example section should be visible at the same time, as close to another as possible. This is used by grouping these elements in a communicative device named *spatial adjacency* [23]. Because the example section itself consisted of a sequence of images of which the ordering should be preserved, the sequence is mapped to a communicative device named *bookshelf*. The bookshelf's layout strategy is parameterized, in this case the strategy is to try to achieve a left-to-right, top-to-bottom ordering first, and to use a temporal overflow strategy when it proves impossible to fit all images on a single screen. The resulting hierarchy is sketched in Figure 5.

4.3 Qualitative constraint level

While the communicative device hierarchy described above reflects the most basic design decisions about the way the

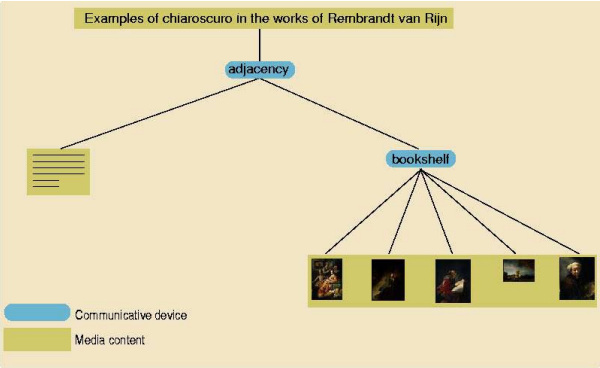

Figure 5: Example communicative device hierarchy.

document should be communicated to the user, the mutual relationships among the media items have not been established. This is done in the qualitative constraint level, which converts the communicative device hierarchy to a graph structure, for example the graph drawn below in Figure 6.

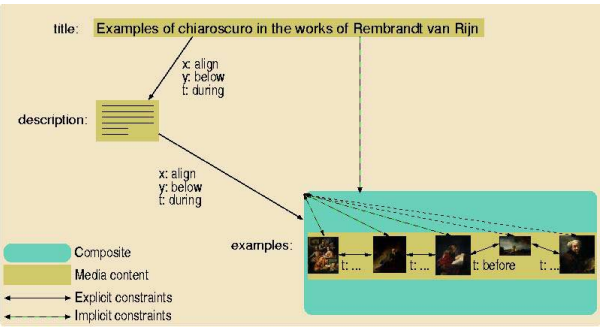

Figure 6: Example qualitative constraint graph.

The graph structure consists of nodes and edges, where the nodes represent the media items and the edges between the nodes are labeled with the constraints that relate them. *Composite* nodes can be used to model useful hierarchical relationships between media items at the constraint level.

Figure 6 represents the resulting graph after backtracking over several alternative solutions for converting the communicative device structure displayed in Figure 5. In this case, it turned out that the user's screen size is too small to display more than one painting at a time. As a result, all alternatives that tried a left-to-right, top-to-bottom ordering of the paintings failed, and the bookshelf has resorted to its overflow strategy: it decides to show the paintings one after the other, sequentially ordered in time (represented by the `before` constraints on the temporal dimension that applies to all images in the figure). During the time the images are shown, it makes sure that the title and descriptive text are also shown (represented by the `during` constraints in the figure). Also note that to define the communicative devices in terms of qualitative constraints, only a limited number of constraints need to be specified directly. Most constraints can be automatically generated by the system. For example, if the title is to be displayed during the description, and the description is to be displayed during the examples, the system automatically derives that the title is to be displayed during the examples. These automatically generated constraints are used when checking consistency rules such as "always show a title when displaying something else". In this case, the system knows that the title is shown during the examples, even when this is not explicitly specified by the transformation rules.

4.4 Quantitative constraint level

To be able to check whether a proposed multimedia layout can be realized, all the constraints need to be resolved on the lowest level. For the spatial and temporal dimensions, this means that all the qualitative constraints need to be converted into numeric or quantitative constraints. So if three images of a certain size are to be positioned left of one another with a certain minimum padding, at this point the system needs to do the associated mathematics to find out whether and how this can be done: it reformulates all the qualitative constraints into numerical constraints, fills in the actual sizes of the images and acceptable padding distances, and tries to solve the given set of constraints. The con-

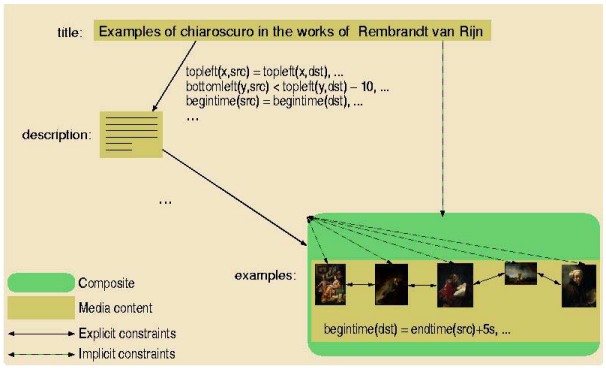

Figure 7: Example (partial) quantitative constraint graph.

version process involves many quantitative constraints that are automatically generated, and Figure 7 shows only a few of them. This is, however, very efficient from an implementation point of view: the more constraints that are added, the smaller the constraint variable domains become, and the faster a solution will be generated.

Note that part of the information generated at this step is only needed to make sure that layouts meet the given constraints. Parts of the solution itself are too low-level to be useful in high-level formats. Other parts of the solution, however, are directly used and copied almost verbatim into the encoding of the final-form presentation.

4.5 Final-form generation

In the last step, the information accumulated in the previous steps is used to generate the final presentation in SMIL. A snapshot of the result is shown in Figure 8.

The resulting SMIL markup is listed in Figure 9. As one can see, the encoding used for the layout specification in the head is rather low level, and these are indeed the direct values generated by solving the set of quantitative constraints. In contrast, the temporal hierarchy in the body has been

Figure 8: Snapshot of the resulting SMIL presentation (RealPlayer).

generated on the basis of the qualitative (Allen) constraints, realizing `during` constraints with `<par>` elements in SMIL, and `after` constraints with `<seq>` elements.

```
<smil>
  <head>
    <meta name="generator" content="Cuypers 1.0"/>
    <layout>
      <root-layout id="root-layout" background-color="black" width="400" height="690"/>
      <region id="title" left="10" top="5" width="400" height="50" fit="meet"/>
      <region id="descr" left="10" top="55" width="400" height="200" fit="meet"/>
      <region id="img" left="10" top="255" width="400" height="400" fit="meet"/>
      <region id="ptitle" left="10" top="655" width="400" height="35" fit="meet"/>
    </layout>
  </head>
  <body>
    <par>
      <text region="title" src="...query to multimedia database..."/>
      <text region="descr" src="..."/>
      <seq>
        <par dur="10"> ... 1st painting+title ... </par>
        <par dur="10"> ... 2nd painting+title ... </par>
        <par dur="10"> ... 3rd painting+title ... </par>
        <par dur="10"> ... 4th painting+title ... </par>
        <par dur="10">
          <img region="img" src="..."/>
          <text region="ptitle" src=".."/>
        </par>
      </seq>
    </par>
  </body>
</smil>
```

Figure 9: SMIL encoding of the presentation shown in Figure 8.

5. IMPLEMENTATION

To exploit the possibilities offered by on demand multimedia presentation integration, we have integrated the Cuypers core presentation generation engine with an off-the-shelf HTTP server (Apache), as depicted in Figure 10.

The server parses XML input as shown in Figure 4, using the XML parser of Apache's Xerces framework. The result is, via the DOM interface, converted by a Cocoon Java

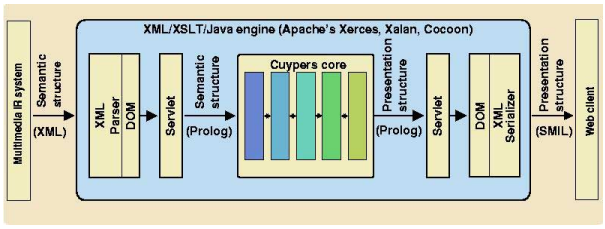

Figure 10: The core Cuypers architecture and its integration within the Apache HTTP server.

servlet to an equivalent Prolog term. This Prolog term is the actual input taken by the core of the presentation engine, which consists of a number of transformations written in ECLiPSe [29]. ECLiPSe allows the transformations to combine, within a single runtime environment, standard Prolog rule-matching and back-tracking with high-level constraint solving techniques. This allows high level transformation rules to generate alternative layouts using lower-level sets of constraints. Layouts with constraints that prove to be insolvable automatically evaluate to false and cause the system to backtrack, trying alternative layout strategies. In addition, the layout rules can exploit Prolog's unification mechanism as a powerful and extensible selector mechanism, without the need to implement a special purpose selector language such XPath [8]. When the constraints for a given layout can be solved by ECLiPSe, this solution is returned back to the servlet. The servlet converts the result back to XML (in this case SMIL), again using Cocoon's DOM interface.

The example described above focuses on the use of rhetorical structures as the main technique for describing the input, and on SMIL for describing the final-form output. The core of the Cuypers presentation engine, however, is independent of these formats. Any input that can be transformed to a set of communicative devices can be supported by plugging in a rule set that transforms the input to a set of communicative devices. The same applies to the output format, which can be modified by adapting the lower-level rules that use the (solved) constraints to generate the final form output.

The constraints we currently use for the temporal dimension are based on the temporal relations defined by Allen [1]: *equals, before, meets, overlaps, during, starts* and *finishes*, with similar relations for the spatial dimensions X and Y. For the stacking order of the media items (the "Z" dimension), we use *above* and *below* constraints. Properties of these qualitative constraints, such as symmetry (A below B $\equiv$ B above A) or transitivity (A during B *land* B during C $\rightarrow$ A during C) are described using the Constraint Handling Rules (CHR, [12]) library of ECLiPSe.

6. RELATED WORK

Generation of synchronized multimedia presentations from higher level descriptions is not novel in itself. Spatial and temporal constraints for specifying multimedia are used, for example, in the Madeus system [18]. We share our objective to realize cross-platform and cross-media adaptation with preservation of the intended semantics with the ZyX document format [4]. While we generate the entire document flow on the basis of a number of knowledge sources, the ZyX

approach is essentially based on augmenting an existing document so that it can be adapted while preserving the main information flow.

Within the AI community, a common reference architecture for model-based multimedia presentations has been developed. This Standard Reference Model for Intelligent Multimedia Presentation Systems (SRM-IMMPS) [5] is based on the synthesis of media content, while we focus on reusing existing content from an annotated multimedia repository. Other relations between SRM-IMMPS and our work are described in [24].

Our work is also closely related to the work of Elisabeth André, who described the use of AI planning techniques in combination with constraint solvers in her WIP and PPP systems [2]. The main contribution of our approach is that it integrates the several processing steps into a single runtime environment so that the system can freely backtrack across the different levels. This allows high-level presentation decisions to be re-evaluated as a result of constraints that turn out to be insolvable at the lower levels (e.g. pixel level). Nevertheless, the individual levels remain conceptually separated, which allows the definition of small, declarative design rules instead of the single hierarchy of planning operators used by André. In Cuypers, semantic relations such as the rhetoric structure are encoded as an explicit level of abstraction, whereas these are used within WIP as implicit design guidelines for the implementation of the generation plan. Additionally, Cuypers uses ECLiPSe as a commonly available, off-the-shelf logic constraint programming (CSP) tool while WIP used a dedicated planner.

7. FUTURE WORK: TOWARDS THIRD GENERATION MULTIMEDIA

Similar to third generation textual content, third generation multimedia will focus on machine-processable content. Richly annotated multimedia presentations will not only facilitate intelligent Web retrieval and brokering services, but also facilitate reuse of media content in other presentations. In the long term, when there is a sufficient amount of annotated multimedia available, systems such as Cuypers would be able to operate without the multimedia database depicted in Figure 1, and, instead, operate directly on multimedia content found on the Web.

Note that W3C document formats such as SMIL and SVG already anticipate this by allowing documents to contain embedded annotations. In addition, ISO's MPEG-4 [17] also allows embedded annotations. It remains unclear, however, which annotation languages are the most appropriate, and we are currently investigating various alternatives for the encoding of our metadata. We are investigating not only the use of RDF-based languages such as RDF Schema and DAML+OIL [26], but also approaches that build directly on top of XML Schema, such as the description schemes developed by the MPEG-7 community [16].

Adequately annotated multimedia is a key pre-requisite for this multimedia variant of the Semantic Web. Unfortunately, current multimedia authoring tools provide little support for producing annotated multimedia presentations. Much of the underlying semantics of the overall multimedia presentation and the media fragments it contains remains implicit and is only present in the head of the author. In contrast, in the Cuypers system discussed above, it is relatively easy to generate such annotations automatically. Since the entire presentation-generation process in the Cuypers system is based on explicitly encoded knowledge, this knowledge can be preserved and encoded as rich metadata annotations in the final-form presentation. Note that such metadata annotations can arise from different knowledge sources and describe different abstraction levels. For example, when the system is used to generate richly annotated SMIL, the metadata section of the SMIL document may contain metadata about the individual media items (as retrieved from the underlying media database), the rhetorical structure of the overall presentation, domain-specific knowledge of the application, etc. It could also generate a report of the design rules and user profiles that were used to justify the chosen end-result (e.g. the machine-readable equivalent of "this presentation contains much hi-end video because it is generated for users with a broadband network environment"). This could, for instance, be used by the browsers to help with automatic detection of errors in the settings of the end-user's profile.

While our future research will focus on generating richly annotated multimedia presentations, we are also looking into extending the Cuypers engine to generate other presentation types, including SVG and VRML. In addition, we are currently working on interfacing the engine with the Mirror [9] multimedia information retrieval system. In particular, we are working on improving the automatic generation of the semantic structures (such as the rhetorical structure used in the example). This generation process should not only take into account a semantic model of the application domain, but also some form of discourse model to provide guidelines on how to convey subjects from that domain to the user.

The current implementation already uses a declarative encoding of the design, user and platform knowledge. These different types of knowledge are, however, still intertwined. This part of the system needs to be redesigned to be able to manipulate the different types of knowledge through interfaces that are tailored to the different tasks and roles of the users that will need to control them, and to be able to encode the required knowledge in a declarative and reusable way. We expect the Semantic Web to play a key role in achieving these goals.

Acknowledgements

Part of the research described here has been funded by the European ITEA/RTIPA and the Dutch Dynamo and ToKeN2000 projects.

Examples are taken from a ToKeN2000 demonstrator, and all media content has been kindly provided by the Rijksmuseum in Amsterdam. Our CWI colleagues Steven Pemberton and Krzysztof Apt provided many valuable insights.

8. REFERENCES

[1] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–844, November 1983.

[2] E. André. WIP and PPP: A Comparison of two Multimedia Presentation Systems in Terms of the Standard Reference Model. *Computer Standards & Interfaces*, 18(6-7):555–564, December 1997.

[3] T. Berners-Lee. *Weaving the Web*. Orion Business, 1999.

[4] S. Boll, W. Klas, and J. Wandel. A Cross-Media Adaptation Strategy for Multimedia Presentations. In *ACM Multimedia '99 Proceedings*, pages 37–46, Orlando, Florida, October 30 - November 5, 1999. Addison Wesley Longman.

[5] M. Bordegoni, G. Faconti, M. Maybury, T. Rist, S. Ruggieri, P. Trahanias, and M. Wilson. A Standard Reference Model for Intelligent Multimedia Presentation Systems. *Computer Standards & Interfaces*, 18(6-7):477–496, December 1997.

[6] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 Specification, February 10, 1998. W3C Recommendations are available at http://www.w3.org/TR.

[7] J. Clark. XSL Transformations (XSLT) Version 1.0. W3C Recommendations are available at http://www.w3.org/TR/, 16 November 1999.

[8] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. W3C Recommendations are available at http://www.w3.org/TR/, 16 November 1999.

[9] A. P. de Vries. *Content and Multimedia Database Management Systems*. PhD thesis, Univerity of Twente, 1999.

[10] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge Representation on the Web. In F. Baader, editor, *International Workshop on Description Logics (DL'00)*, 2000.

[11] J. Ferraiolo. Scalable Vector Graphics (SVG) 1.0 Specification. W3C Candidate Recommendations are available at http://www.w3.org/TR, 2 November 2000.

[12] T. Frühwirth. Theory and Practice of Constraint Handling Rules. *Journal of Logic Programming*, 37(1-3), October 1998. Special Issue on Constraint Logic Programming, P. Stuckey and K. Marriot, Eds.

[13] L. Hardman and J. van Ossenbruggen. Device Independent Multimedia Authoring. In *W3C Workshop on Web Device Independent Authoring*, Bristol, UK, October 3-4, 2000.

[14] International Organization for Standardization. Information Processing — Text and Office Information Systems — Standard Generalized Markup Language (SGML), 1986. International Standard ISO 8879:1986.

[15] International Organization for Standardization/International Electrotechnical Commission. Information technology — Processing languages — Document Style Semantics and Specification Language (DSSSL), 1996. International Standard ISO/IEC 10179:1996.

[16] International Organization for Standardization/International Electrotechnical Commission. MPEG-7: Context and Objectives, 1998. Work in progress.

[17] International Organization for Standardization/International Electrotechnical Commission. Information technology – Coding of moving pictures and audio, 1999. International Standard ISO/IEC 14496:1999 (MPEG-4).

[18] M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismaïl, and L. Tardif. Madeus, an Authoring Environment for Interactive Multimedia Documents. In *Proceedings of ACM Multimedia '98*, Bristol, UK, 1998.

[19] D. E. Knuth. *TeX: The Program*, volume B of *Computers and Typesetting*. Addison-Wesley Publishing Company, 1986.

[20] W. C. Mann, C. M. I. M. Matthiesen, and S. A. Thompson. Rhetorical Structure Theory and Text Analysis. Technical Report ISI/RR-89-242, Information Sciences Institute, University of Southern California, November 1989.

[21] G. Rossi, D. Schwabe, and A. Garrido. Design Reuse in Hypermedia Applications Development. In *The Proceedings of the Eighth ACM Conference on Hypertext and Hypermedia*, pages 57–66, Southampton, UK, April 1997. ACM, ACM Press.

[22] L. Rutledge, B. Bailey, J. van Ossenbruggen, L. Hardman, and J. Geurts. Generating Presentation Constraints from Rhetorical Structure. In *Proceedings of the 11th ACM conference on Hypertext and Hypermedia*, pages 19–28, San Antonio, Texas, USA, May 30 – June 3, 2000. ACM.

[23] L. Rutledge, J. Davis, J. van Ossenbruggen, and L. Hardman. Inter-dimensional Hypermedia Communicative Devices for Rhetorical Structure. In *Proceedings of International Conference on Multimedia Modeling 2000 (MMM00)*, pages 89–105, Nagano, Japan, November 13-15, 2000.

[24] L. Rutledge, L. Hardman, J. van Ossenbruggen, and D. C. Bulterman. Implementing Adaptability in the Standard Reference Model for Intelligent Multimedia Presentation Systems. In *The International Conference on Multimedia Modeling*, pages 12–20, 12-15 October 1998.

[25] L. Rutledge, J. van Ossenbruggen, L. Hardman, and D. C. Bulterman. A Framework for Generating Adaptable Hypermedia Documents. In *Proceedings of ACM Multimedia*, pages 121–130, Seattle, Washington, November 1997. ACM Press.

[26] F. van Harmelen and I. Horrocks. Reference description of the DAML+OIL ontology markup language. http://www.daml.org/2000/12/reference.html. Contributors: Tim Berners-Lee, Dan Brickley, Dan Connolly, Mike Dean, Stefan Decker, Pat Hayes, Jeff Heflin, Jim Hendler, Deb McGuinness, Lynn Andrea Stein.

[27] W3C. Synchronized Multimedia Integration Language (SMIL) 2.0 Specification. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, 21 September 2000. Edited by Aaron Cohen.

[28] W3C. XHTML 1.0: The Extensible HyperText Markup Language: A Reformulation of HTML 4.0 in XML 1.0. W3C Recommendations are available at http://www.w3.org/TR/, January 26, 2000.

[29] M. Wallace, S. Novello, and J. Schimpf. ECLiPSe: A Platform for Constraint Logic Programming, 1997.