A Representation of Media for Multimedia Authoring and Browsing Systems

Ethan V. Munson and Mark Pfeiffer
Dept. of Electrical Engineering & Computer Science
University of Wisconsin-Milwaukee
{munson,pfeiffer}@cs.uwm.edu

Abstract

This paper describes the TDO model of media, which describes media as they are viewing by multimedia authors and the authoring software they use. This model is the basis of the MSPEC medium specification language, which is used to configure a style sheet system for multimedia authoring systems. The TDO model is compared to two other models drawn from multimedia and intelligent systems research and the possibility of creating a general model of media is discussed. An appendix describes recent extensions to the TDO model.

Introduction

Systems that handle multiple media often must adapt their behavior to the characteristics of the different media that they support. This is true for systems that distribute multimedia data over networks (Mayer-Patel & Rowe 1997), systems that use artificial intelligence techniques to *automatically* produce multimedia presentations (Maybury 1993) and systems that help *people* produce multimedia presentations (Graham, Harrison, & Munson 1992; Weitzman & Wittenburg 1996; Quint, Roisin, & Vatton 1995; Layaida & Sabry-Ismail 1996). After all, media such as text, graphics, and video differ considerably in important qualities such as data volume, ability to express certain concepts, and the ways that their presentation can be controlled.

Since adaptation to different media is such an important quality of a multimedia system, it might seem reasonable to expect systems to perform this adaptation from formal, declarative descriptions of the media they support. In general, this is not the case. Most current systems either hard-code details about different media and the ways they are used together (e.g. the Madeus multimedia authoring system (Layaida & Sabry-Ismail 1996)) or treat a medium as a black box data structure which has very few exposed characteristics other than a description of its size and transmission requirements (e.g. the Continuous Multimedia Toolkit (Mayer-Patel & Rowe 1997)).

We are interested the use of explicit, declarative models of media to control the behavior of multimedia authoring systems. Our interest in this topic began with the development of a style sheet[1] system, called Proteus (Graham, Harrison, & Munson 1992; Munson 1994), which could be adapted to serve different media through declarative media descriptions. Initially, these descriptions were made in the form of C++ objects, but they are now written in a small specification language called MSPEC (Munson 1997). Using this approach, Proteus has been adapted to work with independent text, graphics, and video formatters in the Ensemble system (Graham 1992) and with Multiple Presentation Mosaic (Marden & Munson 1997), an experimental WWW browser that supports style sheet control of HTML document appearance and multiple simultaneous views of HTML documents.

In the next section, we present the TDO model of media that is currently used in Proteus. In the next section, we compare this model to two other models of media. The final section discusses directions for research on models of media. An appendix describes more recent work extending the TDO model.

The TDO Model of Media

The TDO model of media is designed to meet the needs of an authoring system. Authoring systems provide their users with a collection of object types that can be created, modified, composed, laid out and formatted for display or playback. The authoring-oriented view of a medium is almost certainly different from what would be appropriate for other services, such as intelligent systems or multimedia networking tools.

The TDO model is summarized by the following definition.

Definition 1 *A* medium *is a triple* $M = (T, D, O)$, *where* T *is a set of primitive data types,* D *is a set of dimensions in which layout is performed, and* O *is a set of formatting operations with typed parameters.*

[1] A *style sheet* is a specification of how a document should look. Style sheets vary from system to system, but they generally control the layout of the objects that make up a document as well as the details of their appearance (e.g. color and font). Examples of style sheets include LaTeX style files, Microsoft Word templates, and Cascading Style Sheets for WWW documents.

```
medium Text;

c++-definitions {
#include "AlignType.h";
}

primitives {
  Text (string) :
    creator(createTextGenNode, setTextGenNode);
}

dimensions {
  horizontal(width, left, right, hmiddle, horizpos);
  vertical(height, top, bottom, vmiddle, vertpos);
}

types {
  enum AlignType {
    leftjustify = LEFT,
    centerjustify = CENTER,
    rightjustify = RIGHT,
    blockjustify = JUSTIFY };
}

attributes {
  visible(true) : boolean;
  fontFamily("times-roman") : string;
  size(12.0) : real;
  bold(false) : boolean;
  italic(false) : boolean;
  indent(0.0) : real;
  justify(LEFT) : AlignType;
  lineSpacing(1.0) : real;
  fgColor("black") : string;
  bgColor("white") : string;
  hyphenate(false) : boolean;
  minHyph(5.0) : real;
  minLeft(2.0) : real;
  minRight(2.0) : real;
}

functions {
  uroman(real) : string calls upperRoman;
  lroman(real) : string calls lowerRoman;
  arabic(real) : string calls arabicStr;
}
```

Figure 1: The medium specification which configures Proteus for use with Ensemble's text medium.

The three sets that compose a medium deserve further elaboration.

Primitive Types: Every medium supports a set of data types that are the raw material for documents that can be displayed in that medium. Examples of data types include text, two-dimensional graphical shapes, video clips, and audio clips.

Dimensions: In every medium, the objects that make up documents are laid out in a coordinate space having one or more dimensions. Media differ in both the number and types of dimensions they use. A digital audio medium will have only a time dimension, while a three-dimensional animation medium will have a total of four dimensions (three spatial dimensions and time).

Operations: The final feature that distinguishes media in the TDO model is the formatting operations that they support. A formatting operation takes data drawn from the primitive types and converts it into a form that can be displayed or played back. Most formatting operations are controlled by style parameters such as font (for text media), stroke width (for graphics media), or frame rate (for video media). Examples of formatting operations include filling and stroking (for graphics), and linebreaking and pagination (for text).

The MSPEC language

The TDO model of media is the basis for MSPEC, a language for describing media that is used to configure how the Proteus style sheet system works with applications that support different media.[2] An MSPEC file is translated into C++ code that implements a custom interface to Proteus that its application client will use to access style information specified in style sheet files.

Figure 1 shows an example MSPEC file that was used to describe the text medium of the Ensemble system. About half of the material in this file corresponds directly to the TDO model, while the remaining material is required for building a valid C++ interface to Proteus. The parts of the file that correspond to the TDO model are:

- The `primitives` section is used to define the types that the medium supports. This medium has only one type, `Text`. Elements of this type can be created by providing a single argument of type `string`.
- The `dimensions` section defines the layout dimensions of the medium. Each dimension's line specifies the name of the dimension and the names that are to specify the extent (first name in parentheses) and position (remaining four names) of objects in that dimension.
- The `attributes` section defines the names and types of the parameters used by the medium's formatting operations. For example, `italic` is a boolean parameter used when linebreaking paragraphs. The `types` section is used to define types for attributes other than the built-in types (`real`, `boolean`, `string`).

MSPEC is not necessarily a "perfect" specification of a medium — it is simply a practical application of the

[2] A complete description of the MSPEC language is beyond the scope of this paper, but can be found in an earlier paper on the TDO model (Munson 1997).

TDO model, which otherwise might seem abstract and difficult to exploit.

Other Models of Media

The Data Type Model

In multimedia research, there exists an implicit model of a medium as a data type. This model is rarely discussed explicitly, though Buford's definition of multimedia hints at it. He says

> It is the simultaneous use of data in different media forms (voice, video, text, animations, etc.) that is called multimedia. (Buford 1994, page 2)

This model works well for applications that deal with documents in their final form, such as programs which transmit video, audio, and other media over networks. The final forms of different media are fairly distinct, and for transmission and play-back applications like the Continuous Media Toolkit (CMT) (Mayer-Patel & Rowe 1997), they can generally be summarized by a small set of characteristics, like data volume and the level of real-time guarantees that they require.

The data type model does not do a good job of serving applications which must compose presentations of documents. Even documents that are nominally "single medium" are often constructed from objects of diverse type. For example, a television news program may be created using combinations of text labels, static graphic charts, static images scanned from photographs, computer animations, recorded audio, recorded video, and live video. The result of combining these seven different data types will be a stream of video data, i.e. data from a single medium. So, under the data type model, is this one medium or a combination of seven? The answer is not clear.

The TDO model is more general and powerful than the data type model for three reasons:

1. It does not assume that a medium supports only a single data type. Under the TDO model, a medium may support many types and the types supported by different media do not have to be disjoint. For example, an author creating a video document may use text for titles and labels. Thus, a video authoring tool must support both the video clip and text data types.
2. The TDO model recognizes that media with identical types can differ in their dimensions. For example, 3D animation uses the same types of graphical objects as static 3D graphics, but adds the ability to lay out those objects in the time dimension.
3. Most importantly, the data type model ignores the formatting operations that are at the heart of any authoring tool. A text-authoring application must support linebreaking and pagination in their full complexity. In contrast, a video-authoring application may support text for labels but may require that the user linebreak longer sections of text manually. This difference between what each medium *does* is just as, if not more, important than the types of objects they operate on.

The AHV Model

Another model of media was developed by Arens, Hovy, and Vosser (Arens, Hovy, & Vossers 1993). This model, which we call the AHV model, was designed to meet the needs of intelligent systems that must decide how to present information to human users. In the AHV model, the central object is an *exhibit*, which is a collection of simple exhibits. A *simple exhibit* uses a single medium to display an *information carrier* on a *substrate*. In document processing terms, an exhibit is equivalent to a document and a simple exhibit is a document element. Information carriers are foreground objects and the substrate is the background. Information carriers convey information along *channels*, which are qualities of the carrier such as font choice for text carriers or label type (text or number) for picture carriers.

In the AHV model, a medium has seven characteristics:

Carrier Dimension: The number of dimensions required to present information to the user.

Internal Semantic Dimension: The number of semantic dimensions in an information carrier.

Temporal Endurance: Whether the appearance of information can change while it is being presented (e.g. in visibility or position).

Granularity: Whether dimensions are discrete or continuous.

Medium Type: The kind of sensory output produced (aural or visual).

Default Detectability: A measure of intrusiveness to the user.

Baggage: A gross measure of the effort required to comprehend material in the medium.

When a system using this model needs to decide how to display information, it compares these qualities of media with related qualities of the information to be displayed: dimensionality, transience, urgency, order, density, naming, and volume. The system finds the best match for the information among the available media and uses that best match to display the information.

It is difficult to make a direct comparison between the TDO and AHV models of media, primarily because they approach media from very different viewpoints. Both models emphasize the importance of dimensions and have a mechanism for representing style parameters. In general, the AHV model takes a higher-level viewpoint and places an emphasis on human perception of media. For instance, the AHV model does not treat the temporal dimension as a first-class dimension probably because we perceive temporal and spatial layout effects differently, even though layout can be specified similarly for both types of dimensions. The AHV model does not make explicit mention of data types

and appears to assume that a medium has only a single data type. Channels in the AHV model appear to mix appearance concepts with semantic concepts, since the value of the "label" channel for a picture can be either "number" or "text," which most document systems would treat as semantically distinct.

Discussion and Conclusions

Looking at the three media models presented here, it should be clear that the nature of existing models of media is determined by how they will be used. Each model supports its native domain well, but is not adequate for other domains.

- The data type model is well-suited to the networking and distributed-system applications that are so important in current multimedia research. These applications are primarily used for transmitting documents in their final form, which is often a uniform data type.
- The TDO model is designed expressly for supporting authoring systems and focuses on the technical issues in converting an abstract representation of the document, which often contains many data types, into a form that can be displayed or played back. The model emphasizes the mechanics of producing output from the stored form of the document, rather than describing how information presented in a medium impacts the people perceiving it.
- The AHV model focuses on how media can be used to convey chunks of information, which is a critical issue for intelligent systems that must decide which medium to use to convey an idea to a person.

Still, these models have substantial overlap. Both the data type model and the TDO model recognize the importance of data types to media. The AHV and TDO models have many more similarities, including their attention to dimensions and the parameters that control formatting.

We believe that it is possible to create a general theory of media that covers all these domains. As a starting point, we are extending and refining the TDO model (see the appendix of this paper). One extension makes explicit how the final form of a document is produced. We are also very interested in exploring how the TDO model can be combined with models like the AHV model that emphasize perceptual aspects of media.

Appendix

The TDAO Model

The TDO model is not sufficiently rigorous to be called a theory. Its central problem is a lack of precision in the description of its three sets, particularly the set of operations. It relies on our intuition about authoring tools and media rather than mathematical rigor. This may not be a serious problem in the case of the primitive types and the dimensions, which correspond closely to well-understood notions from computer science (types) and mathematics (dimensions). But the formatting operations are clearly functions and are not defined with sufficient rigor for us to understand how they work. For example, the model does not describe their domains and ranges and does not state whether they can be composed.

So, we propose an improved model of media, which we call the TDAO model.

First, we add a fourth set to the model, A, the set of attributes, representing the style parameters that control the formatting operations.

Definition 2 *A* medium *is a four-tuple*

$$M = (T, D, O, A),$$

where T is a set of types, D is a set of dimensions in which layout is performed, O is a set of formatting operations, and A is a set of attributes.

Types Next, we expand our definition of T, the set of types, to include not only the primitive media data types which are the initial input to the formatting process, but also the types used internally in the formatting process and the types of the attributes.

Definition 3 *The set of types, T, is formed by the union of three subsets: T_P, the primitive media data types; T_O, the types of data produced by the formatting operations; and T_A, the types of the attributes in A. These three sets are not necessarily disjoint.*

$$T = T_P \cup T_O \cup T_A$$

The individual types are simply sets of values.

Definition 4 *A type, $t_i \in T$, is a set of values. Each t_i may be finite (e.g. booleans) or infinite (e.g. ASCII strings), atomic (e.g. integers) or composite (e.g. two-dimensional splines).*

Dimensions Our definition of the set of dimensions has been modified slightly by the addition of discrete dimensions and a clear statement that the dimensions of a medium form a coordinate space.

Definition 5 *The set of dimensions for a medium, D, has k members, whose Cartesian product is the coordinate space in which material is laid out.*

$$d_1 \times d_2 \times \ldots \times d_k$$

Each dimension, d_i, may be continuous or discrete, bounded or unbounded.

Attributes The elements of the set of attributes, $a_i \in A$, represent the style parameters that control the medium's formatting process. Each attribute has a type, $t \in T_A$, which specifies the set of values that the attribute may hold. Examples of possible attributes include stroke width (for 2D graphics), font size (for any medium supporting text), transition style (used with video and having values like "cut," "fade," and "dissolve").

Operations The most complex aspect of the TDAO model is the formatting operations that compose the set O.

A formatting operation is a function that takes some kind of input along with a corresponding set of attribute values and produces output. In some cases, this output can be displayed or played back directly, but in others, it will serve as the input to another formatting operation. Examples of formatting operations include filling and stroking (for 2D graphics), texture mapping (for 3D graphics), linebreaking and pagination (for text).

Let us consider, as an example, a hypothetical fill operation for a 2D graphics medium that supports color gradients. Its input data would be a path (a list of connected splines). The attributes controlling the operation might be *color1*, *color2*, and *angle*. If *color1* and *color2* are the same, then the path is filled as a solid. Otherwise, the path is filled with a gradient that changes smoothly from *color1* to *color2* in the direction defined by *angle*. The output of the operation is a filled path.

We might describe this operation more formally like this:

$$\text{Fill} : \text{path} \times \text{color1} \times \text{color2} \times \text{angle} \longrightarrow \text{filled path}$$

We can generalize this specification to produce a definition for the set of formatting operations.

Definition 6 *A medium's set of operations, O, contains k members where $k \geq 1$. Each operation, $o_i \in O$ is a tuple*

$$\begin{array}{lcl} o_i & = & (t_i^{in}, t_i^{out}, A_i, f_i), \text{where} \\ & t_i^{in} & \text{is the input data type,} \\ & t_i^{out} & \text{is the output data type,} \\ & A_i \subset A & \text{is the set of } m \text{ attributes relevant to the} \\ & & \text{operation, where } A_i = \{a_{i1}, \ldots, a_{im}\}, \\ & f & \text{is the function performed by the} \\ & & \text{operation, where} \\ & & f : t_i^{in} \times a_{i1} \times \ldots \times a_{im} \longrightarrow t_i^{out} \end{array}$$

Consider two operations, o_i and o_j, where

$$\begin{array}{lcl} o_i & = & (t_i^{in}, t_i^{out}, A_i, f_i), \textit{and} \\ o_j & = & (t_j^{in}, t_j^{out}, A_j, f_j). \end{array}$$

We can say that these two operations are composed by stating that the output of one is the same as input of the second,

$$t_i^{out} = t_j^{in}.$$

We can describe a medium that applies the same operation in a iterative manner by specifying that two operations have the same function,

$$f_i = f_j$$

Future Work We are continuing to refine the TDAO model. Areas still needing work include the specification of operations that have multiple input types and a more complete specification for operation composition.

Once the TDAO model is more mature, we expect to modify the MSPEC language to conform with it. The primary change we expect to make is to add a new section specifying operations and their input/output behavior explicitly. This should allow Proteus to be modified to play a more active role in deciding when parts of a document need to be reformatted and to provide even better incremental performance.

References

Arens, Y.; Hovy, E. H.; and Vossers, M. 1993. On the knowledge underlying multimedia presentations. In Maybury, M. T., ed., *Intelligent Multimedia Interfaces*. AAAI Press/MIT Press. chapter 12.

Buford, J. F. K. 1994. *Multimedia Systems*. SIGGRAPH Books. New York: ACM Press.

Graham, S. L.; Harrison, M. A.; and Munson, E. V. 1992. The Proteus presentation system. In *Proceedings of the ACM SIGSOFT Fifth Symposium on Software Development Environments*, 130–138. Tyson's Corner, VA: ACM Press.

Graham, S. L. 1992. Language and document support in software development environments. In *Proceedings of the Darpa '92 Software Technology Conference*.

Layaida, N., and Sabry-Ismail, L. 1996. *Maintaining Temporal Consistency of Multimedia Documents Using Constraint Networks*. Multimedia Computing and Networking. SPIE. 124–135.

Marden, Jr., P. M., and Munson, E. V. 1997. Multiple Presentations of WWW Documents Using Style Sheets. In Ebert, D., and Nicholas, C., eds., *Proceedings of NPIV '97, the Workshop on New Paradigms in Information Visualization and Management*. ACM Press.

Maybury, M. T., ed. 1993. *Intelligent Multimedia Interfaces*. AAAI Press/MIT Press.

Mayer-Patel, K., and Rowe, L. 1997. Design and performance of the berkeley continuous media toolkit. In Freeman, M.; Jardetzky, P.; and Vin, H. M., eds., *Multimedia Computing and Networking*, 194–206. SPIE.

Munson, E. V. 1994. *Proteus: An Adaptable Presentation System for a Software Development and Multimedia Document Environment*. PhD dissertation, University of California, Berkeley. Also available as UC Berkeley Computer Science Technical Report UCB/CSD-94-833.

Munson, E. V. 1997. Toward an operational theory of media. In Nicholas, C., and Wood, D., eds., *Principles of Document Processing*, number 1293 in Lecture Notes in Computer Science. Palo Alto, CA: Springer-Verlag.

Quint, V.; Roisin, C.; and Vatton, I. 1995. A structured authoring environment for the world-wide web. *Computer Networks and ISDN Systems* 27(6):831–840.

Weitzman, L., and Wittenburg, K. 1996. Grammar-based articulation for multimedia document design. *Multimedia Systems* 4(3):99–111.