A Model-Driven Approach to Content Repurposing

Zeljko Obrenovic and Dusan Starcevic
University of Belgrade

Bran Selic
IBM Rational Software

The Multimedia Metamodel defines platform-independent multimedia concepts, opening the way for novel approaches to designing content repurposing solutions. Designers can use the metamodel to create content and add metadata to existing content, simplifying content analysis and repurposing.

Internet-enabled devices, such as cell phones, PDAs, desktops, laptops, and wearable PCs, have quite different requirements and presentation capabilities. Nonetheless, online content developers typically work independent of devices, and often the resulting content is unsuitable to them. To enable efficient content access through various pervasive-access devices, we must rethink how we create and specify content.

Content repurposing adjusts existing content to enable its reuse for various device profiles and usage scenarios.[1] However, meeting content repurposing's full potential is a challenging task. There are hundreds of device profiles available for accessing online content; the profusion of formats and standards for such content—and a lack of consensus on how to unify them[2]—makes the situation even worse.

Other domains have faced such problems, and it's useful to analyze their experiences and methods. In the software development community, for example, developers exchange data using various tools from different vendors. In this case, the model-driven approach lets developers efficiently exchange and transform different data and metadata structures (see http://www.omg.org/mda/index.htm). Can we effectively borrow from such solutions?

Here, we present a model-driven framework for content repurposing. The essence of our proposed solution is the Multimedia Metamodel, which introduces concepts and mechanisms from the multimedia domain. We've used this metamodel to explore novel approaches to designing content repurposing solutions.

Existing solutions

Many existing multimedia authoring tools offer limited content repurposing by transforming their native format to and from other formats. However, these export/import content bridges are unsuitable for the highly dynamic online world. Developing content bridges is a difficult and costly process, as bridges require detailed knowledge of proprietary formats and interfaces. To allow broader content repurposing among various platforms, we need up to N^2-N bridges, where N is the number of target platforms.[3] Furthermore, a particular bridge's processing logic is not necessarily reusable in constructing other bridges, which greatly increases the development costs.

Web standards such as the Hypertext Markup Language (HTML), the Extensible Markup Language (XML), and Cascading Style Sheets (CSS) can help content providers meet multipurpose Web publishing challenges. The World Wide Web Consortium defined these document markup and style-sheet languages to facilitate Web device independence, content reuse, and network-friendly encoding.[4] Structured documents with style sheets let developers present the same document on various Web devices. Because HTML, XML, and CSS are declarative data formats, they're easily converted to other formats. They're also more likely to be device-independent and tend to live longer than programs. However, such standards cover only a subset of multimedia content space and usage scenarios.

Multimedia researchers have attempted to separate knowledge content from presentation. To this end, Ashwin Ram and his colleagues used the Procedural Markup Language (PML).[5] PML lets developers specify knowledge structures, the underlying physical media, and the relationships between them using cognitive media roles. Developers can translate the PML specification into various presentations depending on the context, goals, and user expertise. However, such solutions do not directly address the problem of repurposing existing content. Edward Posnak,

1070-986X/04/$20.00 © 2004 IEEE Published by the IEEE Computer Society

Table 1. Mapping the Multimedia Metamodel to the OMG's MDA levels.

OMG MDA Level	Multimedia Metamodeling Architecture	Description
M3: *Meta-metamodel*	The Meta Object Facilities (MOF)	*The MOF is an OMG standard that defines a common, abstract language for metamodel specification. MOF is a meta-metamodel—the model of the metamodel (sometimes called an ontology).*
M2: *Metamodels*	The Multimedia Metamodel	*The Multimedia Metamodel provides a common and standardized language for sharing and reusing knowledge about phenomena from domains relevant to the design of multimedia solutions. It's a "metamodel" because it's the abstraction (model) of platform-specific models.*
M1: *Models*	Platform-specific schemas (such as XHTML, Wireless Markup Language, and SMIL schemas)	*Platform-specific models of multimedia content.*
M0: *Objects, data*	Content data (such as WML, XHTML, and SMIL files)	*Instances of platform-specific models.*

Greg Lavender, and Harrick Vin proposed a framework that simplifies multimedia software component development by facilitating reuse of code, design patterns, and domain expertise.[6] Their solution lets components dynamically adapt presentation quality to the available resources in heterogeneous environments. This framework is primarily aimed at decoding and processing digital audio and video data. Piero Fraternali proposed a methodology and tools for conceptual development of online Web applications. Developers can use his abstract notation to specify structure, navigation, and presentation semantics.[7] However, this approach is more suitable for conventional Web access and data-oriented Web applications, and is not directly applicable to online access for pervasive devices.

A model-driven approach

Our work is inspired by the Object Management Group's (OMG) Model-Driven Architecture (MDA), which facilitates system specification and interoperability using hierarchically organized formal models.[8] Specifically, MDA uses a platform-independent base model (PIM), and one or more platform-specific models (PSM), each describing how the base model is implemented on a different platform.[9] The PIM is thus unaffected by the specifics of different implementation technologies, and it's not necessary to model an application or content for each new technology or presentation format.

Generic framework

Our work focuses on two areas:

- Designing the unified Multimedia Metamodel, which synthesizes knowledge and common concepts from different multimedia domains into a single uniform view.
- Applying the Multimedia Metamodel to content repurposing problems.

In terms of the MDA architecture, the Multimedia Metamodel falls within the metamodel layer (see Table 1). The Multimedia Metamodel's role in our overall framework is similar to that played by the Common Warehouse Metamodel (CWM) in the business data integration domain.[10] Multimedia needs a similar generic framework, because of the similarities and overlap between various multimedia content standards, such as MPEG-4, Virtual Reality Modeling Language (VRML), and Synchronized Multimedia Integration Language (SMIL).[11]

The Multimedia Metamodel offers a conceptual basis for various content repurposing solutions. It lets us construct uniform, high-level views on various existing platforms that we can use to repurpose or integrate platform-specific content. Using the metamodel's concepts, we can also create new content in platform-independent terms or add general metadata to existing content, which can simplify content analysis and aid repurposing.

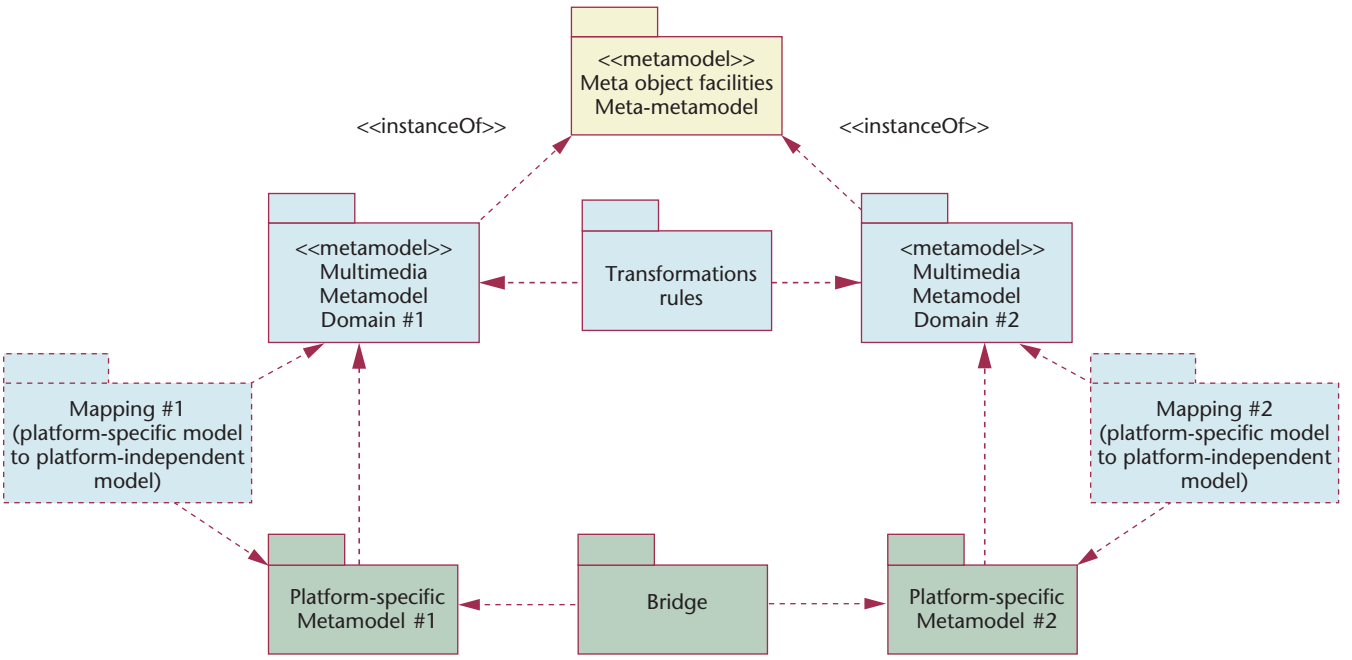


Figure 1. Models and transformations. The first platform-specific model might be an XHTML schema, for example, while the second model might be a speech synthesis platform, such as a Java Speech API.

Model transformations are central to the MDA approach; we therefore use it as a basis for model-driven content repurposing. In MDA, platform-independent models are initially expressed in a platform-independent modeling language, and later translated to platform-specific models by using formal rules to map the PIMs to an implementation platform (see Figure 1). Although bridge transformations work directly with platform-specific metamodels (schemas), in MDA, more general approaches offer the greatest possibilities. Developers can specify content model transformation using a set of rules defined in terms of the corresponding higher-level metamodels. The transformation engine itself can be built on any suitable technology, such as Extensible Style Sheet Language Transformation tools.

Building model-driven transformation tools is much more efficient than bridge building because the number of modules is linearly dependent on the number of target platforms. With a model-driven approach for each new platform, we develop only two new modules that map the new format into the platform-independent form and vice versa. With only two or three formats, bridges are efficient for communicating. However, given the need for up to N^2-N bridges, five formats increase the bridges required to 20, while 10 formats require 90 bridges and 20 formats require 380 bridges. In contrast, for 5, 10, and 20 formats, the number of model-driven transformations required are 10, 20, and 40, respectively.

Benefits

Although conceptual design of multimedia content isn't new, a model-driven approach brings many advantages. Our approach is based on standard technologies such as the Unified Modeling Language (UML) and XML, which are familiar to many software practitioners and are well supported by tools.[12] Therefore, it's not necessary to develop complex solutions from scratch, and developers can reuse existing model-driven solutions and experiences from other domains. We use existing UML modeling tools, XML parsers, and software frameworks, developing only code that extends, customizes, and connects those components according to common and standardized language defined in the Multimedia Metamodel.

The metamodel concept is strongly related to the ontology concept from knowledge management communities.[13] From one viewpoint, the Multimedia Metamodel is an ontology that provides a uniform view on important relations among multimedia concepts. Unifying these different views can give us a fresh view that might raise new ideas and approaches. The unified ontology of multimedia concepts can provide a context in which we might perceive many subtle relations. The Multimedia Metamodel could thus be used as a basis for collaborative development of knowledge about multimedia phenomena.[14] Finally, using metamodels and ontology concepts also makes it possible for us to use other knowledge

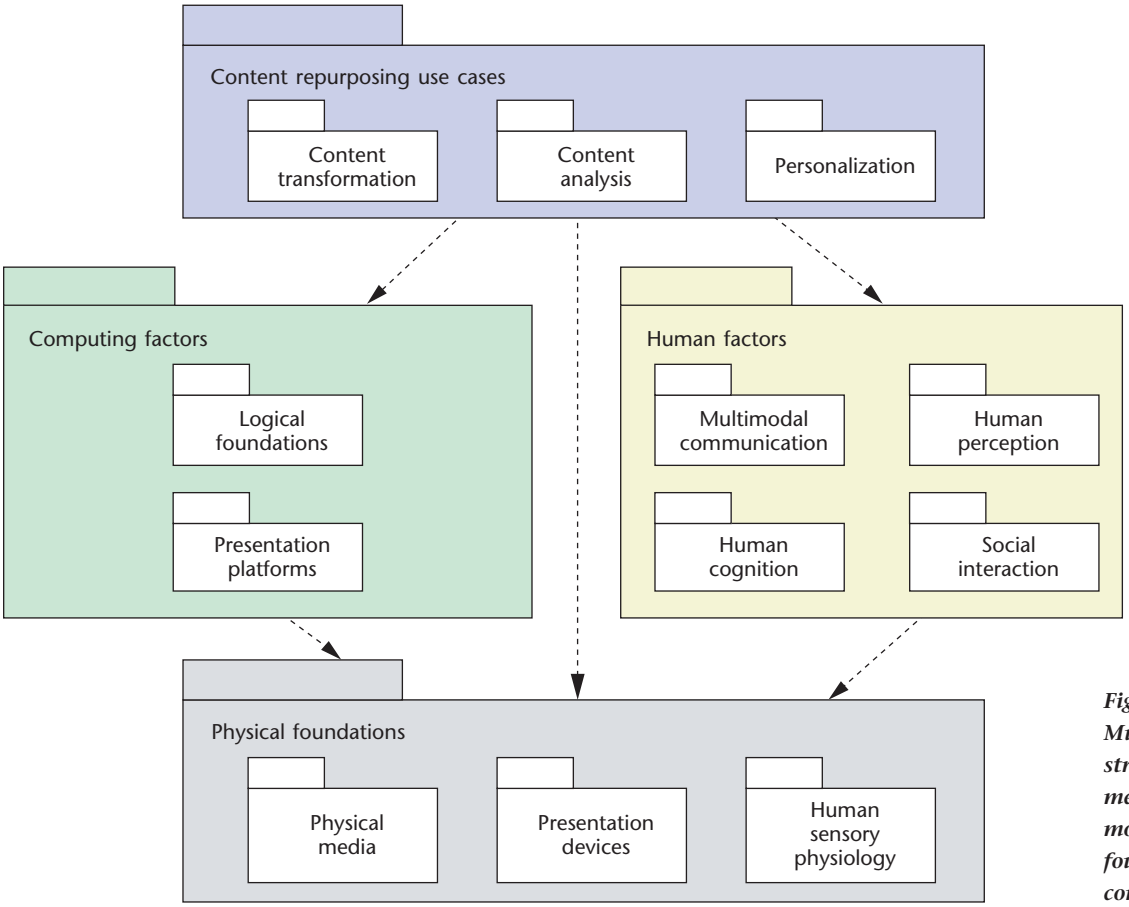


Figure 2. The Multimedia Metamodel structure. The metamodel has four modules—physical foundations, computing factors, human factors, and content repurposing use cases—each offering a different view on the content.

management technologies such as data mining, which can help us find interesting data patterns.

The Multimedia Metamodel

Our aim with the Multimedia Metamodel was to identify basic concepts from each multimedia domain and the relations among those concepts. Our practical goals were to achieve

- a set of precisely defined terms and structured definitions of multimedia concepts;
- high expressiveness, to facilitate efficient descriptions;
- knowledge base coherence and interoperability, using standard modeling and storage technologies; and
- metamodel scalability, to give us the means to define domain concepts at different abstraction levels.

To describe the metamodel, we used the Meta Object Facilities (MOF), a UML subset (see http://www.omg.org/mof). UML is an open standard, and we've widely used its standard mechanisms for defining extensions for specific applications contexts.

As Figure 2 shows, we partitioned the Multimedia Metamodel into four modules: physical foundations, computing factors, human factors, and content repurposing use cases. Each module represents a different viewpoint on multimedia content and content repurposing.

Physical foundations

The physical foundations module introduces concepts for modeling real-world physical properties for use in multimedia presentations. It contains three packages: physical media, human sensory physiology, and presentation devices.

The *physical media* package has concept definitions for stimuli, such as light and sound, and their sensible properties, such as intensity (amplitude) or frequency (pitch). The *human sensory physiology* package contains concept definitions related to the human sensory apparatuses that

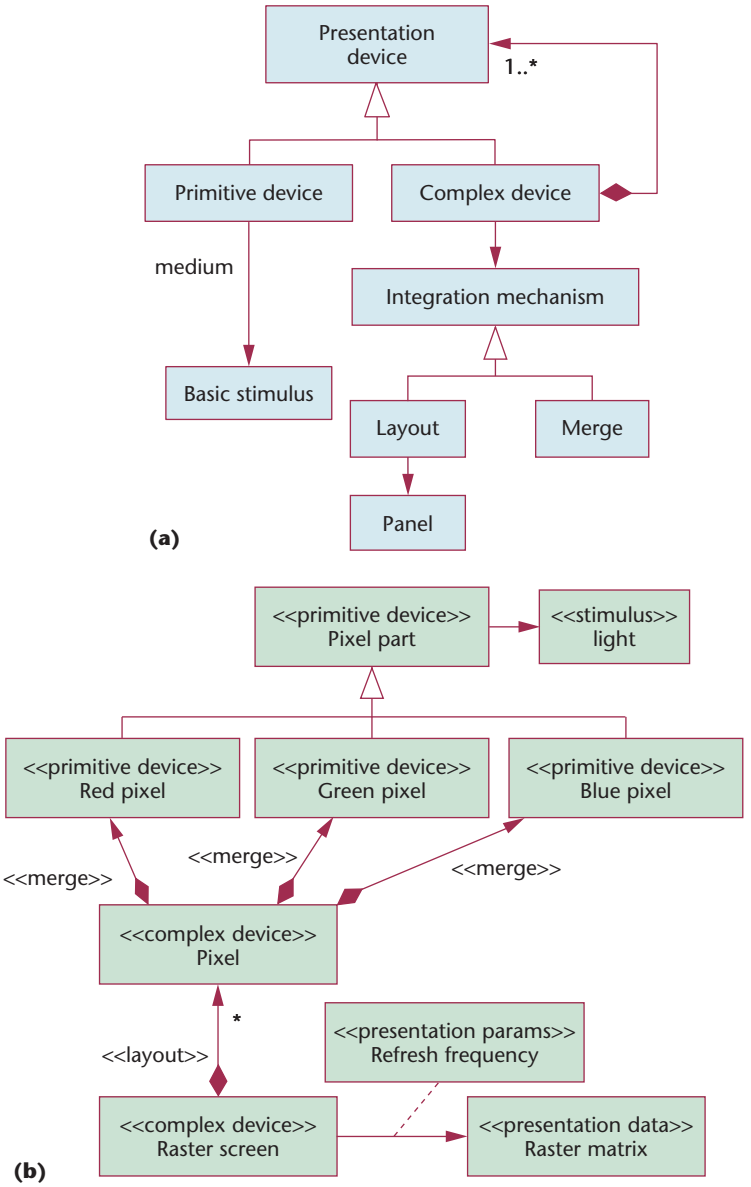


Figure 3. Overview of the presentation devices package. (a) A simplified presentation device metamodel, and (b) an example raster screen display.

process stimuli. Using this knowledge, for example, we can simplify a data presentation by excluding signals that humans are insensitive to, by exploiting effects such as frequency masking or time masking.[15]

The *presentation devices* package defines the metamodel of multimedia presentation devices (see Figure 3a). This package is connected with the physical media package, as we define a presentation device as a stimulus source. The presentation device can be simple or complex; complex devices integrate two or more other devices using merge or layout integration mechanisms. A merge mechanism creates new presentation devices by fusing two simpler ones. A layout presentation device arranges two or more devices on a common panel. We've used this package to describe various presentation devices used in our research, such as the raster screen display shown in Figure 3b.

Computing factors

The computing factors module introduces basic multimedia concepts from computing technology grouped into two packages: logical foundations and presentation platforms.

The *logical foundations* package extends modeling elements with basic logical multimedia concepts.[11] It consists of four subpackages:

- *presentation dimensions* define concepts related to a presentation's spatial and temporal dimensions, including coordinate systems and time bases;
- *coding* defines concepts related to content's physical representation (for example, defining that the same content can have textual or binary encoding);
- *composition* defines mechanisms that seamlessly combine various types of multimedia objects, including temporal composition (synchronization) and spatial composition mechanisms; and
- *quality of service* defines ways to quantify a description of a presentation platform's desired quality level.[12]

The *presentation platform* package consists of subpackages that define generic descriptions of standard presentation platforms. We derived these concepts by generalizing existing platforms. The package includes five modules. The 2D and 3D graphics subpackages define various standard elements for 2D and 3D graphics, such as shapes and transformations. The textual subpackage defines common elements of various textual platforms, such as HTML or Rich Text Format (RTF). The text-to-speech subpackage defines basic elements of speech synthesis platforms, while the audio module defines common concepts for working with nonspeech audio.

Human factors

The human factors module defines various concepts related to user interactions with online multimedia content. It consists of four packages.

Table 2. Human perception, human cognition, and social interaction packages.

Metamodel	Sample concept categories (classes)	Sample concept subcategories (subclasses)	Examples (objects)
Human perception	Pattern recognition	Visual recognition	Shape, letter, face
		Audio pattern recognition	Melody, phoneme
	Grouping	Visual	Grouping by similarity, motion texture, symmetry, proximity, parallelism, closure, good continuation
		Audio	Grouping by pauses between words and sentences, voice color
	Highlighting	Visual	Highlighting by color, polarity, brightness, orientation, size, motion, flicker, depth, shape
		Audio	Highlighting by intensity, rate, speed
	3D cues	Visual 3D cues	Stereo vision, motion parallax, linear perspective (converting lines), relative size, shadow, familiar size, interposition, relative height, horizon
		Audio 3D cues	Interaural time (or phase) difference, interaural intensity (or level) difference, head-related transfer functions (HRTFs), head movement, echo, high frequencies attenuation for distant objects
Human cognition	Memory	Spatial memory	Arranging documents in 3D office model
	Attention	Attention by pop-out	Attention by motion
	Context	Context and details	Exploring hierarchical structures
	Goals	Long-term goals	Overall aim of user's interaction
		Short-term user goals	Current application task
Social interaction	Person	Person's descriptions	User profiles
	Avatar	Username	Line prefix in textual chat
		3D character	Network games
	Collaborative activity	Collaborative filtering	Book or music recommendations
	Session	Temporary session	Chat, videoconferences
		Persistent session	Email, discussion groups, news
	Space	Physical space	Home, job, classroom

Table 2 summarizes three of them:

- The *human perception* package identifies common perceptual concepts related to user interfaces, such as that humans more readily notice highlighted elements.
- The *cognitive mechanisms* package includes several mechanisms recently identified in HCI research, such as spatial memory, attention, and curiosity.
- The *social interaction* package defines common concepts found in online communities, such as the avatar.[16]

The fourth package, *multimodal communication*, defines computing mode and multimodal communication concepts. A computing mode is our metamodel's main concept; we define it as a form of interaction designed to engage some of the human capabilities. While designing a user interface, for example, the designer defines an interactive language that determines which messages and levels to include in the interaction. We classify messages that a mode can send in three main categories: sensual, perceptual, and cognitive. A computing mode can be simple, representing a primitive presentation form, or complex, integrating other modes to simultaneously use various modalities.[17]

Content repurposing use cases

The content repurposing use cases module provides concepts to describe service functions that support multimedia content operation and management. Because this module defines mechanisms for content transformation, analysis, and

Table 3. A sample mapping of perceptual effects to textual and text-to-speech platforms.

Effect	Textual platforms	Text-to-speech platforms
Pattern recognition	Letter and word recognition	Phoneme, spelling, words
Grouping	Grouping by similarity (font family), letter proximity (words), parallelism (rows of text), word proximity (paragraphs), closure and surroundedness (frames and tables)	Gender, age, name; word-breaking rules; sentence-breaking rules; silence
Highlighting	Size (bold and font size), orientation (italic), color (color of text), flicker (blinking text)	Emphasized text; voice volume, speed, pitch
3D cues	Interposition (*z*-index), shadow	Available stereo formats

personalization, it can make the Multimedia Metamodel an active, well-integrated part of content repurposing solutions. This module consists of three packages: content transformation, content analysis, and personalization.

The *content transformations* package provides mechanisms for defining content conversions into mappings and transformations. Developers use mappings between abstract layers and use transformations within abstract layers (see Figure 1). It's possible, for example, to define mappings between platform-specific models, such as XHTML, and corresponding higher-level metamodels, such as the textual platform metamodel. Conversely, it's possible to define transformations between various presentation platform metamodels at the same abstraction level, such as between textual and text-to-speech platform metamodels. The generic mechanisms this package defines can serve as a foundation for more elaborate solutions.

The *content analysis* package defines a framework for content examination approaches, such as data mining. Developers can use the analysis results to evaluate content, especially if the results are connected to perceptual and cognitive metamodels. Contemporary user interfaces often fail to create perceptually feasible artifacts. Using this package, developers can analyze interfaces according to various criteria such as human perceptions, ergonomics, and QoS.

The *personalization package* defines abstract mechanisms to customize multimedia content based on user profiles. Individuals process sensory stimuli in their own unique ways, and each individual has his or her own preferred sensory mode, which might change depending on context.[18] Therefore, to achieve effective personalization, this package correlates various concepts from human factors, media characteristics, and presentation platform possibilities.

Model-driven content repurposing

To illustrate some of the possibilities of a model-driven approach, we apply it to two content repurposing problems: perceptual content repurposing and new multimedia content design. In the first case, we use the high-level Multimedia Metamodel terms to create intermediate representation for various content formats. In the latter case, we integrate the high-level multimedia terms into an existing modeling environment to create new multimedia content.

Perceptual content repurposing

Perceptual content repurposing demonstrates our repurposing strategy, which attempts to keep the original content's perceptual preferences. To illustrate this, Table 3 offers simple examples of high-level perceptual content mappings into primitives available on two platforms: textual platforms such as XHTML or WML, and text-to-speech platforms such as JavaSpeech, Microsoft Speech API, and VoiceXML.

Based on this mapping, we're developing simple tools that

- parse XHTML and HTML files,
- create a new file with higher-level markups using terms such as "group" and "highlighting," and
- transform this high-level markup into text-to-speech presentation.

Figure 4 shows this transformation on an example using XHTML, Microsoft Speech API XML, and WML.

High-level user interface modeling

It's generally easier to repurpose content if the author's intention is clear. Primitives that describe

<table>
<tr><td>PIM XML fragment</td><td colspan="3"><CONTENT>
<CONTEXT type =" lang " param =" en ">
< GROUPING type =" proximity ">
This is
<HIGHLIGHT type =" orientation ">
sample
<HIGHLIGHT type =" weight ">
text
</ HIGHLIGHT >
</ HIGHLIGHT >.
</ GROUPING >
</ CONTEXT >
</ CONTENT></td></tr>
<tr><td>XSL Transformations</td><td><xsl:template match =" HTML/BODY ">
< CONTENT>
<xsl:apply -templates />
</ CONTENT>
</ xsl:template >

<xsl:template match =" SPAN [@lang] ">
< CONTEXT type =" lang ">
< xsl:attribute name =" param ">
<xsl:value -of select =" @lang " />
</ xsl:at tribute >
< xsl:apply -templates />
</ CONTEXT >
</ xsl:template >

<xsl:template match =" P">
< GROUPING type =" proximity ">
<xsl:apply -templates />
</ GROUPING >
</ xsl:template >

<xsl:template match =" B ">
< HIGHLIGHT type =" weight ">
<xsl:appl y-templates />
</ HIGHLIGHT >
</ xsl:template >

<xsl:template match =" I ">
<HIGHLIGHT type =" orientation ">
<xsl:apply -templates />
</ HIGHLIGHT >
</ xsl:template ></td><td><xsl:template match =" UI ">
< SAPI >
<xsl:apply -templates />
</ SAPI >
</ xsl: template >

<xsl:template match = " CONTEXT [@type='lang'] ">
< LANG >
< xsl:attribute name =" langid ">
<xsl:value -of select =" java: SAPITools.getLangId(@param) " />
</ xsl:attribute >
< xsl:apply -templates />
</ LANG>
</ xsl:template >

<xsl:template match =" GROUPING [@type='proximity'] ">
<xsl:apply -templates />
< SILENCE msec =" 1500 " />
</ xsl:template >

<xsl:template match =" HIGHLIGHT [@type='orientation'] ">
< PITCH middle =" 6">
<xsl:apply -templates />
</ PITCH >
</ xsl:template >

<xsl:template match =" HIGHLIGHT [@type='weight'] ">
< VOLUME level =" 90 ">
<xsl:apply -templates />
</ VOLUME >
</ xsl:template ></td><td><xsl:template match =" UI ">
< WML>
< CARD >
<xsl:apply -templates />
</ CARD >
</ WML>
</ xsl:template >

<xsl:template match = "GROUPING [@type= 'proximity'] ">
< P>< xsl:apply -templates />
</ P>
</ xsl:template >

<xsl:template match = "HIGHLIGHT [@type='orientation'] ">
< EM>< xsl:apply -templates />
</ EM>
</ xsl:template >

<xsl:template match = "HIGHLIGHT [@type ='weight'] ">
< STRONG >< xsl:apply -templates /> </ STRONG >
</ xsl:template ></td></tr>
<tr><td>Direction</td><td>↑</td><td>↓</td><td>↓</td></tr>
<tr><td>PSM XML fragment</td><td><HTML>
<BODY >
< SPAN lang =" en">
< P >This is
< I >sample < B >text </ B ></ I >.
</ P >
</ SPAN >
</ BODY >
</ HTML></td><td><SAPI >
< LANG langid =" 409 ">
This is
< PITCH middle =" 6">
sample
< VOLUME level =" 90">
text
</ VOLUME >
</ PITCH >.
< SILENCE msec =" 1500 " />
</ LANG>
</ SAPI ></td><td><WML>
<CARD >
< P >
This is
< EM>
sample
< STRONG > text </ STRONG >
</ EM>.
</ P >
</ CARD >
</ WML></td></tr>
<tr><td>Format</td><td>XHTML</td><td>XML SAPI</td><td>WML</td></tr>
</table>

Figure 4. Simplified version of a possible XML-based implementation of perceptual content repurposing in HTML, Microsoft SAPI XML, and WML formats. We made the transformations using XSL, which here translates XHTML into a platform-independent format. We also made two transformations that translate a platform-independent format into SAPI XML and WML format, respectively.

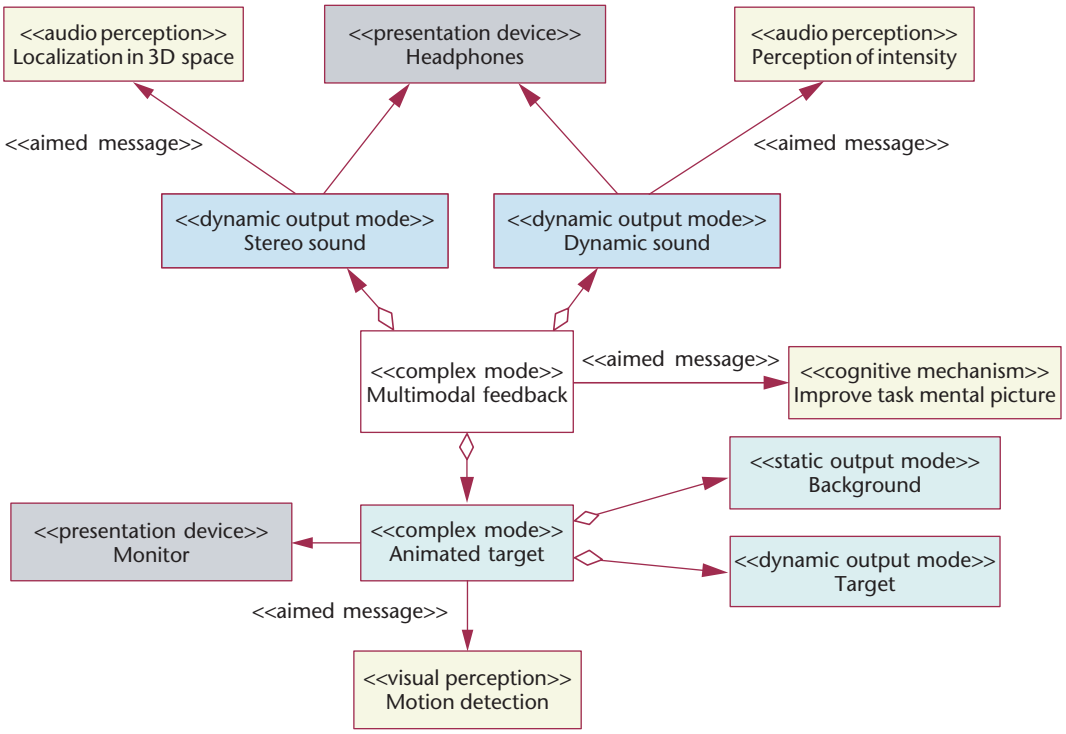


Figure 5. A model of a multimodal pursuit-tracking environment.

most existing multimedia content are at a relatively low abstraction level, and thus it's sometimes impossible to determine the author's original intent. Higher-level multimedia interface models can help better track the content author's aims.

To illustrate this, we described the environment for evaluating multimodal feedback in dynamic pursuit-tracking tasks. The dynamic pursuit-tracking interaction paradigm has valuable applications in surgery, low-visibility mission navigation, and flight navigation and orientation. We examined the effects of multimedia presentation and multimodal feedback in this task, extending the visual feedback with various sonification paradigms.[19]

Figure 5 shows the high-level model of our multimodal pursuit-tracking environment. We used UML's extension mechanisms (stereotypes) to describe such models. The pursuit-tracking interface's multimodal feedback is a complex presentation mode that integrates a visual presentation and two audio modes: a static background presentation and an animated target to attract the user's visual motion-detection perceptual mechanism. The first audio mode is designed to produce 3D stereo effects using stereo cues, while the second audio mode changes the sound intensity to warn users when they've introduced an error.

We can use high-level models to automate some design phases for new multimedia interfaces. For example, we're developing tools to generate Java-based multimedia interface frameworks. These tools take as input the XML description of a high-level model exported from UML models, parse it, and produce Java code files with an abstract Java framework that represents a basic user interface design. The framework also supports generic mechanisms that designers can use to extend the framework with platform-specific modality implementations.

Future work

Our model-driven approach and proposed design solutions are useful not only for many multimedia designers and researchers, where the model-driven tools can help them create better multimedia interfaces, but also for lecturers and students of multimedia courses. In the latter case, the unified Multimedia Metamodel offers context for sometimes subtle relationships between multimedia concepts. The metamodel can also facilitate the collaborative creation of broader knowledge about multimedia phenomena.

In our future work, we plan to extend the proposed metamodel and include domains from related fields, such as user modeling and intelligent tutoring systems that deal with high-level user models. We're also designing multimodal test environments, reusable multimedia components, and data mining tools for evaluating var-

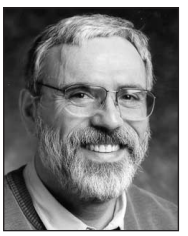
ious aspects of multimedia and multimodal communication. **MM**

References

1. G. Singh, "Media Spaces," *IEEE MultiMedia*, vol. 6, no. 2, April–June, 1999, pp. 18-19.
2. R. Gonzales, "Disciplining Multimedia," *IEEE MultiMedia*, vol. 7, no. 3, July–Sept. 2000, pp. 72-78.
3. S. Brodsky, "XMI Opens Application Interchange," white paper, IBM research, 1999; http://www-3.ibm.com/software/awdtools/standards/xmiwhite0399.pdf.
4. H.W. Lie and J. Saarela, "Multipurpose Web Publishing Using HTML, XML, and CSS," *Comm. ACM*, vol. 42, no. 10, Oct. 1999, pp. 95-101.
5. A. Ram et al., "PML: Adding Flexibility to Multimedia Presentations," *IEEE MultiMedia*, vol. 6, no. 2, April-June 1999, pp. 40-52.
6. E.J. Posnak, R.G. Lavender, and H.M. Vin, "An Adaptive Framework for Developing Multimedia Software Components," *Comm. ACM*, vol. 40, no. 10, Oct. 1997, pp. 43-47.
7. P. Fraternali and P. Paolini, "Model-Driven Development of Web Applications: The Autoweb System," *ACM Trans. Information Systems*, vol. 28, no. 4, Oct. 2000, pp. 323-382.
8. S.J. Mellor, A.N. Clark, and T. Futagami, "Model-Driven Development," *IEEE Software*, vol. 20, no. 5, Sept./Oct. 2003, pp. 14-18.
9. J. Bezivin, "From Object Composition to Model Transformation with the MDA," *Proc. 2001 Tech. Object-Oriented Languages and Systems* (Tools USA), IEEE CS Press, 2001, pp. 350-355.
10. J. Poole et al., *Common Warehouse Metamodel*, OMG Press, 2002.
11. S. Battista, F. Casalino, and C. Lande, "MPEG-4: A Multimedia Standard for the Third Millennium, Part 2," *IEEE MultiMedia*, vol. 7, no. 1, Jan.–Mar. 2000, pp. 76-84.
12. B. Selic, "A Generic Framework for Modeling Resources with UML," *Computer*, vol. 33, no. 6, June 2000, pp. 64-69.
13. M. Gruninger and J. Lee, "Ontology Applications and Design," *Comm. ACM*, vol. 45, no. 2, Feb. 2002, pp. 39-41.
14. C.W. Holsapple and K.D. Joshi, "A Collaborative Approach to Ontology Design," *Comm. ACM*, vol. 45, no. 2, Feb. 2002, pp. 42-47.
15. K.R. Boff, L. Kaufman, and J.P. Thomas, eds., *Handbook of Perception and Human Performance: Sensory Processes and Perception*, vols. 1 and 2, John Wiley & Sons, 1986.
16. L.M. Jessup and D. Robey, "The Relevance of Social Issues in Ubiquitous Computing Environments," *Comm. ACM*, vol. 45, no. 12, Dec. 2002, pp. 88-91.
17. M.M. Blattner and E.P. Glinter, "Multimodal Integration," *IEEE MultiMedia*, vol. 4, Winter 1996, pp. 14-24.
18. E.P.D. Pednault, "Representation Is Everything," *Comm. ACM*, vol. 43, no. 8, Aug. 2000.
19. Z. Obrenovic, D. Starcevic, and E. Jovanov, "Experimental Evaluation of Multimodal Human Computer Interface for Tactical Audio Applications," *Proc. IEEE Int'l Conf. Multimedia and Expo (ICME 2002)*, IEEE CS Press, 2002, pp. 29-32.

Zeljko Obrenovic is currently a doctoral student at the University of Belgrade, where he is a teaching assistant and lecturer at the School of Business Administration. His main research interests include human–computer interaction, biomedical engineering, and software methodologies. Obrenovic received his BS and MS degrees in computer science from the University of Belgrade in 1999 and 2002, respectively.

Dusan Starcevic is a professor at the School of Business Administration, University of Belgrade, and a visiting professor at the university's School of Electrical Engineering. His main research interests include distributed information systems, multimedia, and computer graphics. Starcevic received his BS and MS in electrical engineering and his PhD in information systems from the University of Belgrade.



Bran Selic is Principal Engineer at IBM Rational Software and an adjunct professor of computer science at Carleton University. He has more than 30 years' experience in designing and implementing large-scale real-time systems, and is the principal author of *Real-Time Object-Oriented Modeling* (John Wiley & Sons, 1994). Selic received his BS in electrical engineering and MS in systems theory from the University of Belgrade in 1972 and 1974, respectively.

Contact Obrenovic at the School of Business Administration, University of Belgrade, Jove Ilica 154, 11000 Belgrade, Serbia and Montenegro; obren@fon.bg.ac.yu.