Multimodal Content Adaptations for Heterogeneous Devices*

Susan McRoy, Songsak Channarukul, and Syed S. Ali
mcroyi@uwm.edu,songsak@scitech.au.edu,syali@uwm.edu

September 6, 2005

Abstract

We present **Multiface**, a multimodal dialog system that allows users to interact using different devices such as desktop computers, PDAs, and mobile phones. Users can request information and will receive multimodal responses, where the presented content and its modality are customized dynamically and continuously to the individual and the device that they are using. In addition, the system will attempt to assess the user's understanding and adapt its future responses accordingly. Multiface uses a plan-based approach to produce dynamically adaptive content and modalities from an annotated document and models of the user and device.

1 Introduction

In this paper, we will consider how information content and modality can be adapted to best suit the user, the communication device, and the nature of the content. Our model of the user considers user skills (topics the user is presumed to know about), user preferences (modalities that support the user's physical, cognitive, or environmental constraints), and a general skill classification, to which a domain expert or teacher can associate specific information needs. Our model of the device considers resource capabilities and preferences (such as bandwidth, color, audio, screens) as recommended by the World Wide Web Consortium [11]. Our model of the content includes a representation of organizational structure of the overall content, the rhetorical relationships among constituents down to the paragraph level, and information about what skills the constituents may provide or require.

We have implemented a system, Multiface, that dynamically combines information from these three models (user, device, and content) to select content, allocate content to modalities, and combine multimodal information in space and time. Our aim is to construct an **intelligent dialog system**, suitable for applications such as training. We define intelligent dialog systems as those that are concerned with the effective management of an incremental, two-way, user-system interaction. Users are able to express needs, to make requests, and to indicate whether or not they have understood what the system has presented. (These abilities are independent of the mode of communication, which, for example, may involve typed text or direct-manipulation.) Content to be presented, as well as the system's model of the user (for example, the user's apparent expertise), can change dynamically during an interaction. In general, an intelligent dialog system must deal

*We acknowledge the financial support of the National Science Foundation, Assumption University, Wright State University, and Intel Corporation.

with both input and output (interpretation and presentation) and must monitor the effectiveness of its actions. By contrast, so-called intelligent presentation systems [7] are primarily concerned with output issues and do not monitor the effectiveness of the user's learning.

Throughout an interaction, Multiface evaluates and adapts to the user's level of understanding of presented concepts, by administering tests of understanding that have been specified by the document author or that are generated by the system dynamically. It also allows the user to report non-understanding at any time. When the user's level understanding drops (as measured by tests and user reports of non-understanding), the system will provide enriched or alternative content to facilitate learning. When understanding is better than expected, remedial content will be suppressed and more sophisticated content may be added, to keep the user's interest. Alternative content can also be associated with functional classifications of users (such as patient or nurse).

To facilitate dynamic adaptation, Multiface continually evaluates and unifies constraints from both the user and the device into a single set of constraints, called **interaction constraints**. Then, Multiface completes a three phase process to select and realize the content. First, the system performs **adaptive content determination** to build an initial list of content goals. Then, it performs **modality allocation** to construct a list of modality goals based on the interaction constraints. (If necessary, content may be added or suppressed at this time.) Finally, it performs **multimodal output realization** to determine the layout of the presentation. The result is a combination of content, modality, and layout that satisfies both the user and the device.

Existing research on adaptation in both hypermedia and dialog systems includes the customization of content based on user models and interaction history [8, 47]. Some researchers have also investigated device-centered adaptations that range from low-level adaptations such as conversion of multimedia objects [51] (*e.g.,* video to images, audio to text, image size reduction) to higher-level adaptations based on multimedia document models [6]. However, to our knowledge, no work has been done on integrating and coordinating both types of adaptation. Also, unlike previous work, we do not require multiple copies of the same document to be pre-authored for different device types. Instead we adapt to the user and the device dynamically, from a single core representation of content. Our approach is to represent the entire process declaratively, as facts and rules in a generic planning framework, JAM [22]. It represents a balance between the fine-grained adaptiveness of systems that generate content from conceptual representations (but which are necessarily limited in scale) and more coarse-grained adaptiveness of adaptive navigation systems or modality-transforming systems, (but which do not have a rich model of the user or the content).

2 A New Platform for Multimodal Adaptation

We are investigating the optimum combination of content, modality, and layout for each person, device and document is still under investigation. Our contribution to this area includes a system that allows us to assess the effectiveness of particular choices and to easily experiment with alternatives. In particular, we have created an application (an intelligent tutoring shell [23]) that allows us to experiment with different combinations of content, modality, and layout to assess the impact of these strategies on users' satisfaction and learning. In this application, a user will provide a profile of his skills, select from a fixed list of possible devices (including workstations, personal digital assistants, and mobile phone), and choose a learning goal based one of our documents (either blood pressure management or programming data structures). The system will then interact with the user to help him learn or review this content and to assess his learning using the strategies that are

currently under investigation. Over the course of the interaction, the system will dynamically adapt the content presented on the basis of dynamically changing information the user. This information includes both the specific skills of a given user and the most appropriate classification of the user. (The information will be updated on the basis of what sections of a document a user reads, what claims of non-understanding the user makes, and what scores the user receives after taking a test associated with some document constituent.) The overall strategy and the input parameters can be changed easily because we have a domain independent, plan-based approach to selecting content, modality, and layout. We have also created authoring tools that allow the specification of new content and the design of new tests for assessing users' learning.

The core of our approach is a completely plan-based description of how to adapt content and modalities from an annotated document and models of the user and device.

This approach allows us to test and refine our theories of how best to adapt the content and modality to the user and device. It also allows us to show others exactly what theory is guiding the system's behavior at any given moment and to show the impact of changes to the input models on the fly. The approach makes use of domain specific and domain independent information. Domain specific rules are automatically created from an annotated document that defines content units and semantic relations among those units. Domain independent rules represent general knowledge about user models, device characteristics, and modality relations.

3 System Architecture

The architecture of our system connects a web-based interface to a back-end planning agent. Users can interact with Multiface from arbitrary devices; each device will interact with Multiface over standard TCP/IP using any browser that supports XHTML+SMIL.

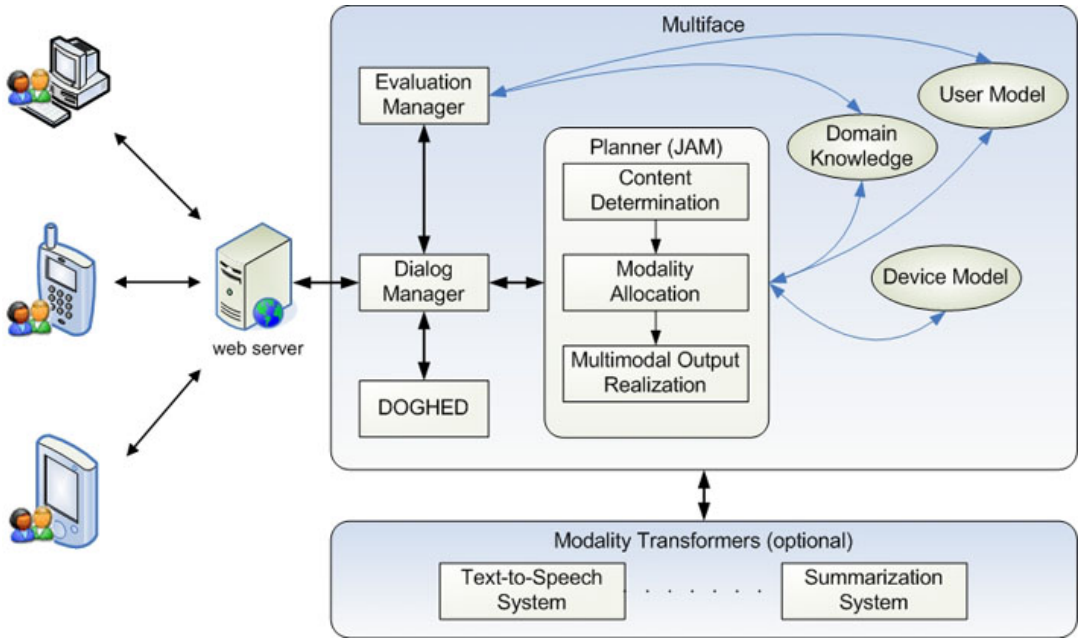


Figure 1: The Architecture of Multiface

The general architecture of our system is shown in Figure 1. It is composed of the following components:

- A *Web Server* mediates the communication between user's devices and Multiface. The web server must be able to handle Java Server Pages (JSP), which we use to receive user's responses and direct presentation output to user's browser.
- *JAM* [22] is a Java-based, intelligent agent. We use JAM as a general-purpose planner to reason about domain knowledge, the user and the device to generate presentation plans.
- *Domain Knowledge* includes annotated documents that define document structure, content units, and semantic relations among them.
- The *User Model* specifies information about user's expertise level, skills, and modality preferences. It is used to determine appropriate content units for individual users and to bias the selection of modalities.
- The *Evaluation Manager* performs incremental assessments of the user's understanding of the content. The results are used to update the user model.
- The *Device Model* specifies characteristics of various devices and how they affect modality selection.
- *DOGHED* (Dialog Output Generator for HEterogeneous Devices) [14] is a system for generating multimodal output. It extends a template-based generation system called YAG (Yet Another Generator) [30, 13] to produce output in either SMIL or XHTML+SMIL.
- *Modality Transformers* are distinguished components used to transform one modality to the other (*e.g.,* Text-to-Speech (TTS) or Summarization System).

When users wish to initiate a session, they must first log on to Multiface and specify the device that they are currently using. A new instance of Multiface is created for this user and annotated documents and the user and device models are converted from an external XML representation into facts and rules of JAM using XSL Transformation (XSLT) [54]. A list of possible topics is then presented to the user and the user is directed to select an initial content goal for the interaction. If the user chooses, he may also select a task (such as review) that will automatically select the initial content goal for the interaction, as well as setting a related content preference, such as summary. Multiface maps the user's selection onto a presentation goal for its planner and initiates planning. The planner then reasons about the content, user, and device to create an appropriate presentation plan. This plan includes a specification of content, modalities, and layout. This plan, expressed as a feature structure, is then realized by planning as SMIL instructions for multimodal presentation. Finally the instructions are passed to the user's browser and the user's response will be returned to Multiface for further processing.

4 The Master Document

All versions of content are generated from a single master document that has been annotated to include information about its organizational structure, semantic (rhetorical relationships) among

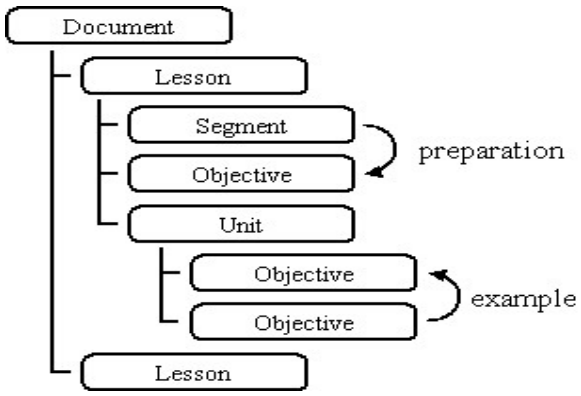


Figure 2: Examples of subsections with semantic relations

various organizational units, locations for assessments, information dependencies among document components and modality-related properties of document components.

The syntactic structure of the master document is a tree (see Figure 2). The **document** is the root, which is composed of one or more lessons. A **lesson** is composed of segments, objectives, and units. A **unit** (a group of objectives or segments that have a related purpose in a presentation) is composed of objectives or segments. An **objective** (a group of segments that have a single purpose e.g. educate the users about the meaning of high blood pressure) is composed of one or more segments. A **segment** is the smallest constituent in the document. It is a paragraph-level piece of text, a heading, or a graphic from the document.

The semantic structure within the master document is captured by means of semantic (rhetorical) links among document objects. Our usage is based on Rhetorical Structure Theory (RST) [26]. Such a link indicates the function between two objects within a document, *i.e.,* between a segment and another segment, a segment and an objective, a segment and a unit, or an objective to another objective. Multiface's planner will make use of these links when reasoning about the appropriateness of each segment for a given user. The use of links also allows a document author to reuse text without repeating it, by allowing links to relate objects from different parts of the document or even objects from different documents. We make use of fifteen link types: Title, Background, Elaborate, Example, Procedure, Real-result, Possible-result, Solution, Concession, Nucleus, Figure, Summary, Restatement, Comparison, and Warning.

Semantic links are added during the authoring process (see Section 8.) If desired, additional types of links can be defined at that time. In the absence of link information, a document can still be presented, but it will not be adapted on the basis of user stereotype information (although it may be adapted based on a user's individual skills).

Information dependencies among document components specify the prerequisite structure of the document. We call this structure the **Skill Network**. A Skill Network represents what skills a content unit (objective, unit, and lesson) *provides* or *requires*. If we consider each content unit as a node in a document, the skill network will describe dependencies among these nodes, yielding a directed graph.

Modality information is associated with individual segments of the document. Modality properties include the duration of a video or audio, the width and height of images or audio. In addition, we represent **modality relations** among pairs of segments that are intended to complement or

substitute for each other.

Multiface's modality relations are based on the TYCOON (TYpes of COOperatioN) framework [27] that is used to observe, evaluate, and specify cooperation among different *input* modalities. Multiface adopts some of these relations to help determine *output* modalities. In the TYCOON framework, there are six cooperation types between modalities as follows:

- **Equivalence**: All modalities equally convey the same information.
- **Specialization**: A specific modality is always used to convey the information.
- **Redundancy**: Several modalities are used to convey the same information.
- **Complementarity**: Different modalities which convey different pieces of information are used together to convey the information. The information is considered incomplete if some modalities are missing.
- **Concurrency**: Several modalities convey independent information at the same time.
- **Transfer**: One modality is used by another modality.

These cooperation types define functional relationships between modalities. For example, an image and its caption are considered *complementary*. Therefore, a caption is useless if the image that it is complementing is not presentable. An audio and its textual counterpart are considered *equivalent*. When audio is not allowed (*e.g.,* when a user is in a noisy environment or when a device does not have speakers), presenting text alone would be better. On the other hand, when a screen space is limited, presenting audio alone might be a better way to present such a segment. In Multiface, we define two types of modality relations: 1) content-specific modality relations and 2) generic modality relations.

Figure 3 illustrates some segment relations within Objective `O231`. Here, Segment `S23101-2` is an audio version of Segment `S23101`. Both segments are considered *equivalent*. Segment `S23111-3` is *redundant* to Segment `S23111-1` and it is a *summary* of `S23111-1`. Segment `S23112-1` and `S23112-2` are two image segments that are *equivalent*. Both images have Segment `S23113` as its *caption* therefore Segment `S23113` has a *complement* relation to both of them.

All four types of knowledge (the syntactic and semantic document structure , the skill network, and the modality information) are represented in a dialect of XML and are converted automatically to facts in the knowledge base of Multiface's planner, which is represented in JAM. They will be used together with the user and the device model to adapt the interaction to the user and the device.

4.1 Planner Representations

Prior to run-time, the document specific XML representations (the syntactic and semantic document structure and the skill network) and the document-independent XML representations (the user model, the modality information and modality relations) are processed into a static set of JAM facts and planning rules. The document specific information represents a default plan for the presenting the document, namely to present each content object in the order in which it was defined in the document. We call this the normative content. (The actual content presented to a user is further constrained by the document independent knowledge, which will include or omit portions of content.)

When the file that defines the document structure is read, each content item (lesson, unit, and objective) is converted to an individual `CD_post-normative-content` plan.

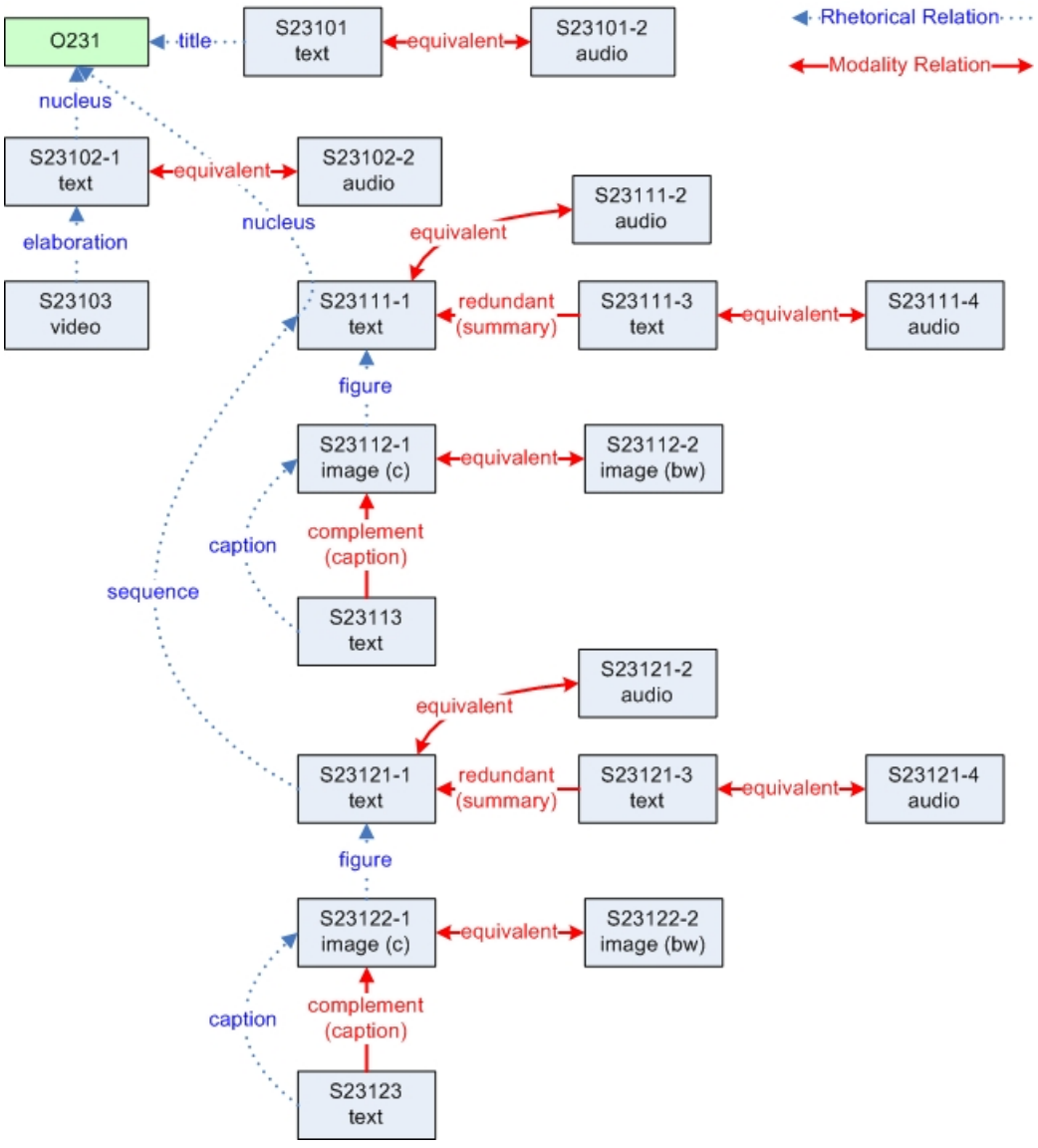


Figure 3: Segment relations in Objective `O231`.

```
Act:      CD_post-normative-content(?user, ?scope, "U21", ?position)
Preconds: user(?user)
          content(?scope) OR task(?scope)
          content(?position) OR ?position == "STOP"
Plan:     CA_RegisterItem(?user, ?scope, "S2101", ?position)
          CA_RegisterItem(?user, ?scope, "S2102", ?position)
          CA_RegisterItem(?user, ?scope, "S2103", ?position)
          CA_RegisterItem(?user, ?scope, "S2104", ?position)
          CA_RegisterItem(?user, ?scope, "O211", ?position)
          CA_RegisterItem(?user, ?scope, "O212", ?position)
          CA_RegisterItem(?user, ?scope, "O213", ?position)
          CA_RegisterItem(?user, ?scope, "O214", ?position)
          // There is a test for Unit U21.
          Test(?user, ?scope, "U21")
```

Figure 4: Plan for setting the normative content of Unit U21.

Figure 4 show the plan corresponding to the structure of Unit U21, which is a component of Lesson L02. The ?user argument in normative content plans specifies the user who requests the presentation. The ?scope argument specifies the content unit or task that the user has requested. The ?position argument specifies a location where the content item will be inserted into the content goal list.

In this example, to present Unit U21 is to present:

- Segment S2101 (the TITLE of Unit U21),
- Segment S2102 (the BACKGROUND of Unit U21),
- Segment S2103 (the NUCLEUS of Unit U21),
- Segment S2104 (the ELABORATION of Segment S2103),
- Segment S2102 (the BACKGROUND of Unit U21),
- Objectives O211, O212, O213, and O214,
- A test after Unit U21+.

The pedagogy of Lesson L02 and its units (Unit U21 to U26) and objectives (*e.g.,* O211, O212) comprise the core content of Lesson L02. The actual content presented to a user is further constrained by the document independent knowledge, which will include or omit portions of content. This basic pedagogy together with a relevant skill network (to be described next) complete the pedagogy of Lesson L02.

The skill network and the modality relations are represented as sets of facts. Figure 5 shows a set of facts that have been derived from the XML representation of a skill network. This network specified that in order to learn Lesson L02, users need to either already know about Types and Operators or will need to learn Lesson L01 first, because Lesson L02 requires skills Types and Operators that Lesson L01 provides. Similarly, to learn Unit U23, users will likely need to learn Unit U21 and U22 first, because they provide the skills (BasicPointer and PointerRules respectively) required by Unit U23.

Modality information and modality relations are represented similarly.

```
skill("Types", "Types")
skill("Operators", "Operators")
skill("BasicPointer", "Basic concept of pointer")
skill("PointerRules", "Basic rules of pointer")
skill("MorePointers", "Advanced topics for pointers")
provides("L01", "Types")
provides("L01", "Operators")
provides("U21", "BasicPointer")
requires("U21", "Types")
requires("U21", "Operators")
provides("U22", "PointerRules")
requires("U22", "BasicPointer")
provides("U23", "MorePointers")
requires("U23", "PointerRules")
provides("O231", "MorePointers")
requires("O231", "PointerRules")
requires("L02", "Types")
requires("L02", "Operators")
```

Figure 5: Sample skill network as facts in Multiface's planner.

5 The User Model

In Multiface, there are two types of user models; the **generic user model** describes information relevant to users in particular classes; while the **specific user model** keeps track of information about individual users. Our generic user model follows the user modeling methodology called stereotypes [45, 46] that captures default information about groups of people. We represent both parts of the user model as a collection of facts or assumptions about the user who is using the system.

Information about classes of users is provided by the developer of a domain (See section 8.) This information includes what types of information each type of user should (and should not see) and what skills that type of user can be presumed to have. (Information about skills may be overridden dynamically during a specific user's interaction with the system.)

Information about individual users is given both as part of a registration process, and also on the basis of the user's interactions. Information provided during registration includes what modalities users prefer (or dislike) and allow (or disallow), as well as what skills the user has. However, information about skills will be revised by the the system dynamically, on the basis of what sections of the document a user reads, what claims of non-understanding the user makes, and what scores the user receives after taking the test associated with some document constituent. Whether a test is required, or just reading is sufficient is determined by the domain expert when the master document is authored. Thus, different testing requirements may be specified for different users. Both the classification of a specific user and the skills associated with a specific user will be adjusted dynamically during the course of an interaction, and the planning of subsequent parts of the interaction will reflect these changes.

Figure 6 shows the plan that would be created to set up the user at the `intermediate1` level. We assume that a novice user has no skills and that an intermediate (`intermediate1`) user has two skills (`Types` and `Operators`). When execution begins, the XML representation of the user model is converted automatically to a set of plans with assertions for Multiface's planner.

In addition to the generic user model, there is a specific user model for each individual user. For example, in the XML file created for the user "`Frodo`", it specifies that he is at the `intermediate1`

```
Act:  SetupUser(?user, "intermediate1")
Plan: RETRACT express(?user)
      ASSERT express(?user, "Background")
      ASSERT express(?user, "Comparison")
      ASSERT express(?user, "Example")
      ASSERT express(?user, "Figure")
      ASSERT express(?user, "Nucleus")
      ASSERT express(?user, "Possible-result")
      ASSERT express(?user, "Real-result")
      ASSERT express(?user, "Restatement")
      ASSERT express(?user, "Summary")
      ASSERT express(?user, "Title")
      ASSERT express(?user, "Warning")
      ASSERT user_possess(?user, "Types")
      ASSERT user_possess(?user, "Operators")
```

Figure 6: Plan for setting the user at the `intermediate1` level.

level and possesses two skills (`Operators` and `PointerRules`). He prefers text and color, allows image, dislikes audio, and disallows video. Figure 7 shows the plan that will be created when the XML is processed. Both plans in Figures 6 and 7 are examples of how initial facts about the user are asserted in the planner.

```
Act:      LoadProfile("Frodo", ?scope)
Preconds: content(?scope) OR task(?scope)
Plan:     ClearProfile("Frodo")
          InitializeUserConstraints("Frodo", ?scope)
          ASSERT __UserHasSkill("Frodo", "Operators")
          ASSERT __UserHasSkill("Frodo", "PointerRules")
          ASSERT __UserPrefers("Frodo", ?scope, "text")
          ASSERT __UserAllows("Frodo", ?scope, "image")
          ASSERT __UserPrefers("Frodo", ?scope, "color")
          ASSERT __UserAllows("Frodo", ?scope, "audio")
          ASSERT __UserDislikes("Frodo", ?scope, "audio")
          ASSERT __UserDisallows("Frodo", ?scope, "video")
```

Figure 7: Plan for setting up a specific user model for the user "`Frodo`".

6 Device Models

Our **device model** is a representation of modality preferences and various properties of the user's device. It specifies what modalities are *preferred*, *allowed*, *disliked*, and *disallowed* by the device in the same way that the user's modality preference does. However, it is different from the user's modality preference in that it also specifies properties of the device that are used as constraints. Multiface uses this information to filter out content that is not suitable for the user's current device.

We have taken an object-oriented approach to achieve the necessary flexibility, by:

- classifying devices into primary types such as desktop computer, hand-held computer, and mobile phone,
- defining what modalities are *preferred* and *allowed* for each device type, and
- specifying other devices that belong to the existing device type but have some different properties and modality preferences.

```
<device type="handheld-computer">
  <modalities>
    <modality id="audio" allowed="no"/>
    <modality id="text" preferred="yes"/>
    <modality id="image" allowed="yes" preferred="no"/>
    <modality id="color" allowed="no"/>
    <modality id="video" allowed="no"/>
  </modalities>
  <properties>
    <property id="screen-size" value="small"/>
    <property id="network-bandwidth" value="none"/>
  </properties>
</device>
```

Figure 8: Example of a device model for handheld computers.

```
<device type="smart-handheld-computer" inherits="handheld-computer">
  <modalities>
    <modality id="audio" allowed="yes" preferred="no"/>
    <modality id="image" preferred="yes"/>
    <modality id="color" allowed="yes"/>
  </modalities>
  <properties>
    <property id="screen-size" value="medium"/>
    <property id="network-bandwidth" value="low"/>
  </properties>
</device>
```

Figure 9: Example of a device model for smart handheld computers.

For example, we define handheld computers as devices whose audio modality is *disallowed* because most devices in this category support only basic audio (*i.e.,* beeping). Its device model is given in Figure 8. We also define a device type, called "smart handheld computer" that inherits modality preferences and properties from "handheld computer" but its audio modality is *disliked* (*i.e., allowed* but still not *preferred*). Figure 9 shows a device model of smart handheld computers.

Even more specific devices such as a "Sony CLIÉ-TH45" can be specified that inherits from "smart handheld computer" where its audio and video modality are set to *preferred* due to its high-end speaker, headphone supports, and better screen resolution. Its device model can be specified as shown in Figure 10.

A plan for setting up a device model is automatically generated from the specified device model, represented in XML. Figure 11 gives an example of the plan for setting up the device model for Sony CLIÉ-TH45 (associated with the XML representation given in Figure 10).

```
<device type="Sony Clie-TH45" inherits="smart-handheld-computer">
  <modalities>
    <modality id="audio" allowed="yes" preferred="yes"/>
    <modality id="video" allowed="yes" preferred="yes"/>
  </modalities>
  <properties>
    <property id="screen-size" value="medium">
      <width>230</width>
      <height>320</height>
    </property>
    <property id="network-bandwidth" value="medium"/>
  </properties>
</device>
```

Figure 10: A device model of Sony CLIÉ-TH45.

```
Act:      SetupDevice(?user, ?scope, "Sony Clie-TH45")
Preconds: user(?user)
          content(?scope) OR task(?scope)
Plan:     ClearDeviceModel(?user)
          InitializeDeviceConstraints(?user, ?scope)
          ASSERT __DeviceAllows(?user, ?scope, "Sony Clie-TH45", "audio")
          ASSERT __DevicePrefers(?user, ?scope, "Sony Clie-TH45", "audio")
          ASSERT __DeviceAllows(?user, ?scope, "Sony Clie-TH45", "video")
          ASSERT __DevicePrefers(?user, ?scope, "Sony Clie-TH45", "video")
          ASSERT __DevicePrefers(?user, ?scope, "Sony Clie-TH45", "image")
          ASSERT __DeviceAllows(?user, ?scope, "Sony Clie-TH45", "color")
          ASSERT __DevicePrefers(?user, ?scope, "Sony Clie-TH45", "text")
          ASSERT __Requires(?user, ?scope, "Sony Clie-TH45", "width", "<=", 230)
          ASSERT __Requires(?user, ?scope, "Sony Clie-TH45", "height", "<=", 320)
```

Figure 11: Plan for setting up a device model for Sony CLIÉ-TH45.

7 Interaction Constraints

To address both the preferences and constraints expressed in user and device models, Multiface creates a unified set of constraints, called **interaction constraints**. These constraints affect how Multiface allocates modalities for each segment in two ways. First, interaction constraints are used to filter out segments that are not applicable, either because of the user, the current device, or both. Second, Multiface also uses interaction constraints to combine two or more modalities when there are several applicable alternatives.

Preference Interpretation	`allowed`	`preferred`
Prefer	`yes`, `undefined`	`yes`
Allow	`yes`, `undefined`	`undefined`
Dislike	`yes`, `undefined`	`no`
Disallow	`no`	`yes`, `no`, `undefined`

Table 1: Interpretation of `allowed` and `preferred` parameters.

Multiface determines user's modality preferences based on the two parameters specified in the specific user model. Table 1 presents various combinations of these two parameters and their interpretation. For example, when the user allows (`allowed = yes`) and prefers (`preferred = yes`) a certain modality, that modality is marked as a ***preferred*** modality. When a parameter is undefined, we decrease the level of preference such that it is not biased toward any strong preference. For example, when the user prefers (`preferred = yes`) a certain modality but does not specify whether it is allowed or not, we take this as a *preferred* modality. Similarly, if the user does not prefer or dislike a modality (`preferred` is undefined) and that modality is allowed or undefined, we consider this an ***allowed*** modality. However, if the user dislikes a certain modality (`preferred = no`) and this modality is allowed or undefined, we consider this a ***disliked*** modality. Finally, if the user does not allow (`allowed = no`) any modality, that modality will be marked as a ***disallowed*** modality, no matter what the user specifies in the `preferred` parameter.

Multiface creates interaction constraints by unifying user and device constraints specified for individual users and their current devices. Table 2 presents how different user and device constraints will be unified. Sample scenarios associated with each case are also given.

The plan given in Figure 12 implements the unification process that creates interaction constraints from user and device constraints according to Table 2. In addition, a data driven plan will add an interaction constraint whenever the plan for setting up the device asserts a `__Requires` fact (such as the maximum screen size of a device).

Case	User Constraint	Device Constraint	Interaction Constraint	Sample Scenarios
1.	prefer	prefer	prefer	Both user and device prefer text.
2.	prefer	allow	allow	A user prefers image but the device's screen resolution is not very good.
3.	prefer	dislike	allow	A user prefers video and the device can display video but has a low bandwidth.
4.	allow	prefer	prefer	A device has good audio but the user is neutral about it.
5.	allow	allow	allow	Both user and device do not have a strong preference for video.
6.	allow	dislike	dislike	A user is neutral about video and the device cannot play video very well.
7.	dislike	prefer	allow	A user dislikes audio but the device favors it over other modes (*e.g.,* a telephone).
8.	dislike	allow	dislike	A user dislikes video and the device is neutral about it.
9.	dislike	dislike	dislike	Both user and device dislike video.
10.	disallow	anything	disallow	A user is deaf therefore audio is disallowed.
11.	anything	disallow	disallow	A device does not have a speaker therefore audio is disallowed.

Table 2: Unifying user and device constraints to interaction constraints.

Rule 1

```
Conclude: __UnifyConstraints(?user, ?scope)
Preconds: user(?user)
          content(?scope) OR task(?scope)
Plan: DO-ALL (?modality)
             (modality(?modality))
        // Case 1 and 4
        IF ((user-constraint(?user, ?scope, "prefer", ?modality) OR
             user-constraint(?user, ?scope, "allow", ?modality)) AND
            device-constraint(?user, ?scope, "prefer", ?modality)) THEN
          ASSERT interaction-constraint(?user, ?scope, "prefer", ?modality)
        // Case 2 and 5
        ELSE IF ((user-constraint(?user, ?scope, "prefer", ?modality) OR
                  user-constraint(?user, ?scope, "allow", ?modality)) AND
                 device-constraint(?user, ?scope, "allow", ?modality)) THEN
          ASSERT interaction-constraint(?user, ?scope, "allow", ?modality)
        // Case 3
        ELSE IF (user-constraint(?user, ?scope, "prefer", ?modality) AND
                 device-constraint(?user, ?scope, "dislike", ?modality)) THEN
          ASSERT interaction-constraint(?user, ?scope, "allow", ?modality)
        // Case 6
        ELSE IF (user-constraint(?user, ?scope, "allow", ?modality) AND
                 device-constraint(?user, ?scope, "dislike", ?modality)) THEN
          ASSERT interaction-constraint(?user, ?scope, "dislike", ?modality)
        // Case 7
        ELSE IF (user-constraint(?user, ?scope, "dislike", ?modality) AND
                 device-constraint(?user, ?scope, "prefer", ?modality)) THEN
          ASSERT interaction-constraint(?user, ?scope, "allow", ?modality)
        // Case 8 and 9
        ELSE IF (user-constraint(?user, ?scope, "dislike", ?modality) AND
                 (device-constraint(?user, ?scope, "allow", ?modality)) OR
                  device-constraint(?user, ?scope, "dislike", ?modality)) THEN
          ASSERT interaction-constraint(?user, ?scope, "dislike", ?modality)
        // Case 10 and 11
        ELSE IF (user-constraint(?user, ?scope, "disallow", ?modality) OR
                 device-constraint(?user, ?scope, "disallow", ?modality)) THEN
          ASSERT interaction-constraint(?user, ?scope, "disallow", ?modality)
        END-IF
      END-DO
```

Figure 12: Plan for creating interaction constraints.

8 Authoring

Multiface requires representations of the the user model, the domain (skill network), the master document (content and pedagogy) in order to dynamically adapt the interaction. These representations must be authored by an expert in the domain. These tasks vary significantly in their difficulty and re-usability. We will consider each in turn.

Creating the user model is not especially difficult or labor intensive, and the results are domain independent. For many tasks, it is sufficient to define only a small number of user types (less than ten). There are typically only a small number of semantic link types (we use fifteen) defined in the document representation, so defining what to suppress or express is similarly easy to state and modify. Because the files are so small (less than 50 lines) we have done this by editing the XML directly.

Creating the skill network is more difficult. This task requires that the domain expert identify all the skills (concepts) in the domain and create a dependency structure over them. While in theory this might be created before any documents are created (as part of the process of creating an outline), in practice we have found this easiest to do after an initial content has been created. The size of the networks and the number of skills involved suggests that it would be very helpful to have a tool that creates a visual representation of the network. While we have not created such a graph visualization tool ourselves, this problem is common to any task that makes use of graph-based information and so there may be existing visualization tools that could be used.

Creating the master document represents the most significant amount of work, comparable to time needed to develop a new textbook or technical manual (not including the typesetting that happens after the content is written). However, unlike a new textbook, the same Multiface document can be use to create a variety of different dynamically adapted interactions to suit different users, tasks and devices. For example, the same document related to blood pressure management, can be used by patients who want to know how to control high blood pressure, by family caregivers who help monitor blood pressure at home, by nurses or nursing students who must know how to counsel patients to seek followup care and how to maintain blood pressure equipment, and they will each only see those parts of the document that are relevant to their needs and in language they could be expected to understand. We have been concerned with making the authoring process tractable and hence our approach relies on a coarse-grained text-based representation of content (about the level of a paragraph) rather than on a fine-grained, concept-based representation[1] used in work such as IDAS, PEBA II, and PPP [44, 33, 1]. Also, the specific semantic labels that are used are not crucial per se, as long as the document author uses them consistently later when specifying models of different classes of users. We have taken an existing set of documents (portions of the *Blood Pressure Measurement Education Program Instructor Manual* [37] and the Stanford CS Library [40]) and hand-created the necessary segment and pedagogy files. This effort required a few days per lesson for a person with technical expertise.

We have designed two separate tools for to help non-technical users create a Master Document. Creating the master document requires specifying two types of information: the segments that contain the actual words or file references that make up the document and the pedagogy which specifies the hierarchical and sequential organization for the (unadapted) document. The latter will also include semantic relations (links) among different document components. The first tool

[1] Fine-grained approaches represent text at the word or phrase-level and use natural language generation techniques to generate text from logic-based representations of meaning.

allows one to author a master document from scratch. The tool provides a form-based interface for creating new organizational units of each type, for adding semantic links between units, and for specifying preferred locations for assessments of the user's knowledge (*i.e.,* tests). The output is an XML representation that can be used by Multiface. The second tool allows one to take an existing document and add the necessary annotation. In this second tool, an original machine-readable document (e.g. in MSWord or HTML) is semi-automatically processed into a sequence of segments (corresponding, roughly, to paragraphs, figures, tables, and headings). These segments must then be authored into a coarse, hierarchical, syntactic structure, which includes some semantic information about relationships between document objects. Work integrating the two approaches is ongoing.

In addition to the core content, the author of a document can also specify tests to be associated with specific parts of the document. Tests can either be automatically generated or can be authored. Multiface can automatically generate Cloze tests [28] for document components. (A Cloze test is a test of reading comprehension that asks the user to fill in missing words from the presented text.) The master document can specify the frequency of missing words and whether high frequency words should be ignored when selecting these words. Multiple choice, true-false questions and Cloze tests where the missing words have all be predefined can be specified by a document author explicitly. We have constructed a GUI-based tool for authoring pre-defined Cloze Tests.

9 Multimodal Content Generation

Multimodal content generation in Multiface is a three-phase, pipelined process. The first phase is *Content Determination*. It is where Multiface determines what content should be presented to the user, given his skills and content goals. Then, *Modality Allocation* determines what modality should be used to present each unit of content. (This phase may also revise the content selections made during the first phase to better suit the modality constraints of the user and the device.) Finally, *Modality Realization* identifies how to combine these modalities for presentation on a specific device. The entire process is represented declaratively as a set of plans expressed in Huber's [22] JAM framework.[2]

When the user selects a topic, Multiface instantiates and executes a top-level plan. This plan will reinitialize the interaction (triggering a plan to assert a set of interaction constraints based on the the modality preferences indicated by the user and device models and a plan to clear out previously presentation goals) and then it will initiate content determination, modality allocation, and modality realization (layout), which are each represented by separate plans.

9.1 Adaptive Content Determination

The selection of content involves choosing segments from the master document (or generating new segments) and establishing a linear ordering for presentation to the user. The result of the content determination phase will be a set of facts about which segments to present. We call these facts **the content goal list**.

Multiface performs content determination by first identifying the normative content items (which includes no adaptations) and then adapting the original content on the basis of the user.

[2]Each of these phases is represented by about 12 planning rules and, thus for brevity, the rules are not shown here. For a complete discussion of the rules, see Channarukul [12].

Multiface's approach to adaptive selection of content is based upon the approach used in Equuleus [29]. Equuleus adapts the content to individual users, based on their level of expertise, what sections they have already read, and how they have performed on related tests.

Normative content is determined on the basis of the user's primary selection of a topic. Prior to the topic selection, the document from which content is drawn will have been automatically converted into a set of plans. For each document constituent, there is a static plan to generate its constituents, following the hierarchical structure and linear ordering of the document. When the plan to select normative content for a topic is initiated, the execution of the plan will assert that each of the constituents associated with this topic are included in the normative content. The normative content can be both extended and abbreviated depending on the user and device.

One type of addition is based on the rhetorical (semantic) links among constituents of the document. When a constituent is added to the normative content, we may decide to add additional constituents based on the links and general information known about the user. Following Equuleus, a pedagogy for a given document may specify that certain types of links should be considered for particular classes of users. Thus, after a constituent is added to the normative content, if there is a rhetorical link from this constituent to an additional constituent not directly included in the plan and that link is among those applicable to this user, then the additional constituent will be added to the content.

Multiface will also pull in segments outside the current scope (for example from another lesson or a glossary) if there is a valid semantic link to it from a segment that is currently in the normative content.

A third type of addition is based on the users skills and the prerequisite structure associated with the master document. When a document constituent has a particular skill prerequisite and the user lacks that prerequisite then a constituent is added that will provide the prerequisite skills. This is specified as two related plans, the first plan traverses the list of segments of the normative content and invokes the plan for posting its prerequisites. The second plan checks that there is a prerequisite skill that the user lacks, and if so adds a segment that will provide the skill to the normative content.

Multiface will remove content items that the user is already believed to know. Associated with each document constituent is a set of skills that aims to achieve. When the user has these skills, as indicated by the user model, we assume that the constituent may be omitted.

After the normative content has been adapted then the final step is to expand each of the content plans (which may include composite structures such as, lessons or units, so that the final plan is a linear sequence of document segments (the smallest granularity of document object that can be realized).

9.2 Modality Allocation

The goal of the modality allocation phase is to create a **modality goal list** that enhances the content goal list This goal list specifies what modality or a combination of modalities are selected for each segment. created during the previous phase of generation. This task involves specifying what modalities should be used to present each content segment. It may also involve replacing some content with new content that better meets the constraints of the user and the device.

The modality allocation process is as follows: First, Multiface clears the modality goal list and the applicable segment list from its knowledge base. Then, Multiface finds all segments that are applicable to the user and the current device (*i.e.,* those segments that satisfy the interaction con-

straints). After this, there may still be choices among allowable modalities. In this case, Multiface chooses from those that are preferred by user, device, or document. Finally, after modalities have been allocated for each segment, Multiface then checks for dependencies among segments to remove any segments that are no longer applicable.

When the modality allocation phase is finished, the result is the modality goal list. This goal list and the content goal list will be used in the next phase, multimodal output realization.

10 Multimodal Output Realization

The multimodal output realization phase is the final phase of Multiface's adaptation process. The previous two phases of the multimodal content adaptation process specify what content items should be presented (the content goal list) and what modalities are appropriate for these items (the modality goal list). However, they do not specify how to present specified content on the device.

Multiface divides the realization process into three main steps.

1. First, it plans a presentation layout according to a given content and modality goal list that is suitable for presentation on the device. In this step, selected content items will be grouped into clusters. A **cluster** represents a group of content items with a specific layout type. If the length of the presentation is too long, clusters will be grouped into a series of presentation frames. A **presentation frame** is a spatial container that represents what will be rendered on the device at a time. When creating a frame, more clusters will be added as long as the current frame can still accommodate them. The result of the first step is a presentation layout specified as a **layout goal list**.

2. Second, Multiface's controller constructs from the layout goal list a feature structure that represents the content items and the presentation layout.

3. Finally, Multiface uses a template-based multimodal output generation system, called DOGHED (Dialog Output Generator for HEterogeneous Devices) [14] to generate output from the feature structure created by the previous step.

10.1 Presentation Layout Planning

Multiface selects a layout following three general principles. These principles reflect what we think is necessary for users to understand the presentation.

Principle 1: Avoid conflicts between content items The first principle is the most important one. The generated presentation must not have conflicting content, such as visually overlapping text or images in visual channel of the presentation. Similarly, when multiple auditory content items exist, they must not overlap each other in time. However, some overlapping content is acceptable if one item is primary and the others are secondary. For example, text is considered primary and can be on top of a background picture. Similarly, background music is considered secondary to a narrative audio clip. Multiface uses a content's modality and the relations among them to prevent conflicts among content items.

Principle 2: Minimize the number of presentation frames When there are many content items to be presented, the screen of the device might not be able to accommodate all of

them simultaneously. Typical systems will present all items at once, therefore items that are beyond the bottom part of the screen will not be visible immediately. The user will need to scroll down to see them. Although scrolling is generally acceptable for lengthy content presentation, it may not work as well on small devices as it does on a desktop computer. When the presentation is lengthy, Multiface will attempt to divide it into smaller frames. Only one frame will be presented at a time. A few buttons will be available for the user to browse back and forth between frames or restart the presentation from the beginning.

Principle 3: Present highly-related items simultaneously Since presentations might be divided into multiple frames according to the previous principle, it is possible that some items that are highly related will be on different frames. However, an image and its caption should always occur in the same frame. Multiface will attempt to put such items together such that if some items have to be on another frame, all related items will also be in that frame.

Multiface's presentation layout process is based on the framework established by Vernier and Nigay [50] that synthesizes several multimodal output combination types based on earlier frameworks [15, 16, 27]. Vernier and Nigay's framework integrates two existing multimodal output combination frameworks; the combination aspects and the combination schemas. The result is a unified framework for combining multimodal outputs where there are five alternatives for each combination aspect (see Table 3). Therefore, a single combination will have four characteristics in this framework.

	Schema 1	Schema 2	Schema 3	Schema 4	Schema 5
Temporal	Anachronism	Sequence	Concomitance	Coincidence	Parallelism
Spatial	Separation	Adjacency	Intersection	Overlaid	Collocation
Syntactic	Difference	Completion	Divergence	Extension	Twin
Semantic	Concurrency	Complementary	Complementary & Redundancy	Partial Redundancy	Total Redundancy

Table 3: Combination aspects and schemas in Vernier and Nigay's framework.

The **combination aspects** define how multimodal content can be combined in four different aspects as follows:

- **Temporal combination** defines how multiple modalities are presented at various points in time. For example, two modalities are combined *sequentially* if one is temporally rendered right after the other. They are combined *parallelly* if both are rendered at the same time.
- **Spatial combination** occurs when two or more modalities share the same presentation space (*e.g.,* screen, speaker). For example, two modalities are combined *adjacently* if both are rendered close to each other on the screen. When both are rendered on the same location on the screen, they are considered to be *collocated*.
- **Syntactic combination** concerns the logical form of a modality. For example, if two different modalities are combined (*e.g.,* text and image), they are considered syntactically *different*.
- **Semantic combination** focuses on the meaning of the conveyed information with such modalities. For example, when two modalities that convey the same information are presented simultaneously, they are considered to have *total redundancy*.

The **combination schemas** are the second part of Vernier and Nigay's framework. They define how to combine those modalities (see the columns' header of Table 3):

- **Schema 1** corresponds to distant modality combination where the modalities do not overlap each other.
- **Schema 2** corresponds to modality combination with one point of contact where one modality begins at the end point of the previous modality.
- **Schema 3** corresponds to modality combination with a non-empty intersection where modalities overlap but neither is contained within the other.
- **Schema 4** corresponds to modality combination with inclusion where one modality is wholly contained within the other modality.
- **Schema 5** corresponds to modality combination with the same characteristics where two modalities begin at the same point and also end at the same point.

When two multimedia objects are combined, different schemas might be used in each combination aspect. For example, when an image and its caption are combined. The following combination schemas are used:

- Schema 5 is used in the temporal aspect because they appear at the same time and duration (temporal-parallelism),
- Schema 2 is used in the spatial aspect because an image is adjacent to its caption (spatial-adjacency),
- Schema 2 is used in the syntactic aspect because a caption completes an image (syntactic-completion), and
- Schema 2 is used in the semantic aspect because a caption complements an image (semantic-complementary).

Table 4 presents other sample modality combinations in temporal and spatial aspect. We use the combination aspect and schema information to determine the temporal and spatial layout of Multiface's multimodal presentations.

Modalities	Temporal Aspect	Spatial Aspect
texts	Schema 5 (Parallel)	Schema 1 (Separation)
image + caption	Schema 5 (Parallel)	Schema 2 (Adjacency)
text with reference image	Schema 5 (Parallel)	Schema 2 (Adjacency)
image + audio	Schema 4 (Coincidence)	-
image + audio + gesture	Schema 4 (Coincidence)	Schema 4 (Overlaid)
multiple images (related)	Schema 5 (Parallel)	Schema 2 (Adjacency)
text + audio	Schema 4 (Coincidence)	-
text + formatting (*e.g.,* bullets)	Schema 5 (Parallel)	Schema 5 (Collocation)
multiple images (slide show)	Schema 5 (Parallel)	Schema 2 (Adjacency)
text + speech synthesis	Schema 3 (Concomitance)	-

Table 4: Examples of modality combinations and their temporal and spatial aspect.

Figure 13 shows an example layout of a complete presentation, divided into a sequence of frames. Within individual frames there are combinations of a text and an audio (Cluster L2 in Frame 1),

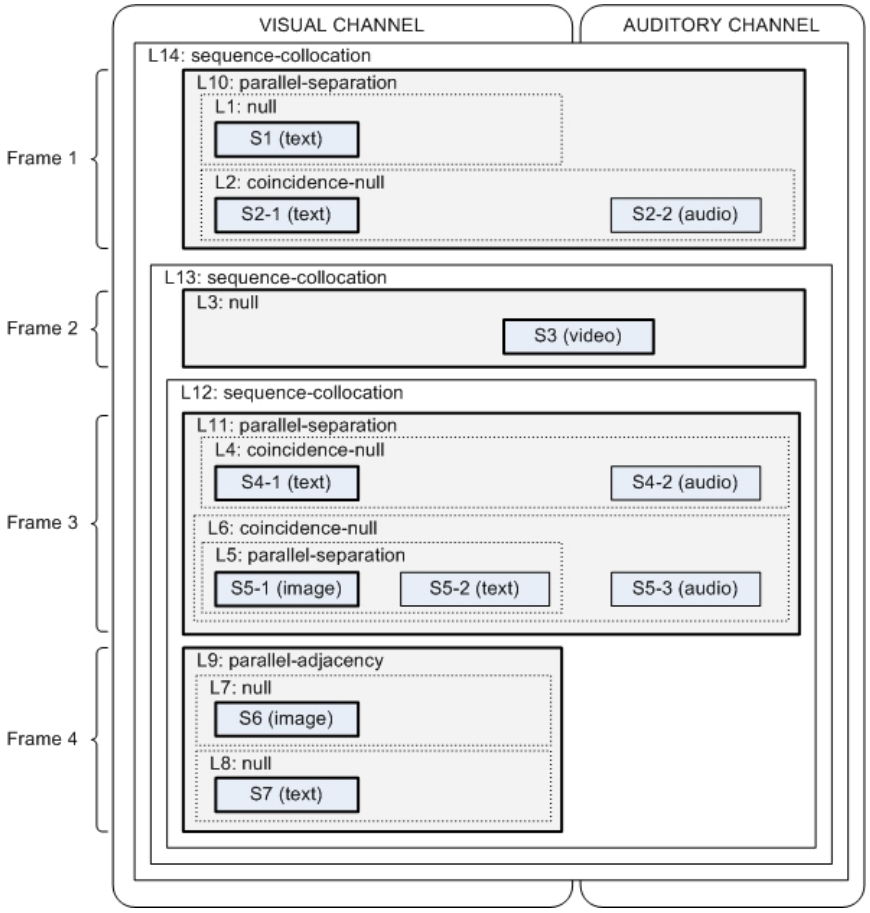


Figure 13: Diagram of layout goals for a series of clusters.

two texts (Cluster L10 of Frame 1), and an image and a caption (Cluster L9 in Frame 3), which use the coincidence-null, the parallel-separation, and parallel-adjacency layouts, respectively.

The result of layout planning will be a set of layout goal predicates that specify how each group of related segments should be combined. Figure 14 shows the layout goals that correspond to the layout shown in Figure 13.

```
// Segment S1
content-layout("Frodo", "O99", "L1", "S1", "null", "null", "null")
// Segment S2-1 and S2-2
content-layout("Frodo", "O99", "L2", "S2-1", "S2-2", "coincidence", "null")
// Segment S3 (Frame 2)
content-layout("Frodo", "O99", "L3", "S3", "null", "null", "null")
top-layout("Frodo", "O99", "L3", "S3", 180)
// Segment S4-1 and S4-2
content-layout("Frodo", "O99", "L4", "S4-1", "S4-2", "coincidence", "null")
// Segment S5-1, S5-2, and S5-3
content-layout("Frodo", "O99", "L5", "S5-1", "S5-2", "parallel", "separation")
content-layout("Frodo", "O99", "L6", "L5", "S5-3", "coincidence", "null")
// Segment S6
content-layout("Frodo", "O99", "L7", "S6", "null", "null", "null")
// Segment S7
content-layout("Frodo", "O99", "L8", "S7", "null", "null", "null")
// Frame 1
content-layout("Frodo", "O99", "L10", "L1", "L2", "parallel", "separation")
top-layout("Frodo", "O99", "L10", "S1", 100)
// Frame 3
content-layout("Frodo", "O99", "L11", "L4", "L6", "parallel", "separation")
top-layout("Frodo", "O99", "L11", "S4-1", 250)
// Frame 4
content-layout("Frodo", "O99", "L9", "L7", "L8", "parallel", "adjacency")
top-layout("Frodo", "O99", "L9", "S6", 170)
// A series of clusters
content-layout("Frodo", "O99", "L12", "L11", "L9", "sequence", "collocation")
content-layout("Frodo", "O99", "L13", "L3", "L12", "sequence", "collocation")
content-layout("Frodo", "O99", "L14", "L10", "L13", "sequence", "collocation")
```

Figure 14: Layout goals for a series of clusters.

10.2 Construction of a Feature Structure

The second step in the realization process is to create a feature structure representation from the presentation layout specified in the layout goal list. A feature structure is a domain-independent input format required by our multimodal output generation system (DOGHED).

Figure 15 shows an example feature structure for Cluster `L12` (`cluster`) that uses the *sequence-collocation* layout (specified in the `template` feature). This cluster contains two frames (Clusters `L11` and `L9`). The visual components of each frame are given in the `content` feature of the `seq-container` template. Here, the first frame (Cluster `L11`) is the first feature structure specified in the `visual` feature. It uses the *parallel-separation* layout. The value for its `first` and `second` features is a feature structure for Clusters `L4` and `L6` respectively. The second frame (Cluster `L9`) is specified next in the `visual` feature. It uses the *parallel-adjacency* layout. Visual components of

this frame (Clusters `L7` and `L8`) are also specified in the `first` and `second` features of the feature structure for this frame respectively. In this cluster, two auditory segments (Segment `S4-2` and `S5-3`) previously in the `auditory` feature of Cluster `L11` have been moved to the `auditory` feature of Cluster `L12`.

```
((template "sequence-collocation")
 (library "xhtmlsmil")
 (cluster "L12")
 (visual ( ((template "seq-container")
            (library "xhtmlsmil")
            (content ((template "parallel-separation")
                      (library "xhtmlsmil")
                      (cluster "L11")
                      (first ...L4's feature structure...)
                      (second ...L6's feature structure...)) ))
           ((template "seq-container")
            (library "xhtmlsmil")
            (content ((template "parallel-adjacency")
                      (library "xhtmlsmil")
                      (cluster "L9")
                      (first ...L7's feature structure...)
                      (second ...L8's feature structure...)) )) ))
 (auditory ( ((template "audio")
              (library "xhtmlsmil")
              (cluster "L4")
              (parent-cluster "L11")
              (id "S4-2")
              (source "audio/audio4.wav")
              (duration "25")
              (begin ?begin)
              (end ?end))
             ((template "audio")
              (library "xhtmlsmil")
              (cluster "L6")
              (parent-cluster "L11")
              (id "S5-3")
              (source "audio/audio5.wav")
              (duration "15")
              (begin ?begin)
              (end ?end)) ))
)
```

Figure 15: Feature structure for Cluster `L12` (using the *sequence-collocation* layout).

Once the feature structure for the presentation is created, the `auditory` feature of the outermost feature structure will contain every auditory segment of the presentation. Each of them will have a `parent-cluster` feature specifying what segment cluster it belongs to at the frame level. The controller evaluates each feature structure in this list to specify a value of its `begin` and `end` features. The values for these features will be strings that specify the beginning and ending of auditory content in SMIL's syntax. The value of the `begin` feature will be the parent cluster's ID (taken from the `parent-cluster` feature) followed by the string "`.begin +`" and the value of the time it takes to play back the first auditory content of this frame up to the current auditory

content. It is calculated from the total duration of all auditory content since the start of this frame plus a one second gap between each of them. Once a new parent cluster's ID is encountered, the time span's value will be reset to zero and started accumulating again for the new frame. The value of the `end` feature will always be the parent cluster's ID followed by the string "`.onend`". These values are required to generate a synchronized audio in SMIL output.

```
(auditory ( ((template "audio")
             (library "xhtmlsmil")
             (cluster "L4")
             (parent-cluster "L11")
             (id "S4-2")
             (source "audio/audio4.wav")
             (duration "25")
             (begin "L11.begin + 0")
             (end "L11.onend"))
            ((template "audio")
             (library "xhtmlsmil")
             (cluster "L6")
             (parent-cluster "L11")
             (id "S5-3")
             (source "audio/audio5.wav")
             (duration "15")
             (begin "L11.begin + 26")
             (end "L11.onend")) ))
```

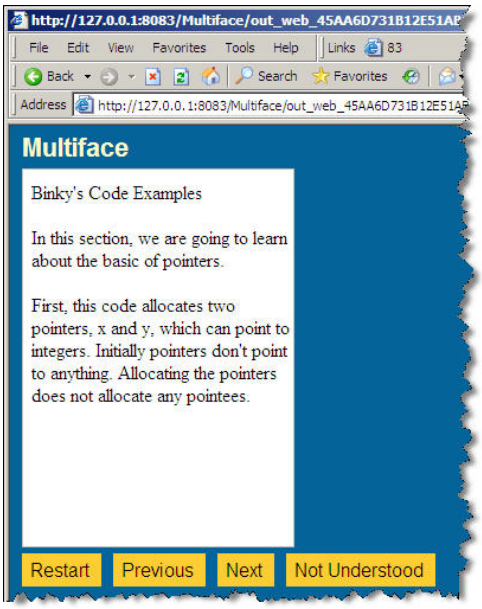
Figure 16: Updated auditory feature of Cluster `L12`.

Figure 16 presents an updated auditory feature of Cluster `L12` given in Figure 15. Here, Segment `S4-2` will be played back when the frame containing Cluster `L11` is rendered. This audio will be played for 25 seconds. However, if the user changes the frame, this audio will stop immediately. The audio of Segment `S5-3` will be played 26 seconds after the frame of Cluster `L11` starts. It also ends immediately if the frame being presented is changed.

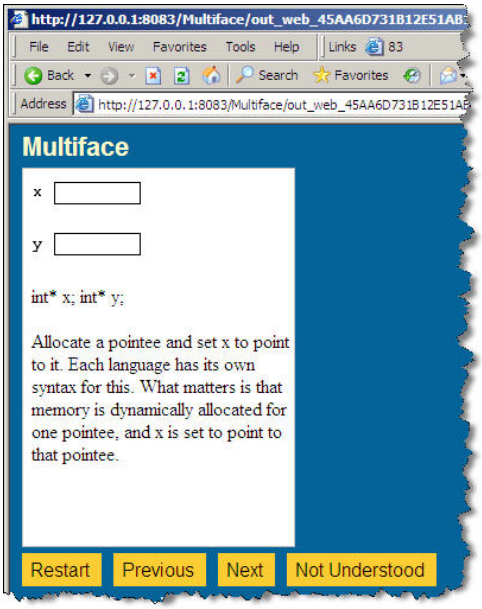
This feature structure will then be passed to a separate, template-based multimodal generator, DOGHED [14]. Templates of DOGHED specify expressions in the SMIL or XHTML+SMIL format, that are instantiated with values given by the feature structure. These SMIL-based expressions are then realized as presentations on the user's display. Figures 17 and 18 show two alternative presentations produced by Multiface for a desktop computer and a PDA, respectively.[3]

[3]The information in the examples is based on a curriculum that was developed by Nick Parlante at Stanford University and is freely available over the Internet. This material may be copied and redistributed so long as the standard Stanford CS Education Library notice on the first page is retained: "This is document 106 in the Stanford CS Education Library. This and other free materials are available at cslibrary.stanford.edu."

Figure 17: A video from a presentation on a desktop computer



The first frame on a PDA



The second frame on a PDA

Figure 18: Two frames from a presentation on pointers on a PDA

11 Related Work

While there are several other systems that adapt to the user and a few that adapt to the device, to our knowledge, very few systems address both, and those that do in only a very limited way (that does not consider the user's apparent understanding.)

11.1 User-centered Adaptations

User-centered adaptations focus on adjusting to the user's preferences, history of interaction with the system, background knowledge of the domain, or other characteristics such as age or job classification. Some, such as [44] take a combination of these factors into account.

Following [25], user-centered adaptations can be classified into three types: 1) adaptations of content 2) adaptations of presentation and modality; and 3) adaptations of structure.

Adaptations of Content In the generation of text and speech, adaptations of content concern the selection of information, the ordering of information, the use of specific rhetorical structures [39], syntactic structures [2], vocabulary [42], or referring expressions [43]. Hypertext and hypermedia systems introduce additional sets of techniques such as page variants (where authors provide different versions of complete pages which are selected on the basis of a user profile), fragment variants or conditional text (which allows for the selection of individual paragraphs or images), fragment coloring (which uses color to indicate the relevance or appropriateness of different parts of the text to the user) and adaptive stretch-text (where the system extends or collapses text when the user clicks on it) [9, 8, 25, 49]. Some adaptive hypertext systems use natural language generation techniques to generate *Dynamic Hypertext* for a page on-the-fly rather than relying on pre-authored texts [17]. These techniques range from simple text templates where a system fills template slots at generation time, to more sophisticated techniques that involve content and sentence planning at a deeper level. Systems that use this technique include IDAS [44], ILEX [24, 31], and PEBA-II [32, 33].

Adaptations of Presentation and Modality These adaptations concern changes to the format and layout of content fragments and the selection of output modalities such as text, speech, or graphics. Systems such as COMET [18], WIP [52], and PPP [1] produce a combination of coordinated modalities to improve generated explanations. AVANTI [20] is a multi-modal tour guide that takes the user's physical abilities into account; for example when a blind person uses the system, audio directions, rather than a visual map, are provided.

Adaptations of Structure To date, adaptations of structure have been considered primarily among authors of hypermedia. Techniques for adaptation of structure include link sorting, link annotation, link hiding or unhiding, and link disabling or enabling. Most of these have analogs in text generation or hypermedia as adaptations of content (*e.g.,* text ordering, text selection or fragment coloring). For example, ELM-ART II [53] marks links to indicate whether a student is ready to follow a particular link or not; the system updates the annotations as students learn more of the material; this is similar to fragment coloring, but the meaning of the annotation is made explicit.

11.2 Device-centered Adaptations

Device-centered adaptations address the fact that different devices have different resources such as screen space, processing power, storage space, network bandwidth, and multimedia capabilities. Existing device-centered adaptations are primarily low-level and involve the trans-coding of multimedia objects to better meet the requirements of the user's device. For example, staying within a single medium, one can translate from GIF to JPEG or from color to gray-scale. One can also translate across media, for example by going from audio to text or from video to still images.

Vetro and Sun [51] distinguish three types of multimedia conversions: 1) syntax conversion, 2) bit-stream scaling, and 3) modal conversion. **Syntax conversions** provide multimedia objects that conform to the formats expected by particular devices. For example, mobile devices typically expect content to be presented using MPEG-4, while desktop machines more often use MPEG-2. **Bit-stream scaling** allows one to reduce the resolution of images sent to mobile devices to suit their smaller screen sizes. This technique is also called *down sampling* or *distillation* of multimedia content. Systems that include this approach include Digestor [4] and Pythia [21]. Another approach is to use text summarization to reduce the space requirements of text [10]. **Modal conversions** are those that translate across modalities such as going from audio to text. For example, the Content Adaptation Pipeline (CAP) [41] performs multimedia trans-coding at run time on so-called middle-ware servers to reduce the workload of the application server. InfoPyramid [34], prepares all necessary conversions off-line, before users request them.

11.2.1 Models of Multimedia Documents

Models of multimedia documents allow one to specify the spatial or temporal relationships among document components and sometimes also interaction capabilities. However, among multimedia document models, the notion of interaction is more limited than in dialog systems. In this context, interaction capabilities include navigational interactions to select different objects connected by hyper-links and interface-control interactions to adjust parameters such as audio volume, font size, or image size. There is no notion of a discourse model or interaction history which would allow the system to reason about past interactions. Current multimedia document models include Hypertext Markup Language (HTML), HTML+TIME, the Synchronized Multimedia Integration Language (SMIL), the Hypermedia/Time-based Structuring Language, HyTime [36], TDAO model [35], and the ZYX model [5].

11.3 Nearly-Hybrid Adaptations

There are some systems that take into account limited aspects of the user and the device, which we will refer to here as hybrid systems. These include Hippie [38], an Internet-based museum guide, Cuypers [48], a general-purpose multimedia transformation engine, REAL [3], a personal navigation system. and AMACONT [19], a system for producing dynamically assembled web presentations. However, none of these systems combines information about the user, the content and the device to the extent that Multiface does. Moreover, none of these systems models the user's current understanding and only one (AMACONT) would allow appear to allow users to migrate from one device to another while continuing the same task, because the rest do not reassess the user and device models once the initial presentation has been created.

Although Hippie can be used either from a personal computer (to prepare a visit to a museum) or from a mobile device (to guide a user during their visit), Hippie really implements two separate

tasks and does not adapt either of them to a different device. It also does not adapt to a user's understanding, as in Multiface, but it will adapt to the user's location and direction within the museum, along with a set of fixed preferences, that were set during the trip preparation phase.

Cuypers produces presentations with layouts that have been adapted to the user's layout-related preferences and to the constraints of the device. Like Multiface it makes use of rhetorical structure to link content units, in this case user preferences can be specified in terms of the the layout of these relations. Unlike Multiface, there is no model of the user's understanding, and the selection of content itself is not adapted.

REAL runs on a variety of devices and adapts to limited device resources, such as the availability of technologies to detect the user's position (*e.g.,* the Global Positioning System or infrared cameras) and the supported modalities of the output device. Its adaptations to the user are limited to the location of the user, their presumed familiarity with the environment and whether they are believed to be rushed or distracted. A central planner uses the values of these variables to generate the most appropriate set of directions: When the output device supports graphics, the system will present maps, otherwise it will produce detailed instructions. When the user is burdened, rushed, or unfamiliar with the environment, it will select easy to follow routes, instead of the shortest ones. REAL is thus a presentation system that takes into account some aspects of the user and the device.

AMACONT creates dynamically assembled web presentations and is the most adaptive of the existing hybrid systems. It accounts for both the capabilities of the device and to a model of the user. The model of the user considers static user properties such as age, gender and general knowledge level, as well as layout-related preferences such as preferences for the relative size of text and images, that may vary over time. A unique feature of this work is that these preferences are learned automatically, on the basis of the user's interaction with the system. (It is a goal of our future work to be able to learn patterns of user preferences.) AMACONT has a much richer notion of document layouts than Multiface; however unlike Multiface (which includes a number of document independent layouts), in AMACONT, alternative layouts are specific to a given document and must be specified by the author of the document in terms of conditionalized expressions parameterized by the values in the user and device models.

12 Summary

The multimodal content generation process presented in this paper integrates both user-centered and device-centered adaptations. Constraints from both the user and the device are unified into a single set of constraints, called **interaction constraints**. The adaptive content determination phase takes into account information about the user and the domain to build an initial list of content goals. The modality allocation phase constructs a list of modality goals based on the interaction constraints. The multimodal output realization phase uses information about the device and the domain to determine the layout of the presentation. The result is a combination of content, modality, and layout that satisfies both the user and the device.

The approach is unique, both because it dynamically accounts for the user, the content and the device and because it represents the entire process declaratively, as facts and rules in a generic planning framework, JAM [22]. The cost for the increased flexibility of our approach is in the amount of knowledge that must be added to support its adaptiveness. Our work represents a balance between the fine-grained adaptiveness of systems that generate content from conceptual representations (but which are necessarily limited in scale) and the more coarse-grained adaptiveness of adaptive navi-

gation systems or modality-transforming systems, (but which do not have a rich model of the user or the content). In Multiface, the master document minimally requires a markup of the subsections of a document (which can be recognized automatically) and of the semantic relations among those subsections. The latter task can be facilitated by GUI-based tools that we have written, however it is a goal of our future work to improve these tools, as current users within our group still prefer to create the markup by hand. Additionally an author can specify what skills, if any, each document component provides or requires and when during the interaction, tests of understanding should be given. The author can also specify the content of those tests, or allow the system to generated them on the fly.

Our current work involves taking our system into the classroom to help validate and revise the user models that we have chosen. In the long term, we hope to be able to learn more of the user model information from users' interaction with the system.

References

[1] Elisabeth André, Thomas Rist, and Jochen Müller. Guiding the user through dynamically generated hypermedia presentations with a life-like character. In *Proceedings of the 1998 International Conference on Intelligent User Interfaces (IUI'98)*, pages 21–28. ACM, 1998.

[2] John Bateman and Cécile L. Paris. Phrasing a text in terms the user can understand. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1511–17, 1989.

[3] Jörg Baus, Antonio Krüger, and Wolfgang Wahlster. A resource-adaptive mobile navigation system. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 15–22, San Francisco, California, USA, 2002. ACM Press.

[4] Timothy W. Bickmore and Bill N. Schilit. Digestor: Device-independent access to the World Wide Web. *Computer Networks and ISDN Systems*, 29(8–13):1075–1082, 1997.

[5] Susanne Boll and Wolfgang Klas. ZYX - a semantic model for multimedia documents and presentations. In *Proceedings of the 8th IFIP Conference on Data Semantics (DS-8): Semantic Issues in Multimedia Systems*, 1999.

[6] Susanne Boll, Wolfgang Klas, and Utz Westermann. A comparison of multimedia document models concerning advanced requirements. Technical Report 99-01, Ulmer Informatik-Berichte, University of Ulm, Germany, 1999.

[7] M. Bordegoni, G.Faconti, M. T. Maybury, T. Rist, S. Ruggieri, P. Trahanias, and M. Wilson. A standard reference model for intelligent multimedia presentation systems. In *Proceedings of the IJCAI '97 Workshop on Intelligent Multimodal Systems*, 1997.

[8] Paul De Bra, Peter Brusilovsky, and Geert-Jan Houben. Adaptive hypermedia: from systems to framework. *ACM Computing Surveys (CSUR)*, 31(4es), 1999.

[9] Peter Brusilovsky. Efficient techniques for adaptive hypermedia. In *Intelligent Hypertext: Advanced Techniques for the World Wide Web*, pages 12–30, 1997.

[10] Orkut Buyukkokten, Hector Garcia-Molina, and Andreas Paepcke. Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices. In *Proceedings of The 10th WWW Conference*, Hong Kong, China, May 2001.

[11] http://www.w3.org/TR/CCPP-struct-vocab, Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C, 2004.

[12] Songsak Channarukul. *Multimodal Content Adaptation for Heterogeneous Devices*. PhD thesis, The University of Wisconsin-Milwaukee, 2005.

[13] Songsak Channarukul, Susan W. McRoy, and Syed S. Ali. Enriching Partially-Specified Representations for Text Realization using An Attribute Grammar. In *Proceedings of The First International Natural Language Generation Conference*, Israel, June 2000.

[14] Songsak Channarukul, Susan W. McRoy, and Syed S. Ali. DOGHED: A Template-Based Generator for Multimodal Dialog Systems Targeting Heterogeneous Devices. In *Companion Volume to the Proceedings of HLT-NAACL 2003*, pages 5–6, Canada, May 2003.

[15] Joelle Coutaz, Laurence Nigay, and Daniel Salber. The MSM framework: A design space for multi-sensori-motor systems. In *Proceedings of the East-West International Conference on Human-Computer Interaction (EWHCI)*, pages 231–241, 1993.

[16] Joelle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May, and Richard M. Young. Four easy pieces for assessing the usability of multimodal interaction: The CARE properties. In *Proceedings of Interact'95*, pages 115–120, 1995.

[17] Robert Dale, Jon Oberlander, Maria Milosavljevic, and Alistair Knott. Integrating natural language generation and hypertext to produce dynamic documents. *Interacting with Computers*, 11(2):109–135, 1998.

[18] Steven K. Feiner and Katheleen R. McKeown. Automating the Generation of Coordinated Multimedia Explanations. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces*, pages 117–138, Boston, 1993. AAAI Press/MIT Press.

[19] Zoltan Fiala, Michael Hinz, Geert-Jan Houben, and Flavius Frasincar. Design and implementation of component-based adaptive web presentations. In *Proceedings of the ACM Symposium on Applied Computing (SAC 2004)*, volume 2, pages 1698–1704, Cyprus, 2004. ACM Press.

[20] Josef Fink, Alfred Kobsa, and Andreas Nill. Adaptable and adaptive information provision for all users, including disabled and elderly people. *The New Review of Hypermedia and Multimedia*, 4:163–188, 1998.

[21] Armando Fox and Eric A. Brewer. Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation. In *Proceedings of The 5th WWW Conference*, Paris, France, May 1996.

[22] Marcus J. Huber. JAM Agents in a Nutshell. Technical report, Intelligent Reasoning Systems, November 2001.

[23] L. Jerenic and V. Devedzic. The friendly intelligent tutoring environment, teacher's approach, January 2000. SIGCHI Bulletin.

[24] A. Knott, C. Mellish, J. Oberlander, and M. O'Donnell. Sources of flexibility in dynamic hypertext generation. In *Proceedings of the Eight International Workshop on Natural Language Generation*. Herstmonceux Castle, UK, June 1996.

[25] Alfred Kobsa, Jurgen Koenemann, and Wolfgang Pohl. Personalized hypermedia presentation techniques for improving online customer relationships. *Communications of the ACM, special issue on Adaptive Web-Based System and Hypermedia Systems*, May 2002.

[26] W. C. Mann and S. A. Thompson. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281, 1988.

[27] Jean-Claude Martin. Six primitive types of cooperation for observing, evaluating, and specifying cooperations. In *Proceedings of American Association for Artificial Intelligence Conference*, pages 35–51. Springer-Verlag, 1999.

[28] M. C. McKenna and R. D. Robinson. *An Introduction to the Cloze Procedure: An Annotated Bibliography*. International Reading Association, 1980.

[29] Susan W. McRoy, Syed S. Ali, and Nipat Nalamlieng. Equuleus: Presentation from Legacy Documents. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 589–596, November 2003.

[30] Susan W. McRoy, Songsak Channarukul, and Syed S. Ali. An Augmented Template-Based Approach to Text Realization. *Natural Language Engineering*, 2003. To Appear.

[31] Chris Mellish, Mick O'Donnell, Jon Oberlander, and Alistair Knott. An architecture for opportunistic text generation. In Eduard Hovy, editor, *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 28–37. Association for Computational Linguistics, New Brunswick, New Jersey, 1998.

[32] Maria Milosavljevic and Robert Dale. Authoring on demand: Natural language generation in hypertext documents. In *Proceedings of the First Australian Document Computing Conference)*, Melbourne, Australia, 1996.

[33] Maria Milosavljevic, Adrian Tulloch, and Robert Dale. Text generation in a dynamic hypertext environment. In *Proceedings of the Nineteenth Australasian Computer Science Conference (ACSC '96)*, Melbourne, Australia, 1996.

[34] Rakesh Mohan, John R. Smith, and Chung-Sheng Li. Adapting multimedia internet content for universal access. *IEEE Transactions on Multimedia*, 1(1):104–114, 1999.

[35] Ethan V. Munson and Mark Pfeiffer. A representation of media for multimedia authoring and browsing systems. In *Representations for Multi-Modal Human-Computer Interaction: Papers from the 1998 Workshop*, pages 49–54. AAAI Press, July 1998. AAAI Technical Report WS-98-09.

[36] Steve R. Newcomb, Niell A. Kipp, and Victoria T. Newcomb. HyTime – The Hypermedia/Time-Based Document Structuring Language. *Communications of the ACM*, 34(11), 1991.

[37] Affiliate Faculty of Blood Pressure Measurement Program. *Blood Pressure Measurement Education Program Instructor Manual*. American Heart Association of Wisconsin, Milwaukee, 1998.

[38] Reinhard Oppermann, Marcus Specht, and Igor Jaceniak. A resource-adaptive mobile navigation system. In Hans-W. Gellersen, editor, *Proceedings of the First International Symposium Handheld and Ubiquitous Computing (HUC'99)*, pages 330–333, September 1999.

[39] Cécile L. Paris. Tailoring object descriptions to the user's level of expertise. *Computational Linguistics*, 14(3):64–78, 1988.

[40] Nick Parlante. The stanford CS education library, 1999. `http://cslibrary.stanford.edu/`.

[41] Thomas Phan, George Zorpas, and Rajive Bagrodia. An extensible and scalable content adaptation pipeline architecture to support heterogeneous clients. In *Proceedings of The 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria, 2002.

[42] Ehud Reiter. A new model of lexical choice for nouns. *Computational Intelligence*, 7(4):240–51, 1991.

[43] Ehud Reiter and Robert Dale. A fast algorithm for the generation of referring expressions. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, volume 1, pages 232–38. MIT Press, 1992.

[44] Ehud Reiter, Chris Mellish, and John Levine. Automatic Generation of Technical Documentation. *Applied Artificial Intelligence*, 9(3):259–287, 1995.

[45] Elian Rich. User Modeling via Stereotypes. *Cognitive Science*, 3:355–366, 1979.

[46] Elian Rich. Stereotypes and User Modeling. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*, pages 35–51, Berlin, 1989. Springer-Verlag.

[47] Amanda Stent, Marilyn Walker, Steve Whittaker, and Preetam Maloor. User-Tailored Generation for Spoken Dialogue: An Experiment. In *Proceedings of ICSLP 2002*, September 2002.

[48] Jacco van Ossenbruggen, Joost Geurts, Frank Cornelissen, Lloyd Rutledge, and Lynda Hardman. Towards second and third generation web-based multimedia. In *The Tenth International World Wide Web Conference*, pages 479–488, Hong Kong, 2001.

[49] Richard Vdovjak, Flavius Frasincar, Geert-Jan Houben, and Peter Barna. Engineering semantic web information systems in hera. *Journal of Web Engineering (JWE)*, 2(1-2):3–26, 2003.

[50] Frederic Vernier and Laurence Nigay. A framework for the combination and characterization of output modalities. In *Proceedings of DSV-IS2000, Lecture Notes in Computer Science*, pages 32–48. Springer-Verlag, 2000.

[51] Anthony Vetro and Huifang Sun. Media conversions to support mobile users. In *Proceedings of The IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2001.

[52] Wolfgang Wahlster, Elizabeth André, Wolfgang Finkler, Hans-J'urgen Profitlich, and Thomas Rist. Plan-based Integration of Natural Language and Graphic Generation. *Artificial Intelligence*, 63(1-2):387–427, 1993.

[53] Gerhard Weber and Marcus Specht. User modeling and adaptive navigation support in WWW-based tutoring systems. In Anthony Jameson, Cécile L. Paris, and C. Tasco, editors, *User Modeling*, pages 289–300. Springer-Verlag, 1997.

[54] http://www.w3.org/TR/xslt, XSL Transformation Version 1.0. W3C, 1999.