katashi nagao

digital content annotation and transcoding

DIGITAL AUDIO AND VIDEO SERIES

Digital Content Annotation and Transcoding

For a listing of recent titles in the *Artech House Digital Audio and Video Library*, turn to the back of this book.

Digital Content Annotation and Transcoding

Katashi Nagao

Artech House
Boston • London
www.artechhouse.com

Library of Congress Cataloging-in-Publication Data
Nagao, Katashi.
Digital content annotation and transcoding / Katashi Nagao.
p. cm. — (Artech House digital audio and video library)
Includes bibliographical references and index.
ISBN 1-58053-337-X (alk. paper)
1. File processing (Computer Science). 2. Data structures (Computer Science).
3. Information storage and retrieval systems. 4. Digital media.
5. File conversion (Computer Science). I. Title. II. Series.
QA76.9.F53N34 2003
005.74'1—dc21 2002043691

British Library Cataloguing in Publication Data
Nagao, Katashi
Digital content annotation and transcoding. — (Artech House digital audio and video library)
1. Digital media 2. Multimedia systems
I. Title
006.7

ISBN 1-58053-337-X

Cover design by Christina Stone

685 Canton Street
Norwood, MA 02062

International Standard Book Number: 1-58053-337-X
Library of Congress Catalog Card Number: 2002043691

10 9 8 7 6 5 4 3 2 1

To Kazue

Contents

Preface

Today digital content is no longer limited to exclusive people. In addition to World Wide Web pages, which are the epitome of digital content, there are many more familiar examples such as i-mode for cell phones, photographs taken by digital cameras, movies captured by digital camcoders, compact discs, music clips downloaded from the Internet, and digital broadcasting. Audiences do not have to care if content is digital or not; however, whether they are produced digitally or not means a great deal.

Digitization has made it easier for us to duplicate these products, and as a result, information remains stable. In short, digitization has facilitated the storage of information.

Information storage, however, is not the only advantage of digitization. It is more important to recognize that manipulation (like editing) has become automatic to some degree. This is not to say that information was unable to be stored and edited at all before. Long ago, for instance, some could change the information on a paper, such as a Buddhist scripture, into another form, such as inscription on a stone. In modern times, some painters imitate paintings and add their own originality, and even some ordinary people record TV programs to edit to extract their favorite scenes and reproduce their own programs.

Digital content enables more because it is programmable to manipulate information. Suppose a person consistently records a favorite program. Its first 15 minutes is boring, but the person wants to see the opening. So the person has to record the whole program, from the beginning to the end. If the structure of the TV program is the same, it should be possible to automatically extract the unwanted first 15 minutes by programming. In another example, it would be difficult to extract a specific part from printed material that contains more than one million pages; it would be very easy, however, to extract the necessary part with a program for extracting topics whose content is digitized.

Thus, digital content has largely changed the way of dealing with information. This book describes some experiments of further developing

digital content to make it possible for people to customize and enjoy it more interestingly.

This book focuses on two technologies: transcoding and annotation. Transcoding is a technology to program the transformation of digital content. This enables content to be changed to the most preferable form for audiences. Annotation is a technology that provides additional information to extend content so as to facilitate the programming. Annotation can also play the role of hyperlink, which connects several on-line resources.

The purpose of this book is not only to introduce these two technologies precisely but to propose the system that integrates these two and to prospect the future directions.

I have great expectations for the future technology of digital content for I believe that the digital content produced by a number of folks can motivate people to better understand both themselves and world events so that society at large will change for the better. I also believe that it is the principal responsibility for researchers to produce a technology that maximizes the utilization of the digital content. It might be outrageous to intend to change social systems; however, on the condition that these technologies can facilitate the voluntary actions of many people and can turn them to the right direction, people will come to notice the nature of the issue. As a result, these technologies will put the things that may at first seem impossible into practice.

Acknowledgments

I greatly appreciate the assistance provided by those who contributed to this book. Student researchers, Shingo Hosoya, Yoshinari Shirai, Shigeki Ohira, Ryuichiro Higashinaka, Mitsuhiro Yoneoka, Daisuke Ito, Kevin Squire, Takeshi Umezawa, Toshiki Fukuoka, Yukiko Katagiri, Miki Saito, and Takashi Oguma, with whom I have worked during my time at IBM supported me in developing prototype systems introduced in this book (Chapter 4). Especially, I thank Yukiko Katagiri for her illustrations in this book. The project leader of GDA (presented in Chapter 4), Koiti Hasida, gave me helpful advice to develop the ideas described in this book. Some IBM researchers, Chieko Asakawa, Hironobu Takagi, Hideo Watanabe, Akiko Murakami, and Koichi Takeda, were my good discussants and provided me with some research materials.

I am indebted to my editors, Tiina Ruonamaa and Julie Lancashire, who have been constant mentors during the writing process, and to anonymous reviewers who gave me helpful advice on wording.

Finally, I am particularly grateful to my wife, Kazue Nagao, without whom I could not have undertaken this effort.

1

Introduction: Survival in Information Deluge

Nowadays, information flows around us all the time. Without any means to cope with it, we would certainly be engulfed by this flood. That is because we usually have little time for choosing the necessary information. As a result, not being able to find the necessary information, which has been buried beneath or mixed with irrelevant information, we are often dissatisfied with our choices.

Why has it become this way? And how can we get out of this situation? Even before the Internet boom, information was already in abundance—it was available in print media and on TV. The situation, however, is a bit different now. Most of the information given by TV and print media is in nature unnecessary, and since we have always been, in general, passive watchers/readers, we have not felt the necessity of choosing any of this information. Simply stated, if the information around us is just a mass of irrelevant stuff, which does not remain in mind after viewing, then no one will be bothered by information overload.

On the Internet, however, there is a large mass of information that an individual cannot ignore, and it increases every day at a tremendous rate. With a situation like this, we cannot trust our choices because the information used to make them can get old or useless at any time. In consequence, we have to continue a labyrinthian quest for necessary information until we become confident enough with the results. In the print world, even though there are innumerable books and magazines, we do not become too confused. The reason is, in the case of magazines, we know in advance that the information

they offer can be paged through. As for books, like the weathered classics, we can choose what to read based on their reputations. In short, the biggest problem concerning content on the Internet is the lack of a general framework to let us distinguish something important from something unimportant.

This book focuses on the technical issues and methods for creating that kind of framework. The technologies described will surely be something to be taken for granted in the near future, like the air we breathe but are not very conscious about. To mankind, this technology will be just as indispensable.

1.1 Digital Content Technology

Digital content has rapidly spread all over the world due to the information distribution technologies developed in the twentieth century. Various kinds of technologies to utilize digital content will emerge, and these technologies will make gradual progress in this century. Our challenge is to develop technologies to create and distribute digital content easily. We need to provide a focal point for the development of technologies to utilize digital content more intelligently and with multiple purposes.

My definition of *content technology* covers technologies to create, store, deliver, manipulate, transform, and reuse digital content. This book mainly focuses on manipulation, transformation, and reuse of digital content.

Some typical instances of manipulation and transformation are personalization and adaptation. *Personalization* is the technique to manipulate or transform digital content such as Web pages to conform to user preference. *Adaptation* means a kind of transformation to adjust size or color of content according to constraints and features of user devices such as mobile phones or *personal digital assistants* (PDAs).

I define *transcoding* as the mixture of personalization and adaptation of digital content. Today, people are accessing the Internet not only through personal computers, but via mobile phones, car navigation systems, and many other devices. In such cases, transcoding plays a major role in information access through the Internet. For example, Web pages created for browsing by PCs are not suitable for browsing by mobile phones—automatic resizing of images and automatic summarization of text are required. Transcoding also considers user preferences such as native languages and handicaps. Transcoding has advantages in connection with networks of narrow bandwidth and also in access by people who have various disabilities.

Transcoding also facilitates reuse of digital content. Inherently, content itself is declarative, therefore, versatile. Some documents may provide a story of someone's life, but sometimes we would like to see the information as a biographic dictionary. Transcoding can make this possible. A single content item can have several aspects, and changing its presentation makes it usable in different situations.

My group has been developing tools to annotate content with additional information. Such information contributes to inference of content meanings. Transcoding handles such meanings or attributes of content and bridges content and a user's intentions by transforming content into a user preferred form.

In this book, I propose a concrete plan and method for creation and management of digital content. Based on our recent research results, this will become a most important resource on the Internet. This plan intends to evolve on-line information to ''knowledge'' and develop a machine to augment human intelligence. Many researchers have tried to realize such a machine but have failed. With our machine, we have a means to get a long-awaited human-powered *intelligence amplifier* due to rapid growth of information technology and infrastructure like the Internet. I am confident that we can have a common global knowledge system based on annotated digital content if we are persistent in achieving the plan. Before explaining the precise plan, I will describe some of the problems of present on-line digital content.

1.2 Problems of On-Line Content

Currently, there are the following problems in on-line content such as Web pages written in *Hypertext Markup Language* (HTML):

1. There have been many efforts on visual aspects of document presentation using HTML tags and style sheets. While their tags define parts of syntactic structure of documents, their meanings or semantics are not defined in the current architecture.

   Tim Berners-Lee, director of the World Wide Web Consortium, has proposed his vision about a new architecture of the Web called the Semantic Web that can specify semantics of Web content [1]. He explained that in the context of the Semantic Web, the word ''semantic'' means ''machine processable.'' He explicitly ruled out the sense of natural language semantics. In his vision,

the semantics of Web content convey what a machine can do with that content. They will also enable a machine to figure out how to convert it. His vision is good, but since the semantics of documents necessarily involve semantics of natural language or human thought, we have to consider deeper semantics of content.

2. The structure of current hypertext is very simple and static so that many people can easily create it. From the dynamic nature of the Web, however, hyperlinks are not always valid and very hard to keep consistent. Furthermore, only the author or other authorized persons can update descriptions of the hyperlinks. In order to make hypertext more flexible, authoring of content and linking between content objects should be individually managed.
3. In the case of conventional books, there are skilled people, usually editors or publishers, who evaluate content of the books, negotiate changes in writing or subjects with the authors, and mediate between the authors and the readers. On the contrary, the authoring of Web content seldom has such a system.

Of course, the World Wide Web provided us with a flexible and open platform for document and multimedia publication and distribution. However, it is very hard to automatically transform current Web content into more convenient forms. In this book, I discuss an additional mechanism for the Web that makes current Web content more intelligent and accessible. Another important keyword in this book is *annotation*.

1.3 Extension of Digital Content

Traditionally, Web content has been created using only human-friendly information. Of course, this makes sense—after all, humans are the intended users of the information. However, with the continuing explosion of Web content—especially multimedia content—I would like to argue that content should be created in a more machine-friendly format—one that allows machines to better understand and process documents. My group proposes a system to annotate documents externally with additional information in order to make them easier for computers to process.

Why would we want to make content more machine-friendly? Well, annotated documents are much easier to personalize than the nonannotated variety. For example, a computer can process textual information annotated with parts of speech and word-senses much better than plain text, and can

therefore produce a nice, grammatically correct, personalized summary, formatted for a cellular phone, or can translate a document from English to Japanese. Normally when dealing with nonannotated text, transcoding to many different formats requires much task-specific effort for each format. By using document annotation, however, content providers put in some extra work early on but receive the benefits of being able to transcode to an endless variety of formats and personal tastes easily, thus reaching a much wider audience with less overall effort.

Annotation is not new, and there have already been several attempts to add it to the Web. One of these was the controversial (and now defunct) ThirdVoice [2], allowing Post-it note style adornment (some might say defacement) of pages. Our annotation is different from the others, because we use annotation to mean helpful information for machines to automatically process the semantics of content. My group has been developing an easy and simple method for constructing a superstructure on the Web, based on external annotations to Web documents. The word ''external'' indicates that the annotations are not embedded in the content itself but linked with the content. Annotated documents are easier for computers to understand and process, allowing personalized content to be created with much less effort and greater quality. This permits content providers to reach a much wider audience with minimal overhead.

In this book, I specify three categories of annotation. One is linguistic annotation, which helps the transcoder understand the semantic structure of textual elements. The second is commentary annotation, which helps the transcoder manipulate both textual and nontextual elements such as images and sounds. Commentary annotations are also effective to evaluate target content like book reviews. The third category is multimedia annotation, which is a combination of the first two types.

My group has also developed a system for semi-automatic and interactive Web document annotation, allowing users to annotate any element of any Web document with additional information. We have also developed a proxy server that transcodes requested contents using information from annotations assigned to them. All types of annotation are described using *Extensible Markup Language* (XML), which is a standard for interoperable data exchange systems. The correspondence between annotations and elements of content is defined using *Uniform Resource Locators* (URLs) and *XML Pointer Language* (XPointer) [3].

The entire process is called *semantic transcoding*, because it provides means for easily transcoding annotated documents using information about their deep semantic content [4]. The current semantic transcoding process

handles mainly text and video summarization, language translation, and speech synthesis of documents containing text and images.

Annotation is also useful for knowledge discovery from content. Using this idea, we are also developing a system that discovers knowledge from Web documents and generates a document that includes the discovered knowledge and summaries of multiple documents related to the same topic.

To better visualize these ideas, consider the following: The conventional Web structure can be thought of as a graph on a plane. We are proposing a method for extending such a planar graph to a three-dimensional structure consisting of multiple planar layers. Such a metalevel structure is based on external annotations on digital content on the Web. Figure 1.1 represents the concept of our approach.

As shown in the figure, our idea of a Web superstructure consists of layers of content and metacontent. The bottom layer corresponds to the set of raw content. The second layer corresponds to a set of metacontent about content of the first layer. We generally consider such metacontent as external annotations.

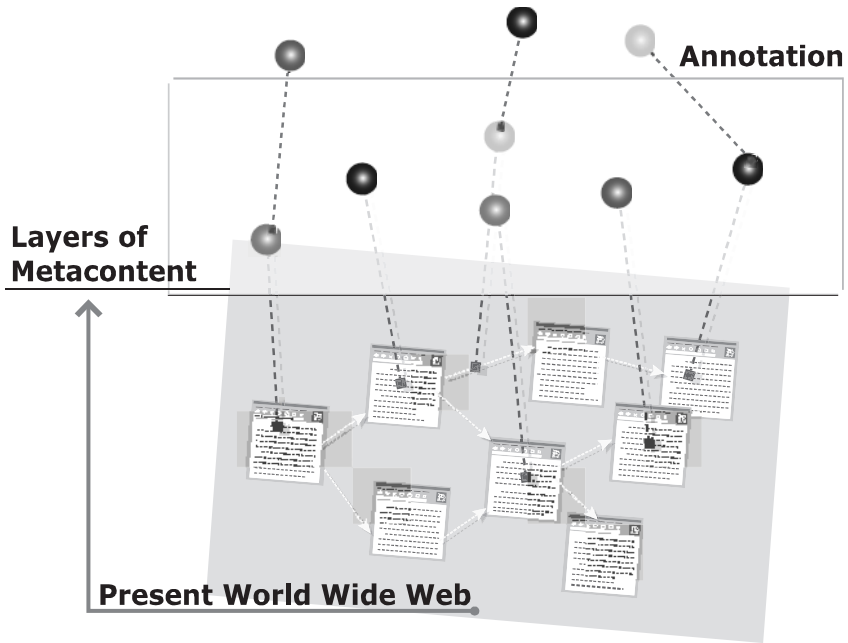


Figure 1.1 Superstructure on the Web.

A popular example of external annotation is comments or notes on Web content created by people other than the author. This kind of annotation is useful for readers evaluating the content. For example, images without alternative descriptions are not understandable for visually challenged people. If there are comments on these images, these people can understand the image contents by listening to them via text-to-speech transcoding.

Another example of annotation is flexible external hyperlinks that connect content with conventional knowledge sources such as on-line dictionaries. External links can be defined outside of the set of link-connected content. Such external links have been discussed by the XML community in the context of *XML Linking Language* (XLink) [5].

There is a large number of documents of great diversity on the Web, which makes some of the documents difficult to understand due to the viewer's lack of background knowledge. In particular, if technical terms or jargon are contained in the document, viewers who are unfamiliar with them might not understand their correct meanings.

When we encounter unknown words in a document—for example, scientific terms or proper nouns—we usually look them up in a dictionary or ask experts or friends for their meanings. However, if there are lots of unfamiliar words in a document or there are no experts around, the work of looking the words up can be very time-consuming. To facilitate this effort, we need (1) machine understandable on-line dictionaries, (2) automated consultation of these dictionaries, and (3) effective methods to show the lookup results.

There is an application that consults on-line dictionaries when the user clicks on a certain word on a Web page, then shows the lookup results in a popped-up window. In this case, the application accesses its inner/on-line dictionaries and the consultation process is automated using the viewer's mouse click as a cue. Pop-up windows correspond to the display method.

Other related applications operate in more or less the same way. There are three major problems with this conventional method:

1. Due to the difficulty of word sense disambiguation, in the case of polysemic (having a diversity of meanings) words, applications to date show all possible word sense candidates for certain words, which forces the viewer to choose the correct meaning.
2. The pop-up window showing the lookup results hides the area near the clicked word so that the user tends to lose the context and has to reread the original document.

3. Since the document and the dictionary lookup results are shown in different layers (e.g., windows), other natural language processing techniques such as summarization, translation, and voice synthesis cannot be easily applied to the results.

To cope with these problems, my group proposes a systematic method to annotate words in a document with word senses in such a way that anyone (e.g., the author) can easily add word sense information to a certain word using a user-friendly annotating tool. This operation can be considered as acreation of a hyperlink between a word in the document and a node in a domain-specific ontology.

Our proposed system basically works as illustrated in Figure 1.2. The proxy server in the middle deals with user interactions, content, and meta-content (annotation) retrievals, and consultation and integration of resources. The proxy server also has a role of transcoding of requested content and is called the transcoding proxy. The details of the transcoding proxy are described in the next chapter.

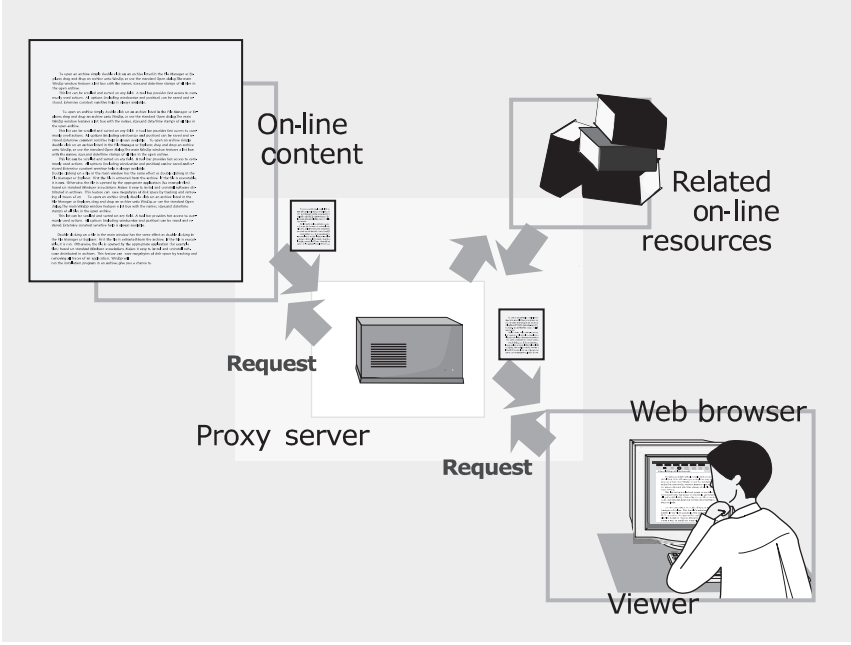


Figure 1.2 Basic configuration of the proposed system.

1.4 Organization of This Book

The rest of this book is organized as follows.

1.4.1 Chapter 2—Transcoding: A Technique to Transform Digital Content

Chapter 2 presents some examples of representation formats or frameworks of digital content such as HTML, *Wireless Markup Language* (WML), *Voice Extensible Markup Language* (VoiceXML), *Synchronized Multimedia Integration Language* (SMIL), and so forth. This chapter also presents comprehensive surveys on content transcoding including transformation, adaptation, and personalization of digital content. Content transformation includes image conversion (change of color, format, resolution), XML-to-HTML transformation by *Extensible Style Sheet Language Transformation* (XSLT), and so forth. Content adaptation covers device-dependent conversion and dynamic content generation from XML databases. Content personalization involves methods for increasing accessibility and usability considering a user's physical and mental aspects. This chapter also explains that transcoding is a key technology in content businesses.

1.4.2 Chapter 3—Annotation: A Technique to Extend Digital Content

Chapter 3 describes several mechanisms to annotate digital content with additional information such as indices, comments, and machine-processable features. Several previous metadata-related works are introduced in this chapter such as Dublin Core Metadata Initiative, *Resource Description Framework* (RDF), and MPEG-7. Authoring systems for annotations such as Annotea, Site Pattern Analyzer, and so forth are also presented. This chapter also discusses several applications made possible or more efficient by annotation, for instance, document and multimedia retrieval, content delivery, and content access control.

1.4.3 Chapter 4—Semantic Annotation and Transcoding: Towards Semantically Sharable Digital Content

Chapter 4 presents the main topic of this book, which integrates topics presented in Chapters 2 and 3. The chapter starts with discussion about semantics of digital content. For example, meaning of words, phrases, and sentences and document structures in the text content are significant constituents of the semantics. Recent activities related to the Semantic Web

are discussed in this chapter. Natural language processing is also an important topic in this chapter. Several techniques and tools for analysis and generation of natural language text are illustrated. Techniques for multimedia processing such as video/audio indexing, visual/auditory object recognition, and multimedia summarization and translation are also explained. This chapter also describes several applications such as semantic transcoding of digital content, knowledge discovery from large volumes of digital content, and restoration of multimedia content.

1.4.4 Chapter 5—Future Directions of Digital Content Technology

Chapter 5 prospects future directions of digital content technology and business, especially content management and distribution based on annotation and transcoding. The important topics of this chapter are how people can overcome the information overload on the Internet and how machines can become intelligent enough to handle essence or meaning of information. The key issues include advanced content management and distribution based on annotations and user profiles, advanced retrieval and reuse of digital content via intelligent software agents, and transcoding-enabled ubiquity of information. This chapter concludes with a discussion about social information infrastructures and their contributions to alteration in our lifestyle.

References

[1] Berners-Lee, T., J. Hendler, and D. Lassila, "The Semantic Web," *Scientific American*, May 2001.

[2] ThirdVoice, "ThirdVoice Home Page," http://www.thirdvoice.com, 2001.

[3] W3C, "XML Pointer Language (XPointer) Version 1.0," http://www.w3.org/TR/xptr, 2001.

[4] Nagao, K., Y. Shirai, and K. Squire, "Semantic Annotation and Transcoding: Making Web Content More Accessible," *IEEE Multimedia, Special Issue on Web Engineering*, Vol. 8, No. 2, 2001, pp. 69–81.

[5] W3C, "XML Linking Language (XLink) Version 1.0," http://www.w3.org/TR/xlink, 2001.

2

Transcoding: A Technique to Transform Digital Content

The main concern of this chapter is adapting or customizing existing digital content according to the user's environments and preferences. Recent advances of Internet technologies and some standards of content representation formats based on XML have created a new opportunity to solve this problem in a more general architecture.

In contrast, the rapid growth of wireless networks and pervasive devices with a wide variety of capabilities and characteristics has provided new constraints on the architecture. In order to be easily carried, the devices must be light and small. This requirement limits the types of user interfaces they can support. Large screens and full keyboards are impossible; a small screen and telephone keypad are more realistic, although some devices may have only a voice interface. In order to run on battery power for useful periods of time, power consumption must be carefully managed, forcing the use of designs with little storage and processing capability. To be connected from anywhere requires wireless connections, which restrict the types of interactions and bandwidth available for accessing content.

People also have a large spectrum of preferences or constraints on information access. Users with some sort of physical or mental disability also create new constraints on content. Considering low vision condition, content should contain larger fonts, enough space between lines and between characters, clear contrast of background and foreground colors, and clear edges of figures. Text-to-voice conversion is very useful for visually challenged people. For hearing-impaired people, auditory content should be changed in

longer time scale or converted into visual content such as text. Considering dyslexia, text content should be paraphrased according to a user's reading capability. In the case of attention and memory deficit disorder, content should be summarized or clarified by showing only its salient segments.

All of these constraints and requirements create difficult challenges in designing a useful system for delivering content to a wide array of devices and people. However, if such an information delivery system could be created quickly and cost-effectively and if it were to integrate with existing information systems, the value to customers would be immense. Transcoding, or adapting content from one form to another, is a key part of satisfying these requirements for rapid, inexpensive deployment of new ways to access existing content.

The media signal processing industry first used the term ''transcoding'' to refer to the task of converting a signal, say a television program, from one format to another—[for example, converting the *National Television System Committee* (NTSC) standard used in America and Japan to the *Phase Alternating Line* (PAL) standard used in much of the rest of the world]—while preserving the content of the program. Although the term has lately been used to mean many different things, here the term refers to the tasks of summarizing or filtering (which modify content without changing its representation) and translating, or converting, content from one representation to another.

The proliferation of wireless devices with very different form factors multiplies the costs associated with Web applications. Different devices require pages to be presented in different markup languages, such as HTML, Compact HTML (including a version called i-mode that is popular in Japan), WML, and VoiceXML. Even with a particular markup language such as WML, different presentations may be required on different devices. For example, for one WML-based phone, choices are most appropriately presented as buttons. For another phone, they might best be presented as hyperlinks. Some forms of content may be much more expensive for a server or network to process than others. The existence of a growing number of client devices with different presentation requirements presents new problems for maintaining Web sites.

In addition, there are business opportunities for network intermediaries, such as *Internet service providers* (ISPs), to make existing Web applications available to a greater market by making them accessible from different devices and people with different preferences. ISPs do not necessarily own the content that they serve to their clients. Instead, they provide value-added services, such as performance improvements via local

caching. Making information available in different forms for different client devices and people using transcoding are additional possible value-added services for them.

2.1 Representation of Digital Content

The term ''application'' is used here to mean any form of program that generates information in response to requests from users; the term ''content'' refers to the information returned in answer to a request, and ''rendering'' means the way the content is presented to the user.

Some markup languages, mostly based on XML, contain content along with tags that identify the meaning of elements in the content. XML is a general markup language so that a designer can decide and extend tag names and their meanings.

Other markup languages, such as HTML and WML, carry both content and rendering instructions, since the tags are primarily concerned with the way the content is presented. Transcoding can modify either the content or the rendering associated with an application. For example, a subset of the content can be selected for a small-screen device, and it can be rendered differently to achieve a richer presentation.

2.1.1 HTML

HTML is a markup language for publishing hypertext on the Web [1]. HTML uses tags such as `<h1>` and `</h1>` to structure a document into headings, paragraphs, lists, hypertext links, and so forth. They are also used for rendering a presentation of the document. A hypertext consists of multiple texts and hyperlinks connecting two of them. A hyperlink is represented by a markup like `<a href="nagao.html">Nagao's page</a>`.

The text between the `<a>` and the `</a>` is used as the caption for the link. It is common for the caption to be in blue underlined text. To link to another Web content, the author needs to give the Web address (commonly called a URL), for instance a link to www.w3.org is written as follows:

```
<a href="http://www.w3.org/">W3C (World Wide Web Consortium)</a>
```

The simple way to add an image to the document is using the `<img>` tag. If the author has an image file called ''nagao_portrait.jpg'' in the same folder/directory as the source HTML file, and it is 200 pixels wide by 150 pixels high, the HTML code for the image would be

```
<img src="nagao_portrait.jpg" width="200" height="150">
```

The src attribute specifies the image file. The width and height are not strictly necessary but help to speed the display of the content. People who cannot see the image need a description that they can read in its absence. The author should add a short description as follows:

```
<img src="nagao_portrait.jpg" width="200" height="150"
    alt="Nagao's face">
```

The alt attribute is used to give the short description, in this case "Nagao's face."

The author can use an image as a hyperlink. For example, the following allows a user to click on the image of the person's face to get to the Web page:

```
<a href="nagao.html"><img src="nagao_portrait.gif"
    width="200" height="150" alt="Nagao's face"></a>
```

HTML tags can also be used for rendering. For instance, the font tag can be used to select the font, its size, and the color. This example sets the size, color, and font:

```
<font size="5" color="red" face="Times New Roman">some text
    ... </font>
```

The size attribute can be used to select the font size as a number from 1 to 7. If a - or + sign is placed before the number, it is interpreted as a relative value. `size="+1"` is used when the author wants to use the next larger font size, and `size="-1"` is used for the next smaller font size. The color attribute is used to set the color and the face attribute to set the font.

Tables are also defined in HTML for rendering data. The author can stretch tables to fill the margins, specify a fixed width or leave it to the browser to automatically size the table to match the contents.

Tables consist of one or more rows of table cells. Here is a simple example:

Name	**Age**
Katashi	40
Kazue	30
Hiroshi	20

The markup for this table is as follows:

```
<table border="1">
    <tr><th>Name</th><th>Age</th></tr>
    <tr><td>Katashi</td><td>40</td></tr>
    <tr><td>Kazue</td><td>30</td></tr>
    <tr><td>Hiroshi</td><td>20</td></tr>
</table>
```

The table element acts as the container for the table. The border attribute specifies the border width in pixels. The `<tr>` element acts as a container for each table row. The `<th>` and `<td>` elements act as containers for heading and data cells, respectively.

Tables are also used for layout. Since HTML itself does not have a method to locate text and images on any specific positions inside the browser window, many HTML authors use table-related tags to create layout-specific content by regarding one total page as a single large table and containing each document element in each table cell. This usage of tables make document analysis difficult for machines to distinguish between data tables and layout tables.

For separation of document styles or layouts and document structures or meanings, style sheets have been introduced in HTML. They are explained in the context of *Extensible Hypertext Markup Language* (XHTML).

2.1.2 XHTML

XHTML is a reformulation of HTML as an instance of XML-based content representation, while original HTML is an application of *Standard Generalized Markup Language* (SGML) [2].

The differences between XHTML and HTML are as follows:

1. *XHTML documents must be well formed:* Well-formedness is a new concept introduced by XML. Essentially, this means that all elements must either have closing tags or be written as a proper empty element (described below), and that all the elements must nest properly.
2. *XHTML element and attribute names must be in lower case:* XHTML documents must use lower case for all element and attribute names. This difference is necessary because XML is case-sensitive—for example, `<p>` and `<P>` are different tags.

3. *XHTML attribute values must always be quoted:* All attribute values must be quoted, even those which appear to be numeric.
4. *Attribute minimization:* XHTML does not support attribute minimization. Attribute-value pairs must be written in full. Attribute names such as compact and checked cannot occur in elements without their value being specified.
5. *Empty elements:* Empty elements must either have an end tag or the start tag must end with `/>`. For instance, `<img src="nagao_portrait.jpg"/>` or `<img src="nagao_portrait.jpg"></img>` are both correct.

In XHTML documents, there must be a DOCTYPE declaration in the document prior to the root element `<html>`. The public identifier included in the DOCTYPE declaration must reference one of the three *document type definitions* (DTDs):

```
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

DTD is a sort of grammar in which XML-based markup languages can be defined. It contains definitions of tag and attribute names, possible attribute values, possible element nesting relations, and so forth. Here is an example of a minimal XHTML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>This is a title</title>
  </head>
```

```
  <body>
    <p>This is a paragraph.</p>
  </body>
</html>
```

Note that in this example, the XML declaration is included. An XML declaration like the one above is not required in all XML documents. XHTML document authors are strongly encouraged to use XML declarations in all their documents. Such a declaration is required when the character encoding of the document is other than the default UTF-8 or UTF-16 (unicode transformation formats) and no encoding was determined by a higher-level protocol.

The main characteristics of XHTML is its extendability. Since XHTML inherits functions from XML and HTML, style sheets originally developed for HTML are used to define document styles, and name spaces invented for XML are used to import other XML-based markup languages defined outside of HTML.

Cascading style sheet (CSS) is first introduced to HTML for separation of content rendering from content representation. It provides a simple means to style HTML or XHTML documents, allowing the author to control visual and aural characteristics; for instance, fonts, margins, line-spacing, borders, colors, layers, and more.

CSS can be represented as either an external file or an internal description. In XML, an XML style sheet declaration is used to define style rules. In order to be compatible with this convention, style elements should have their fragment identifier set using the id attribute, and an XML style sheet declaration should reference this fragment. For example, the following XHTML code has an internal style sheet denoted by the `<style>` element and uses it by the `<code>` element. So, the text inside `<code>`, ''internal style sheet,'' is rendered as a green-colored, bold-faced monospace font in a browser.

```
<?xml-style sheet
    href="http://www.w3.org/Style Sheets/TR/W3C-REC.css"
    type="text/css"?>
<?xml-style sheet href="#internalStyle" type="text/css"?>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" >
  <head>
    <title>An internal style sheet example</title>
    <style id="internalStyle">
```

```
      code {
        color: green;
        font-family: monospace;
        font-weight: bold;
      }
    </style>
  </head>
  <body>
    <p>
      This is text that uses our
      <code>internal style sheet</code>.
    </p>
  </body>
</html>
```

Another important fearture of XHTML is a combination of namespaces. A namespace is a collection of names that are delimited in some way. An XML namespace is a mechanism for delimiting XML elements and attributes that is compatible with DTDs. The XHTML namespace may be used with other XML namespaces. Future work by the *World Wide Web Consortium* (W3C) will address ways to specify conformance for documents involving multiple namespaces.

The following example shows the way in which XHTML could be used in conjunction with *Mathematical Markup Language* (MathML), which is used for description of mathmatical formulas. The following sample code represents the formula $x^2 + 3x + 4$. The `<math>` element indicates the scope of MathML, the `<ci>` element denotes variable identifier, and the `<cn>` element denotes number. The `<plus/>`, `<times/>`, and `<power/>` elements represent add, multiple, and power operators, respectively. The `<apply>` element indicates the scope of arithmetic operations.

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
    <head>
      <title>A Math Example</title>
    </head>
    <body>
      <p>The following is an example of MathML markup:</p>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <plus/>
```

```
          <apply>
            <power/>
            <ci>x</ci>
            <cn>2</cn>
          </apply>
          <apply>
            <times/>
            <cn>3</cn>
            <ci>x</ci>
          </apply>
        <cn>4</cn>
      </apply>
    </math>
  </body>
</html>
```

2.1.3 WML

WML is another XML-based markup language that allows the text portions of Web documents to be presented on cellular phones and PDAs via wireless access [3]. WML is part of the *Wireless Application Protocol* (WAP). WAP works on top of standard data link protocols, such as *Global System for Mobile Communications* (GSM), *code division multiple access* (CDMA), and *time division multiple access* (TDMA), and provides a complete set of network communication programs comparable to and supportive of the Internet protocols.

WML considers the unique requirements of mobile units and is designed to address issues such as small display size, small memory resources, optional keyboards, and unique network characteristics like low bandwidth and high latency.

WML documents are designed to be compact and efficient on slow networks. They also support mobile applications with appropriate features such as events, push/pull, and navigation. To minimize slow requests to a WAP-enabled server, information is delivered to a WML browser in a ''deck of cards.'' Each card is a standalone piece of information that can be displayed on a mobile device's screen. By providing more than one card (the deck) in a single query, the user can navigate small portions of the WML-enabled site quickly without waiting for a response to each individual navigation or action request.

The deck corresponds to the overall <wml> ... </wml> element and encompasses all content within the document. It can optionally contain a head or template element, while it must have one or more cards. Each card can be thought of as an HTML document; cards can link to one another within a deck, or they can link to cards in other decks. By linking to a card in another deck, a separate fetch to the server (or cache) is required to retrieve the deck.

The basic structure of a WML document is as follows:

```
<?xml version="1.0"?>
<!DOCTYPE wml
   "PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
   <head>
     ...
   </head>
   <template>
     ...
   </template>
   <card id="start" title="A WML Document">
     ...
   </card>
</wml>
```

The head element in WML is similar to the <head> ... </head> in HTML, but it supports a very restricted subset of information. In general, the head element provides some access restriction and allows the author to define meta tags. Either of these categories of information is optional; without them, no access restrictions are imposed, and no meta-information is associated with the deck of cards.

The template element allows the author to describe events that will apply to all cards within the current WML element (or deck). These events can be overridden by a card, or left to apply to all cards within the deck.

The card element defines the content that the user sees. Without cards, there would be no application for the mobile device. As discussed previously, a deck consists of one or more cards. Each card can link to one another within or across decks, just as HTML pages can link to one another or to anchors prefixed by # within the page.

A typical card looks like the following:

```
<card id="doc" title="WML Document Structure">
<do type="accept">
   <noop/>
</do>
<p>
   A <b>wml</b>element can consist of<br/>
   <a href="#head">head</a><br/>
   <a href="#template">template</a><br/>
   1..N <a href="#card">cards</a><br/>
   <do type="accept" label="Next">
     <go href="#head"/>
   </do>
</p>
</card>
```

This card uses the bolding tag `<b>`, the line breaking tag `<br/>`, and the anchor tag `<a>`, as well as HTML markup. It also uses a `<do>` element to allow the user to move to the next card. This "do" instruction could easily be collapsed to an `<a>` element.

2.1.4 VoiceXML

VoiceXML is designed for creating audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and key input, recording of spoken input, telephony, and mixed-initiative conversations [4]. VoiceXML is used to bring the advantages of Web-based development and content delivery to interactive voice response applications.

Here are two short examples of VoiceXML. The first is the venerable "Hello World":

```
<?xml version="1.0"?>
<vxml version="2.0">
   <form>
     <block >Hello World! </block>
   </form>
</vxml>
```

The top-level element is `<vxml>`, which is mainly a container for dialogs. There are two types of dialogs: forms and menus. Forms present information and gather input; menus offer choices of what to do next.

This example has a single form, which contains a block that synthesizes and presents "Hello World!" to the user. Since the form does not specify a successor dialog, the conversation ends.

Our second example asks the user for a choice of drink and then submits it to a server script:

```
<?xml version="1.0"?>
<vxml version="2.0">
   <form>
     <field name="drink">
       <prompt>Would you like coffee, tea, or nothing?</prompt>
       <grammar src="drink.grxml" type="application/grammar+xml"/>
     </field>
     <block>
       <submit next="http://www.drink.example.com/drink2.asp"/>
     </block>
   </form>
</vxml>
```

A field is an input field. The user must provide a value for the field before proceeding to the next element in the form.

A grammar employed in the above example is as follows:

```
<grammar mode="voice">
   <rule id="drink" scope="public">
     <one-of>
       <item>coffee</item>
       <item>tea</item>
       <item>nothing</item>
     </one-of>
   </rule>
</grammar>
```

A sample interaction is as follows:

C (computer): Would you like coffee, tea, or nothing?
H (human): Orange juice.
C: I did not understand what you said. (A platform-specific default message.)
C: Would you like coffee, tea, or nothing?
H: Tea.
C: (continues in document drink2.asp)

Currently, it is very difficult to create VoiceXML data from other content representation such as HTML. However, if some semantic features

such as intentions and discourse structures of content are analyzed or annotated, transcoding from static documents to VoiceXML-based interactive conversational content will be possible.

2.1.5 SMIL

SMIL (pronounced "smile") enables simple authoring of interactive audiovisual presentations [5]. SMIL is typically used for multimedia presentations that integrate streaming audio and video with images, text, or any other media type.

The term "media" covers a broad range, including discrete media types such as still images, text, and vector graphics, as well as continuous media types that are intrinsically time-based, such as video, audio, and animation.

Using SMIL, the author can:

- Describe the temporal behavior of the presentation;
- Describe the layout of the presentation on a screen;
- Associate hyperlinks with media objects.

The `<head>` element contains information that is not related to the temporal behavior of the presentation. Three types of information may be contained by head: meta-information, layout information, and author-defined content control.

The `<body>` element contains information that is related to the temporal and linking behavior of the document. It acts as the root element of the timing tree. SMIL timing defines elements and attributes to coordinate and synchronize the presentation of media over time.

Three synchronization elements support common timing use-cases:

1. The `<seq>` element plays the child elements one after another in a sequence.
2. The `<excl>` element plays one child at a time, but does not impose any order.
3. The `<par>` element plays child elements as a group (allowing "parallel" playback).

These elements are referred to as time containers. They group their contained children together into coordinated timelines.

An example SMIL document is as follows:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
   <head >
     <layout>
       <root-layout width="320" height="480" />
     </layout >
   </head>
   <body>
     <par>
       <audio src="audio.rm"/>
       <video src="video.rm"/>
       <textstream src="closed-caps.rt" systemCaptions="on"/>
     </par>
   </body>
</smil>
```

Currenly, SMIL documents are created by multimedia presentation authoring tools and are not customizable on demand. Multimedia transcoding, which is a tranformation of multimedia content according to user preferences, is described in Section 4.8.

2.2 Content Adaptation

Content adaptation is a type of transcoding that considers a user's environment—devices, network bandwidth, profiles, and so forth. A content adaptation system can dynamically filter, convert, or reformat data for content sharing across disparate systems, users, and emerging pervasive computing devices.

The transcoding benefits include:

1. Eliminating the expense of reauthoring or porting data and content-generating applications for multiple systems and devices;
2. Improving mobile employee communications and effectiveness;
3. Creating easier access for customers who are using a variety of devices to purchase products and services.

The transcoding technology enables the modification of Web content, such as converting images to links to retrieve images, converting simple tables to bulleted lists, removing features not supported by a device such as JavaScript or Java applets, removing references to image types not

supported by a device, and removing comments. It can also transform XML-based documents by selecting and applying the right style sheet for the current request based on information in the relevant profiles. These profiles for preferred transcoding services are defined for an initial set of devices.

2.2.1 XSLTs

Unlike HTML, which mixes data and display information, XML is intended to represent the content and structure of data, without describing how to display the information. Although XML allows the structure of a set of data to be defined, it does not usually provide information for changing the structure of the data to conform to a different set of requirements.

For instance, if the application service A uses one structure to represent a set of data, and the other application B uses a different structure to represent the same type of data, XML itself does not provide the information to transform the data from one structure to the other.

W3C is discussing and maintaining a standard recommendation called XSLT [6]. XSLT is a language for transforming XML documents into other documents. It can be used to transform XML into markup languages suitable for display (like HTML), or into XML documents having a different structure.

An XSLT style sheet, which is an XML-based document written in XSLT, contains a set of template rules. A template rule has two parts: a pattern that is matched against nodes in the source tree and a template that can be instantiated to form part of the result tree. This allows a style sheet to be applicable to a wide class of documents that have similar source tree structures.

A template is instantiated for a particular source element to create part of the result tree. A template can contain elements that specify literal result element structure. A template can also contain elements from the XSLT namespace that are instructions for creating result tree fragments. When a template is instantiated, each instruction is executed and replaced by the result tree fragment that it creates. Instructions can select and process descendant source elements. Processing a descendant element creates a result tree fragment by finding the applicable template rule and instantiating its template. Note that elements are only processed when they have been selected by the execution of an instruction. The result tree is constructed by finding the template rule for the root node and instantiating its template.

The following is an example of an XSLT style sheet that transforms XML data into HTML documents:

```
<xsl:style sheet version="1.0"
    xmlns:xsl"http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/TR/xhtml1/strict">
<xsl:template match="/">
<html>
     <head>
       <title>Customers</title>
     </head>
     <body>
       <table border="1">
         <tbody>
           <tr>
             <th>Name</th>
             <th>Age</th>
           </tr>
           <xsl:for-each select="item">
             <tr>
               <td>
                 <xsl:value-of select="name"/>
               </td>
               <td>
                 <xsl:value-of select="age"/>
               </td>
             </tr>
           </xsl:for-each>
         </tbody>
       </table>
     </body>
   </html>
</xsl:template>
</xsl:style sheet>
```

An example of XML data, which is a target of the XSLT transformation, is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-style sheet type="text/xsl" href="transform.xsl"?>
```

```
<personnel>
   <item>
     <name>Katashi</name>
     <age>40</age>
   </item >
   <item>
     <name>Kazue </name>
     <age>30 </age>
   </item>
   <item>
     <name>Hiroshi</name>
     <age>20</age>
   </item>
</personnel>
```

Note that this XML data contains a declaration of a style sheet to be applied that must be named "transform.xsl" and located in the same folder/directory as the XML file. The transformation result can be seen as follows:

Name	**Age**
Katashi	40
Kazue	30
Hiroshi	20

XSLT style sheets are also applicable to any XML-based content that does not have a style sheet declaration. This is an important feature for transcoding, because a single XML document can be converted into several rendering formats in accordance with the corresponding style sheets. An automatic selection and application of style sheets are described in more detail later.

One of the major design issues related to the economics of maintaining Web applications is whether to use XML in place of rendering-based markup languages, such as HTML or WML. For many Web applications, it makes sense to represent content as a single format that describes the semantics of the information. This format is more widely useful than the same content in an HTML or WML format that cannot be easily manipulated for other data access requirements.

However, there are costs involved in writing and maintaining XSLT style sheets. For these costs to be lower than maintaining different versions of the content and applications, style sheets have to be written intelligently, with parameters that allow a single style sheet to work for multiple presentations and with reusable chunks that can be used across applications.

2.3 Transcoding by Proxy Servers

A proxy-based content transcoding is a flexible mechanism for integrating multiple transcoding techniques. For example, an IBM product called *WebSphere Transcoding Publisher* (WTP) [7] performs several kinds of transcoding such as:

- *HTML simplification:* modifying HTML documents to tailor them for reduced function devices, for example, by removing unsupported features;
- *HTML-to-WML transformation:* transforming HTML documents into WML documents suitable to be sent to WML-based phones;
- *XSLT style sheet selection and application:* selecting the right style sheet to convert a particular XML document for presentation on a particular device;
- *Fragmentation:* breaking a document into fragments that can be handled by a particular device and creating the navigation links between the fragments;
- *Image transcoding:* modifying images by converting them to different formats or reducing scale or quality, or both, to tailor the image for presentation on a particular device.

WTP works as a proxy server and is located somewhere between content servers and client browsers. It maintains client device profiles/properties and style sheets and activates several transcoding functions.

2.3.1 HTML Simplification

Although powerful full-function browsers such as Microsoft Internet Explorer 5 (and later) are widely used, many less powerful devices have browsers that support only a subset of the HTML functions supported by the more powerful browsers. In addition, because some features, although

supported, might so strain the resources of the device that they are troublesome, a user might prefer to have a less expensive construct substituted.

For example, there are numerous handheld or palm-size computers running the Windows CE or PocketPC operating system, which uses a reduced function version of Microsoft's Internet Explorer. Devices with even more constrained resources, especially memory and processing power, may have browsers with even less support for HTML functions, such as frames, JavaScript, and so forth.

Despite the large number of limited-function devices, much of the HTML content being written today is produced to take advantage of the advanced capabilities of powerful browsers. Rather than each content provider producing multiple versions based on the capabilities of each possible client device, it is desirable to allow the content to be created once. Transcoding can be used to transform the content into a form suitable for the client device. In WTP, this process of transforming the HTML content based on the capabilities of the client browser is called HTML simplification. This transcoding could be done by characterizing the capabilities of devices and using these characteristics as parameters to direct the transcoding function.

By using a set of properties that describes the capability of a given browser to render various HTML elements, a profile describing the browser can be generated. This profile provides sufficient information to direct the transcoding process. For example, a property that prefers textlinks to images in a wireless network environment might be set to true, indicating that the HTML simplification process should generate a link to a separate page containing an image rather than automatically downloading it and rendering it inline in the page, thus giving the user the choice of whether to spend the time and byte transmission cost of downloading the image to the device.

As another example, some limited-function browsers are unable to render HTML table elements. For this case, WTP uses a property that prefers tables to be converted into unordered lists that can have a true or false value. For a limited browser that does not support tables, this value is set to true to indicate that the table should be converted into an unordered (nested) list. This preserves the relationship between nested tables by representing them as nested unordered lists.

Currently, approximately 20 rules for properties in WTP can be used to guide the HTML simplification process, including properties that indicate which types of image formats are not supported, whether frames are supported, and whether images should be removed entirely.

HTML simplification is only applicable if the content is marked in HTML. The HTML simplification described here is available in WebSphere Transcoding Publisher. It is customizable using the preferences, but flexibility is currently limited to those preferences. Figure 2.1 shows an example of HTML simplification. Figures 2.2 and 2.3 show the before and after HTML documents from Figure 2.1 rendered in a browser.

Before transcoding

```
<HTML>
  <HEAD>
   <title>Personnel Data</title>
  </HEAD>
  <BODY>
   <h1>Personnel Data</h1>
   <table border="1">
   <tr>
   <th>ID</th>
   <th>Name</th>
   <th>Age</th>
  <th>Major</th>
   </tr>
   <tr>
   <td>001</td>
   <td>Katashi Nagao</td>
   <td>40</td>
   <td>Engineering</td>
   </tr>
   <tr>
   <td>002</td>
   <td>Kazue Nagao</td>
   <td>30</td>
   <td>Education</td>
   </tr>
   <tr>
   <td>003</td>
   <td>Hiroshi Nagao</td>
   <td>20</td>
   <td>Economics</td>
   </tr>
   </table>
  </BODY>
</HTML>
```

After transcoding

```
<HTML>
  <HEAD>
   <title>Personnel Data</title>
  </HEAD>
  <BODY>
   <h2>Personnel Data</h2>
   <ul>
   <li>001
   <ol>
   <li>Name: Katashi Nagao</li>
   <li>Age: 40</li>
   <li>Major: Engineering</li>
   </li>
   </ol>
   <li>002
   <ol>
   <li>Name: Kazue Nagao</li>
   <li>Age: 30</li>
   <li>Major: Education</li>
   </ol>
   </li>
   <li>003
   <ol>
   <li>Name: Hiroshi Nagao</li>
   <li>Age: 20</li>
   <li>Major: Economics</li>
   </ol>
   </li>
   </ul>
  </BODY>
</HTML>
```

Figure 2.1 An example of HTML simplification.

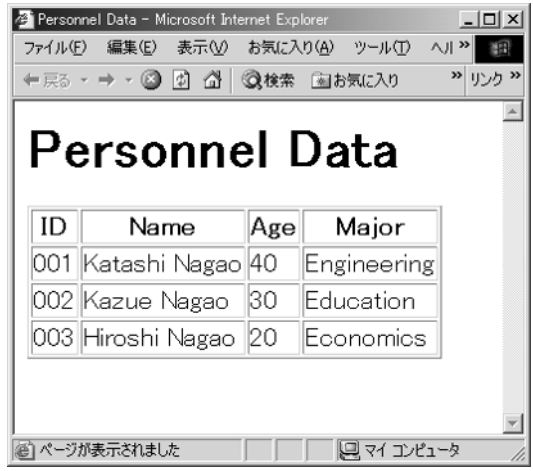


Figure 2.2 Rendering image of original document.

2.3.2 XSLT Style Sheet Selection and Application

An increasingly used approach with XML is to generate content in XML and use XSLT style sheets to transform the information into a form suitable for display, or to transform the structure to a different one suitable for another

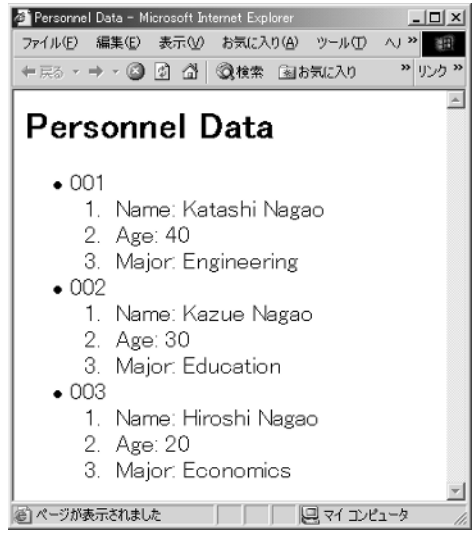


Figure 2.3 Rendering image of transcoded document.

party. Thus, XSLT style sheets can be written to support display of XML content as well as to support the use of XML in *electronic data interchange* (EDI).

WTP provides a framework to aid in using XSLT style sheets to transform XML documents. WTP includes a version of the Xalan XSLT processor [8], an open source XSLT processor supported by the Apache XML Project. The major benefits that WTP provides in dealing with XSLT style sheets and their application to XML documents is a structure for organizing style sheets and support for defining the criteria used for selecting the proper style sheet to be used in the transformation of XML documents. For example, an organization might generate content in XML and create different style sheets to render the content for display on different device types. They might create one style sheet for generating an HTML version, another for a WML version, and so forth.

WTP registers a given style sheet, providing the location of the style sheet, the content type generated (for example, text/html), and selection criteria that must exist in the context of an incoming XML document request in order for the style sheet to be used in transforming that XML document. For example, key-value pairs representing properties associated with a given device, such as the descriptive name of the device, can be used as criteria, guaranteeing that the style sheet will only be used on requests originating from such devices. Additionally, conditions matching all or a portion of a given document URL can be specified, guaranteeing that the style sheet will only be used on one or a small number of documents originating from a given server location.

As the number and types of devices allowing Internet access increase, an approach based on different style sheets for each device type could cause the number of style sheets an organization needs to create and maintain to increase precipitously. As WTP and style sheet tools evolve, there will be increasing support for style sheet parameterization and style sheet internalization, and increasingly complex criteria for selecting the right set of style sheets for a given purpose. Style sheets are standard, general, and extensible, but using style sheet transcoding requires that an XSLT style sheet be written to perform the desired transcoding.

2.3.3 Transformation of HTML into WML

WTP transforms HTML into WML in a two-step process, with an optional third step to further refine the result. The first step takes the original

HTML document and converts it into a *document object model* (DOM) representation using an HTML parser. The DOM is essentially a tree structure, conforming to a W3C standard. Using standard Java *application programming interfaces* (APIs) for traversing and manipulating the DOM, the second step modifies the HTML DOM, creating a DOM containing WML elements, conforming to the WML standard. This transformation is done on an element-by-element basis. For example, an HTML document starts with an HTML element (node), whereas a WML document starts with a WML element. Thus, the transformation logic responsible for processing the HTML node replaces this node with a WML node in the DOM.

The logic for dealing with a particular HTML element is classified into one of three categories:

1. The transformation is done according to one-to-one correspondence, such as the HTML element transformed to the WML element.
2. The source element, and all the children under it, are simply deleted. An example is the APPLET element, which invokes a Java applet. Since it is not possible to transform the Java applet into some other form for execution on the Java-incapable devices, the APPLET element and its contents are simply deleted.
3. The source element is replaced by its children. That is, everything under the top element replaces the top element itself. For example, the FONT element, which indicates text contained within it should be rendered in a font with specified color or size, is replaced by the text it contains, since FONT elements are not supported in WML. In text form, this would be seen as replacing `<font color="red">some text</font>` with simply "some text."

The first case above requires a simple rule for each transformed element, whereas the second and third cases can each be handled by a single processing module, with the HTML elements to be handled simply by providing them to the module in the form of some list. Thus, the module responsible for deleting elements and their children is given a list of element names that should be transformed in this way.

The HTML to WML transcoding approach described above can be used for transforming HTML to other markup languages, such as compact HTML.

2.3.4 Image Transcoding

Images that view well and render reasonably quickly on a PC browser connected to a *local area network* (LAN) may not appear in a usable format on the low-resolution and small-sized screens of mobile devices. In addition, they may take unacceptably long to download and render on such devices. Finally, some devices do not support all image formats; therefore, to display an image at all it may be necessary to convert from the original image format to one supported by the device before transmitting the image to the device.

Image transcoding is performed in WTP by the image distiller engine. For example, an image can be scaled and converted to gray scale to reduce the image size for display on a wireless device. Given an input image to transcode, the image engine must determine three things: the output format that should be used for the image, the scale factor to be applied to the image, and the output quality of the image. Output quality is a measure of how good the image will appear on the screen. High-quality images will look sharp and clear but will generally be considerably larger in data size than lower-quality images. Also, note that low-resolution screens may be incapable of displaying a high-quality image. In this case it is preferable to use a lower-quality image as output, since that will speed both transmission and rendering time.

Deciding the output format is very simple. If the input image is in a format accepted by the device, the format is not changed. Otherwise, the output format is set to be the most preferable one of the formats supported by the device as determined by the *user agent* (UA) field in the *Hypertext Transfer Protocol* (HTTP) request (HTTP header that reflects the type of Web clients).

Scale factor may also be a simple configuration parameter associated with the requesting device. Generally, it will be set so that images appear relatively as big on the screen of the requesting device as they would on a PC screen. PDA devices, however, do not have a fixed screen size. Rather, the screen size is indicated by the pixels header in the UA field included in the HTTP request. When this header is present, the image distiller engine can use this information to dynamically calculate a scale factor based on the difference in size between the screen described by a pixels header and some configurable standard screen size.

Finally, output quality is determined by combining information about the quality of the screen of the requesting device and the configured trade-off that should be made between quality and size for this request. The screen quality is a configured parameter for the device and may be set to high,

medium, or low. The trade-off configuration parameter will generally be associated with a network type: Over a slow wireless link it would be better to favor small size over high-quality images, whereas over an LAN connection high quality would be preferred, since there is little to be gained in reducing the image size. The image engine first chooses a tentative output quality based on the screen capability and then adjusts this size downward if the trade-off parameter is set to either compromise or favor small size.

2.4 Content Personalization

Transcoding works for not only device-dependent adaptation but also user-dependent customization. For example, users may have some suitable conditions to access information such as background and foreground colors, font sizes, media or modalities (i.e., text, graphics, sound), structures or layouts (i.e., less complicated view without pop-up windows), and interfaces (i.e., mouse clicks, keypads, speech).

Content personalization consults personal profiles that describe preferable settings for some user's information access and transcodes requested content to satisfy the conditions. This is a key issue of "information accessibility," which means that all people including seniors and any kind of handicapped people can easily use information systems.

2.4.1 Transcoding for Accessibility

Users can get information through the Web by using various tools. For example, voice-capable browsers for visually challenged people have been widely available. They contribute to improving the environment of information resources for nonvisual access. However, a new problem is arising. Search engines are combined with on-line shopping sites, for example, and news sites serve not only articles, but also various types of information. Authors of Web content tend to cram various functions and many links into one page to improve the usability for visitors on the Web. In order to solve this problem, some guidelines for Web content, browsers, and authoring tools have been established by the *Web Accessibility Initiative* (WAI) of W3C [9]. They have defined detailed guidelines for accessible Web design, but authors rarely follow these guidelines. The main reason is that they do not have enough time to take account of the accessibility issues because of the fierce competition with other sites. They need to make their own sites more attractive to get and keep as many clicks as

possible, and therefore they tend to concentrate on the visual aspects of their content.

Since it is very difficult to rely only on content authors to improve Web accessibility, Asakawa's group at IBM Research is developing a transcoding system to convert already-existing Web content to be more accessible, by providing an intermediary proxy server between servers and clients [10]. While transcoding technology in general has mainly focused on adaptation for non-PC client devices, in order to create truely accessible content, we have to consider several physical and mental conditions of individual users.

Transcoding for accessibility sometimes uses annotation that is explained in the following chapter. There are three types of annotation: volunteer specified, user specified, and automatically created. These annotations are very simple for allowing many volunteers to participate easily and quickly. To simplify HTML documents, the transcoding system mainly gets the differences between two HTML documents. The system also reorders the HTML elements based on annotation of visually fragmented groupings. This section introduces an example of the accessibility transcoding system and the user interface to control personal preferences to the transcoding proxy server.

2.4.1.1 Related Systems

Various researchers have studied content personalization for some specific purposes. Brusilovsky et al. [11] have developed an on-line textbook authoring tool that adapts to students' level of understanding. Their technique involves embedding tags into the original document. Milosavljevic et al. [12] have developed systems for dynamically producing documents from abstract data and information about the user. In both cases, the amount and type of personalization/adaptation is limited compared to our system. It is also much more difficult to adapt current contents to fit these systems, whereas our system will work immediately with content anywhere on the Web. Fink et al. [13] discussed some technique on adapting Web information for handicapped individuals.

A filtering program called Betsie developed by the BBC can transcode BBC Web content into more accessible formats, mainly for visually challenged people [14]. The system has two functions, one for converting a Web document to a document of only text with user-specified color and font size settings, and one for reordering document structure to move indexes to the bottom of a page. Their functions are restricted to some specific sites.

A system called DiffWeb provides a service for getting the differences between two HTML documents [15]. The main use is to compare a target HTML document with a previous document that has been stored by a user. It then presents the newly updated information. When there is any difference between the two versions of the HTML documents, DiffWeb can show that updated information. Each user has an independent list of URLs and it appears when a user logs into the DiffWeb page. The preserved pages can be updated only manually, so when a user prefers to compare a future page with the current version of the page, he or she needs to save it. This means that this service cannot be used spontaneously while surfing the Internet.

ALTifier tries to insert alternative text for images without ALT attributes based on various rules and heuristics [16]. For example, it inserts the title of a linked page, recognizes a small image as a bullet and so on. It is very useful to adopt the title of a page as an alternative text of an image.

2.4.1.2 System Architecture

The system consists of a proxy server that uses the above-mentioned WTP with two databases, one for an annotation database and one for a user profile database. The annotation database is incorporated in a separate annotation server. There are two main modules in the accessibility transcoding system, a transcoding module and a personalization module, as shown in Figure 2.4.

The transcoding module consists of several components: a preference setting module, a simplification module, a reordering module, and a module with experience-based rules. The personalization module accesses the user

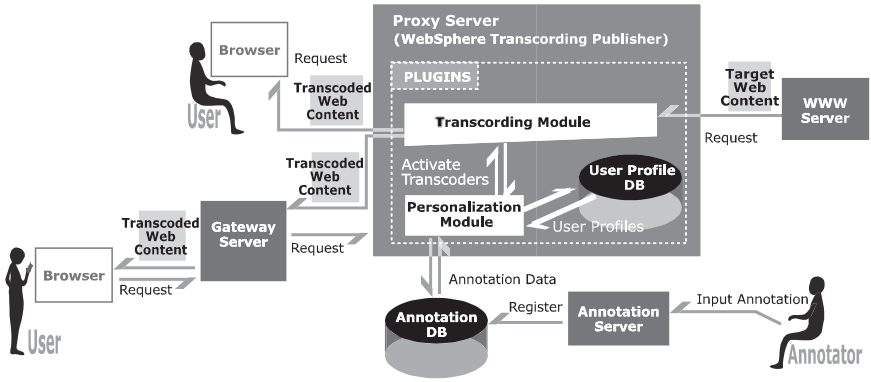


Figure 2.4 Accessibility transcoding system configuration. (*From*: [10]. © 2000 ACM Press. Reprinted with permission.)

profile database and the annotation database, and activates transcoders in the transcoding module.

The preference setting module authenticates users and controls preference setting forms. The simplification module simplifies Web documents based on the differential methods. The reordering module transcodes Web documents based on annotations created by volunteer annotators. The module with experience-based rules transcodes Web content using some heuristics acquired from experiences of nonvisual Web access.

The architecture is extendable so that there are possibilities for including a summarization module, a translation module, a more advanced personalization module, and so forth. The annotation database manages various kinds of annotation files. The user profile database preserves users' settings and user-specific annotations.

2.4.1.3 Transcoding Process

When a document is accessed, the transcoding system creates an automatic annotation data element for the document and stores it in the database. Then it checks the current transcoding mode. There are three transcoding modes, one for simplification, one for reordering, and one for returning the original content. When in the reordering mode, the system changes the page layout for each visually fragmented grouping. If there is a volunteer-specified annotation file, the system refers to it; if not, it looks for a user-specified annotation data. If neither of them exists, it refers to the experience-based rules.

When in the simplification mode, the system simplifies the document based on our differential method. If there is a volunteer-specified annotation file, the system refers to it for getting the differential. Next, it refers to a user-specified annotation file and finally adopts the experience-based rules. When in the original page mode, the original page is modified only based on the automatically created annotations.

2.4.1.4 Preference Setting Module

This module authenticates users. If a user specifies the transcoding proxy for the first time, it asks the user to input a user ID, an e-mail address, and the user's name. This information is used for each user's preference setting.

Any document that is transmitted through the transcoding proxy has ''Settings'' as a link at the bottom of the page. When a user selects this link, the preference setting page will appear. As ''user annotation,'' there are two links, one for selecting a proper form and one for selecting proper content. When ''proper form'' is selected, a user can select a proper form that is

commonly used by him or her. After selecting it, the proper form always appears at the top of that page. If a user prefers to have only an input box and a submission button in the form, he or she can choose that option.

In this way, a user can use Web search engines in an easy way, since it will show only an input box and a submission button. When users select "proper content," the system will insert links at certain candidate elements on the screen. When a user selects one of these links, the system registers the position as a starting point of the proper content in a page. The system has analyzed that these locations might be the proper content based on our experience. For example, a string that has more than 40 characters without any link, or a string that starts under a horizontal separation line might be the start of the proper content. After registering the starting point of the proper content in a page, a user can navigate based on the proper content. This function is useful, since visually challenged users would not need to depend on sighted volunteers.

2.4.1.5 Simplification Module

This module finds the differences between pairs of HTML documents in order to simplify Web documents. It refers to a volunteer-specified annotation file, if it exists, to create a better organized page without destroying the original contents.

This method allows users to access only newly updated information, since it gets the differences between two HTML documents that have the same URLs. It is especially useful for a result page from a Web search engine. For example, a user can search for some keyword today and again a week later, and he or she will be able to easily access the new results using this function because the two HTML documents would have the same URLs. The system can even find a page to be compared with for differences using the network, even when there is no page to be compared with in the cache. This method has two stages, one for specifying or finding an HTML file to use as the basis of comparison, and one for actually finding the differences between the HTML documents.

2.4.1.6 Specifying an HTML Document for the Difference

It is necessary for the system to find a document to be compared with when there is no content with the same URL as the target document in the cache. For example, news sites change URLs everyday for articles. Each URL of an article in a news site often includes the date and the month, such as http://www.anewspaper.com/0423/news/national238.html. In this URL, "0423" means April 23, and "national238" in the file name means that

the article is the eighth news item on April 23rd in the national category. This means that there is no previous file with the same URL in the cache. The system therefore needs a method for specifying the data to be compared with the target data in order to get the differences between the two HTML documents. As a first approach for this purpose, the system lists the neighboring URLs and specifies a URL using the following method:

1. *List URLs in the target document:* After a target HTML file is loaded, the system parses it and lists all URLs described in HREF attributes of anchor tags. From the list, the system selects files that exist in the same directory with the target file, and then adds them to a candidate list.

   Example target: http://www.anewspaper.com/0423/news/national238.html. Selected candidate: http://www.anewspaper.com/0423/news/national239.html.
2. *Select files having the same parent directory as the target document:* When the first method cannot find any candidates, the system then tries to list files that share the parent directory. The following two index.html files exist in different directories, ''cuba.boy'' and ''spokane.slayings,'' but their parent directory is the same as ''2000/US/04/22/'' http://www.anewspaper.com/2000/US/04/22/cuba.boy/index.html; and http://www.anewspaper.com/2000/US/04/22/spokane.slayings/index.html.

   If again no candidate is found, the module searches for files toward the root (topmost parent) directory.
3. *List each index file for each directory:* A site often provides an index data (often named index.html or default.asp) not only for the site's top page, but also for each directory. It is often useful for the system to compare an index data with a target document to get the difference because they tend to use the same common template for all data in the same directory. So the system always checks for an index data and it is listed on the candidate list. For example, for http://www.anewspaper.com/2000/US/04/22/cuba.boy/index.html, the following directories that might contain index data can be extracted from this URL:

   - http://www.anewspaper.com/2000/US/04/22/;
   - http://www.anewspaper.com/2000/US/04/;
   - http://www.anewspaper.com/2000/US/;

- http://www.anewspaper.com/2000/;
- http://www.anewspaper.com/.

These URLs for index files are automatically created from the target URLs. If these URLs do not exist, the module removes the incorrect URLs from the candidate list.

4. *Search previous documents from cache database:* The system uses not only neighboring documents, but also stored documents. The module searches the documents in the cache database and selects any documents of the target URL. These documents are added to the candidate list. Next, the module calculates the differences between the target HTML document and each one in the candidate list. A *dynamic programming* (DP) matching method is used to calculate the differentials. DP matching can calculate the longest common string that is included in two strings. For example, the longest common string between ''abcdef'' and ''abdgf'' is ''abdf.'' This method can be applied to calculate the difference between two HTML documents. The calculation process is as follows:

   - Parse two HTML documents and generate their DOM representations (DOM trees).
   - Traverse and serialize the DOM trees. The module converts a DOM tree for each HTML file into a string structure of nodes. The nodes that should be included in the string structure are selected using HTML TAG information such as BR, TABLE tags, which are used to preserve page layout.
   - Calculate the *longest common "node" string* (LCNS) using DP matching. This string consists of the contents that are the same in the two HTML files, so these nodes should be removed.
   - Remove nodes in the LCNS from original DOM tree. Finally, the nodes in the LCNS are actually removed from the DOM tree for the desired HTML page. If a volunteer-specified annotation file exists, the module tries to detect visually fragmented groupings and use the information. If any element in a group is updated, it does not remove any information in the group.

2.4.1.7 Reordering Module

This module depends on volunteer-specified annotations. In the annotation data, a rendered document is divided into visually fragmented groups,

and each group is annotated with its role and importance. This module reorders each group by referring to its role and importance. In this way, a user can read a document effectively in the new layout. The module also inserts a delimiter between each group, so a user can recognize when the current group changes. Finally, it inserts a description or comment of an image or any other element that is annotated.

2.4.1.8 User-Side Customization Module

This module depends on user-specified annotations. There are two kinds of annotations, such as marking the region of a proper content and selecting a proper input form in a document. Optionally, a user can select to see only an input box and a submission button for the proper form.

There are two modes of document rendering:

- *Full text:* When a starting point for the proper content is specified, it moves the information above the proper content to the bottom of a page. When a proper form is selected, it moves the form to the top of a page. When both of them are specified, the system places the proper form above the proper content.
- *Simplified text:* When a starting point for the proper content is specified, it removes the information above the proper content. When a proper form is selected, it removes all other information in a page.

2.4.1.9 Experience-Based Rules

The system applies the experience-based rules. The main purpose is to move the proper content to the top of a page. There are three functions, depending on the transcoding mode. In the case of the full text transcoding mode, first, it moves image maps to the bottom of the page. Second, it moves forms to the bottom of the page. Finally, it moves link lists to the bottom of the page. In the case of the simplified text transcoding mode, it removes them instead of moving them to the bottom of a page. There are often image maps, forms, and link lists above the proper content, so this function will help users to access the proper content directly. There is another function to remove duplicated links, which is used in both modes.

We often see duplicated links in a page. One link is an image link and another is a text link. Both of them have the same URLs as the anchor text. From this information, the system can decide to remove one of them.

2.5 Framework for Transcoding

Before describing the framework for transcoding, some terms are first defined. Intuitively, data objects represent content, type indicates the form the data is in, properties represent attributes of particular data types, and format combines a type and a set of properties. More precise definitions are as follows:

- A *data object* consists of a sequence of bytes.
- A *type* is one of the *Multipurpose Internet Mail Extensions* (MIME) types. It represents the content description of data objects of that type and specifies the byte-level encoding used to represent the data. For example, data objects of the type image/gif are images in which the data are encoded according to the *graphic interchange format* (GIF).
- A *property* is an attribute particular to a type or set of types that may take on values represented as strings. The special value * represents any value. For example, the type text/xml might have the property *document type definition* (DTD), which could take on values such as "http://www.w3.org/TR/1999/PR-xhtml1-19990824" or "*".
- A *format* consists of one type and zero or more property-value pairs. For example, ("text/xml," (("DTD," "foo"))) is a valid format, is ("text/html," (("case," "upper"), ("length," "500"))).
- A *transcoding operation* takes an input data object d_{in} in a particular format f_{in} (I use the notation $d_{in}(f_{in})$) and converts it into an output data object d_{out} in the specified format f_{out} (using again the notation $d_{out}(f_{out})$). Thus, a transcoding operation is denoted by $(d_{in}(f_{in}), f_{out}) \rightarrow d_{out}(f_{out})$. The type of the input format f_{in} and the type of the output format f_{out} may be identical. If so, and if the set of properties specified by f_{out} is empty, the transcoding operation is the identity transformation and d_{out} is identical to d_{in}.

Properties describe transformations of a more semantic nature than those the MIME type system covers. These transformations may or may not be reversible. For example, text summarization is not reversible, nor is increasing the compression factor in a JPEG image, because both transformations discard information. Other transformations may be reversible, such as inverting the color map of an image. In any event, all such transformations can be described by properties in the transcoding system, because they do not affect the type of the object.

In this book, *intermediaries* are defined as a general class of computational entities that act on data flowing along an information stream. In general, intermediaries are located in the path of a data flow, possibly affecting the data as they flow by. For HTTP streams, for instance, intermediaries might be used to customize content with respect to the history of interaction of individual users, to annotate content, such as marking up URLs to indicate network delay, to add awareness and interaction in order to enable collaboration with other users, to transcode data from one image type to another, or to cache and aggregate content.

The transcoding framework is implemented as an intermediary for a well-defined protocol for document or object retrieval, such as HTTP or WAP. This intermediary can inspect or modify both requests for objects and the responses—that is, the objects themselves. The intermediary performs transcoding operations on these objects. Clients may request that the intermediary transcode responses on its behalf, or the intermediary may make that decision itself based on other factors. To perform this transcoding, the intermediary relies on a set of transcoders, each of which advertises its capabilities. The advertisements specify what sort of output object a transcoder is able to produce given certain constraints on its input. A transcoder that translates Japanese documents into English might specify the following.

((text/html, (("language" "ja"))), (text/html, (("language", "en"))))

A * on the right side of an (attribute, value) pair indicates that the transcoder can produce any value requested for the corresponding attribute. A transcoder can advertise more than one capability.

Once the desired transformation is determined, the capabilities of each transcoder are examined in order to identify a set of transcoders that can perform the requested operation. Once an appropriate set of transcoders has been selected, each one is invoked in turn with two inputs that specify the transcoding request: (1) the output of the previous transcoder (or the original input, in the case of the first transcoder); and (2) a transcoder operation, which specifies one or more of the operations advertised in a transcoder's capabilities statement. Each transcoder operation includes the input format of the object supplied and the output format of the object to be produced, both of which must respect the transcoder's capabilities.

More precisely, a transcoding request R is valid for a transcoder T given that T has a set of capabilities $\{C_1, ..., C_n\}$ if:

1. There exists at least one capability C_i such that the types specified in the input and output formats of C_i are identical to the types specified in the input and output formats of R. Let the set $\{D_1, ..., D_m\}$ denote all members of $\{C_1, ..., C_n\}$ that meet this criterion.
2. There exists a subset E of D such that the union of all property-value pairs in the output formats of the members of E is identical to the set of property-value pairs of the output format of R, and the union of all property-value pairs in the input formats of the members of E is identical to the set of property-value pairs of the input format of R, subject to the following conditions:
   - Any property-value pair in any input format or in the output format of R with a value of * is meaningless; it is as though the pair were not present.
   - Any property-value pair in an output format of a member of E with a value of * will be considered identical to a property-value pair in R with an identical property and with any value.

The operations of different transcoders can be composed in a straightforward way. Generally, the idea is to break down an overall request into a series of subrequests to different transcoders, each of which accomplishes part of the overall goal or some other necessary subgoal. Specifically, a list of subrequests $(S_1,...,S_n)$ can be considered equivalent to an overall request R if the following conditions are met:

- The type of the input format of the first request S_l is identical to the type of the input format of R.
- The type of the output format of the last request S_n is identical to the type of the output format of R.
- Each property-value pair in the input format of R is present in the input format of some subrequest S_i.
- Each property-value pair (P, V) in the output format of R is present in the output format of some subrequest S_j such that there does not exist any subrequest S_k, $k > j$, whose output format contains a property-value pair (P, V'), $V \neq V'$.

The net effect of these conditions is that every property specified in the output format of a request R may take on any value at various stages throughout the chain, as long as the final value that it takes is the one requested in R.

In this framework, the use of properties in addition to MIME types allows transcoders to declare their ability to change attributes of an object other than its type. The transcoding system allows type-preserving and type-changing transformations to be automatically combined in any number and order within the scope of a single intermediary. If one proxy handles type-altering transformations and another one handles type-preserving transformations, the order in which the proxies are stacked dictates the only order in which those transformations may be performed. Of course, it might be possible to give the proxies detailed knowledge of each other. With this knowledge, they could forward requests back and forth until all required transformations are performed. However, one of the advantages of a stacked proxy architecture is that one proxy generally does not know what function the other proxies perform, and may not even know of their existence. This lack of knowledge allows a clean architectural separation of function, but if the functions are heavily intertwined, it makes more sense and is more efficient to combine them in a single intermediary.

In addition, the use of a formally specified language to describe the abilities of transcoders allows transcoders to be packaged and interchanged in a simple and automated way, enabling the creation of transcoder repositories. Thus, an intermediary unable to satisfy a request might automatically search for, retrieve, install, and use an appropriate transcoder. A third-party transcoder repository service could publish these transcoders, enabling any system that knows about the repository to discover, download, and combine the transcoders seamlessly.

2.5.1 Transcoding Framework in Action

Consider the case of a worker away from the office. Suppose he or she is traveling by car, perhaps making a sales call. Suppose further that this worker's Internet-connected mobile phone can request data from the office via a transcoding proxy, and that he or she wants to hear an English audio summary of a long document, such as a sales contract. The mobile phone browser requests the document from the transcoding proxy. The phone-browser knows that the user wants an audio summary, either because it has been preconfigured or through some other means (e.g., because an earpiece is plugged into the phone). Suppose the original document is a *portable document format* (PDF) document written in Japanese. In this case, the phone-browser might specify its preference for an audio summary by including with its request a header line, such as,

Transcoding-Request:
((application/pdf, ((“language”,“ja”))), (audio/mp3,
((“summarize”, “10.0”),
(“language”, “en”))))

To satisfy the request, the intermediary first retrieves the original document from its origin. Because a transcoding specification was included in the original request for data, the intermediary must transcode the data before passing the data along to the client. To satisfy the transcoding request, the intermediary first looks for a single transcoder that can do the job of transforming Japanese PDF documents into summarized, English audio. Because there are no special transcoders for this, the intermediary next tries to find a chain of transcoders that, when applied in sequence, satisfies the request.

The chain of transcoders is determined by simple backward chaining, with the desired output type examined first. If there is no single transcoder that can produce audio/mp3, then the request cannot be satisfied. If a transcoder is available, the input requirements of the transcoder are examined in order to identify another transcoder that can output a matching type. This process repeats until the input type of the last transcoder selected matches the input type of the original transcoding request. Furthermore, the output format of the original transcoding request must be satisfied by the chain of transcoders.

Let us consider this example more carefully:

1. The desired output type is audio/mp3 and the only available transcoder that can output audio is the following:
   ((text/plain, ((“language”,“en”))), (audio/mp3, ()))
   The transcoder’s capability advertisement states that plain text written in English can be converted into audio. This transcoder will be selected as the last transcoder in the chain.
2. Because the transcoder selected in step 1 only accepts English input in plain text, the framework must find a transcoder that outputs plain text in English. Suppose a language-translation transcoder is available:
   ((text/plain, ((“language”,“ja”))), (text/plain, ((“language”,“en”))))
   At this point, two jobs remain. First, the system should find a transcoder that can summarize text and, second, find a transcoder that can convert PDF into plain text.

3. Suppose there are two such transcoders available, a PDF-to-text converter,

   ((application/pdf, ()), (text/plain, ()))
   and a text summarizer,
   ((text/plain, ()), (text/plain, ((“summarize”,“*”))))

. One final problem remains—that is, ordering these last two transcoders. If the PDF converter is selected first, the next step would be to find a summarizer that outputs a PDF document. Because there is no such PDF summarizer, the chain of transcoders cannot be completed. Because our framework implements a search process that can backtrack, it can revoke the selection of the PDF converter and select the summarizer, which then leads to the selection of the PDF-to-text converter.

   The overall sequence of transcoding operations for our example request is as follows:

   ((application/pdf, ()), (text/plain, ()))
   ((text/plain, (())), (text/plain, ((“summarize”,“10.0”))))
   ((text/plain, ((“language”,“ja”))), (text/plain, ((“language”,“en”))))
   ((text/plain, ((“language”,“en”))), (audio/mp3, ()))

Note that the individual transcoding units in this example are reusable and can be combined in many ways with other transcoders as shown in Figure 2.5.

The text summarization transcoder might be used together with a text-to-WML transcoder to allow a WML phone to display a summarized

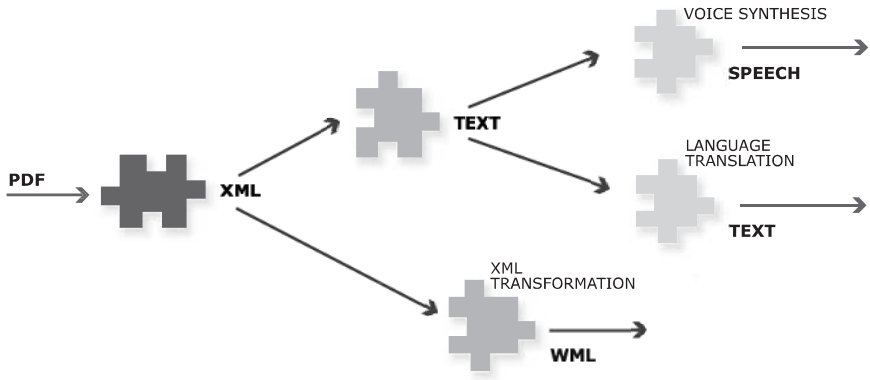


Figure 2.5 Combination of individual transcoders. (*From*: [9]. © 2002 W3C.)

document. The text-to-speech transcoder can be used alone to turn text input into audio output. A language translation transcoder can be combined with the text-to-speech transcoder to turn Spanish text into English speech. A PDF document can be transcoded to text in order to take advantage of a summarization or text-to-speech conversion.

Text-to-speech conversion might be used alongside a transcoder that can convert geographical data of a particular town into *Virtual Reality Modeling Language* (VRML) data in order to navigate a user in the town with an automatically generated audio guide.

2.5.2 Limitations of the Transcoding Framework

Our transcoding framework has two main limitations: (1) the language used by transcoders to express their capabilities is somewhat simplified, and (2) the correct operation of the system depends on the cooperation among the transcoders in setting the properties of the input and output formats. As a result of the first limitation, it is cumbersome for a transcoder to express that it cannot accept input with certain combinations of properties. When a transcoder lists several property-value pairs in a single advertisement, they represent a conjunction of properties. To express disjunction, the properties must be listed in separate advertisements. For example, a transcoder that can accept textual input only in English or German must list all its other restrictions on the input twice, once in conjunction with English, once with German. If a transcoder has several such restrictions on its input, the list of advertised capabilities will quickly become long and unwieldy.

The result of the second limitation of our transcoding framework is that the usefulness of the system as a whole depends on the judicious and correct use by transcoder authors of properties in the input and output formats they advertise. If different transcoders use properties in different ways, or have different policies with respect to when properties should be specified in formats, the system will not function effectively. For example, consider the type application/xscript, which has three different levels, 1, 2, and 3. One transcoder might understand all three levels of xscript, and never make any mention of level in its capabilities advertisement. Another transcoder might only understand levels 1 and 2, and thus advertise that it can accept application/xscript input with the property (''Level,'' ''1'') or (''Level,'' ''2''). These two transcoders could not work together effectively on xscript documents because the output produced by the first transcoder does not specify ''Level'' at all, and therefore cannot be used as input to the second transcoder.

2.5.3 An Implementation of Transcoding Proxies

WTP is an implemented framework for adding intermediary functions to the WWW. WTP provides a Java-based application development environment called *Web Intermediaries* (WBI) [17]. WBI was developed by Maglio and his colleagues at IBM Almaden Research Center and was designed for ease of development and deployment of intermediary applications. Using it, intermediary applications are constructed from four basic building blocks: request editors, generators, document editors, and monitors. They are collectively referred as MEGs (monitors, editors, generators). Monitors observe transactions without affecting them. Editors modify outgoing requests or incoming documents. Generators produce documents in response to requests. WBI dynamically constructs a data path through the various MEGs for each transaction. To configure the data path for a particular request, WBI has a rule associated with each MEG that specifies a Boolean condition indicating whether the MEG should be involved in a transaction based on header information about the request or response. An application (called WBI plug-in) is usually comprised of a number of MEGs that operate in concert to produce a new function. Figure 2.6 shows a WBI plug-in that consists of MEGs. In this figure, an HTTP request from a user client is processed by *request editors* (REs). *Generators* (Gs) accept the modified HTTP request and generate an HTTP response combining an output from a Web server. *Editors* (Es) modify the content of the HTTP response, and *monitors* (Ms) check the processes and generate a log.

Since transcoding is considered as one of intermediary applications, the transcoding framework can be implemented as a plug-in that consists of several MEGs—specifically, the master transcoder, and various specific transcoders (such as GIF-to-JPEG transcoder, or an XML-to-XML converter based on XSL processing, and so forth).

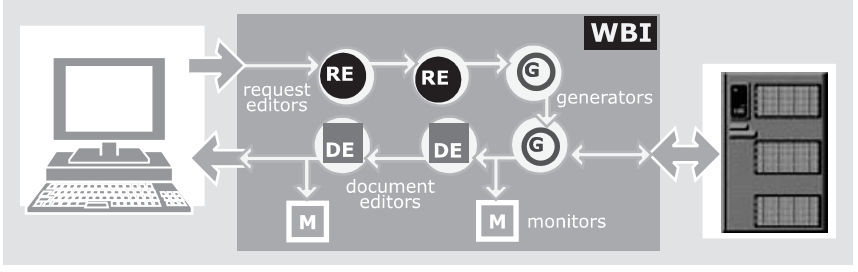


Figure 2.6 WBI plug-in running on a proxy server. (*From*: [9]. © 2002 W3C.)

The master transcoder is a document editor that receives the original object (e.g., a GIF image) as input and produces a modified object (e.g., a JPEG image) as output according to some transcoding requirements. The master transcoder intercepts the data flow between client and server. For each object in the data flow, WTP calls the master transcoder so that it may inspect the request and the original object in order to make an appropriate response. If transcoding is necessary, the master transcoder determines the appropriate transcoder or combination of transcoders. The master transcoder arranges for the appropriate transcoders to be subsequently called by WTP in the correct order.

WBI offers various protocol-specific keywords that allow the specification of rules. During transaction processing, the information available about the transaction (e.g., requested URL, host, content type, and so forth) is matched against the rules of all registered MEGs. If the rule of a MEG is satisfied, the particular MEG will be the next one in the chain to handle the data stream. The master transcoder is registered with WTP in a way that allows it to inspect every request.

In the Java code below:

```
...
MasterTranscoder mt = new MasterTranscoder();
mt.setup("MasterTranscoder", "%true%");
...
```

The special rule ''%true%'' is satisfied in every case, and any MEG specifying this rule is invoked for every request that passes through WTP. Thus, the master transcoder can decide when transcoding is necessary, and if it is, it can then decide on the chain of specific transcoders.

The specific transcoders, which are also document editors, are registered with the master transcoder in order to advertise their capabilities. In addition, the transcoders are also registered with WTP by specifying a rule that determines the conditions under which they will be invoked during a transaction. In addition to the protocol-specific keywords mentioned earlier, an extension mechanism, known as extra rule keys, is available. This mechanism is used by the master transcoder to tell WTP which transcoders to invoke and in what order. Extra rule keys consist of a key-value pair that may be set by any MEG. MEGs can register their rules with WTP so that they will be invoked when another MEG sets a certain value for an extra rule key.

A transcoder that converts GIFs to JPEGs might be registered this way:

```
...
GifToJpegTranscoder g2j = new GifToJpegTranscoder();
g2j.setup( "GIF-To-JPEG", "$GIF-TO-JPEG =%true%", 10);
 ...
```

If the master transcoder determines that the GIF-to-JPEG transcoder should be called, it simply sets the extra rule key condition "$GIF-to-JPEG = %true%".

WTP will then invoke this transcoder to perform the conversion. This obviously works for single-step transcoding, as all the master transcoder must do is set the special condition, and the rest is done by WTP itself.

Things become a little more complicated when the transcoding request can only be satisfied by a combination of transcoders. In this case, the master transcoder must first determine which transcoders to invoke to accomplish the transformation, and the individual transcoders must then be applied in the proper order. To determine which transcoders are needed, the master transcoder considers the input format, the requested output format, and the advertised capabilities of available transcoders. If a single transcoder is available to perform the operation (transforming the input format to the desired output format), it is simply used. If not, the master transcoder searches for a chain of individual transcoders such that (1) the type of the output format of each transcoder matches the type of the input format of the next transcoder in the chain; and (2) each property contained in the input format of a transcoder appears with an identical value (or with the * wildcard) in the output format of a transcoder in the proper place in the chain (or in the input format of the object itself); that is, the most recent instance of the property in the chain must have the correct value.

It can be shown that the overall request is considered satisfied if a hypothetical transcoder with an input format identical to the requested output format can be added to the end of the chain. Thus, this process implements a simple backward-chaining, state-space search in which the goal state is the output format, the initial state is the input format, and the state-transition operators are individual transcoders.

Once an appropriate chain of transcoders is found, there remains the problem of invoking the transcoders in the correct order. This is processed using WTP's transaction data, which lets MEGs associate arbitrary objects with a transaction (HTTP request/response), allowing MEGs later in the processing chain to use objects (information) generated by previous MEGs.

If the transcoding request can only be served by chaining multiple transcoders, the master transcoder simply determines the order of participating operations and stores this information in an object that is then attached to the transcoding request. The master transcoder still sets the condition for the first transcoder in the chain so that WBI can invoke it. The first MEG and each of the following MEGs then set the condition for the next MEG (based on the object stored in the transaction data) until no more MEGs need to be called.

WTP provides various ways for the master transcoder to gather the information it uses to determine the desired output format of an object. One very simple mechanism is to automatically draw conclusions from the HTTP request, such as information about the client that is requesting the data. For example, the following HTTP request could have been issued by a handheld device:

```
GET  http:/www.nagao.nuie.nagoya-u.ac.jp/image.jpg  HTTP/1.0
Accept: */*
User-Agent: Windows  CE
...
```

The master transcoder interprets this as an indication that the client is a device with limited resources for display and connectivity. Of course, there must be a lookup mechanism to identify transcoding operations with a particular device such that the master transcoder can match the device's capabilities, for example, by transcoding each JPEG into a smaller GIF with reduced color depth (monochrome). This saves bandwidth and allows the device to display the image. The UA field is a convenient way to determine standard transcoding operations, such as type conversions or size reduction.

This method can be extended, such that the client specifies allowable or desired transcoding operations in additional HTTP header fields:

```
GET http://www.nagao.nuie.nagoya-u.ac.jp/image.jpg HTTP/1.0
Accept:*/*
...
```

Transcoding-Request:
(image/jpg, ()), (image/gif, ())
Transcoding-Request:
(text/xml, (“DTD”,“a”)), (text/xml, (“DTD”,“b”))
...

In this example, the client specifies explicitly which transcoding operations should be performed through additional header fields

"Transcoding-Request". In the above request, the client asks to transcode each JPEG into a GIF, and to translate XML documents that correspond to the DTD "a" into XML documents that correspond to the DTD "b". WBI provides the necessary infrastructure for each MEG to access protocol-specific header information.

The mechanisms described work for the HTTP protocol but may not work with every HTTP client, much less other protocols, such as *File Transfer Protocol* (FTP) or *Simple Mail Transfer Protocol* (SMTP). If the protocol underlying a request does not offer such functionality, clients can simply register with the transcoding proxy and maintain a client or device profile. These profiles are made accessible to the master transcoder, such that it only needs to determine the client during a transaction to perform a lookup and decide whether a transcoding operation is necessary. Of course, such a service requires the transcoding intermediary to provide a mechanism for clients to register their transcoding needs. Other implementations might make transcoding decisions in different ways. For example, rather than having clients register with the transcoder, the transcoder could present a simple form-based interface that would allow a user to request transcoded objects on a per-request basis, or a specific intermediary might always make certain kinds of transcoding decisions based on a mandate to preserve bandwidth.

2.5.4 Technical Issues of Transcoding

There are many opportunities for future work. The current architecture for transcoding does not support certain computational optimizations. The framework could be extended to allow transcoders to announce the quality of their output or their consumption of computing resources. This would enable the transcoding engine to do a better job of optimizing the data flow through a series of transcoders.

Another direction is to more carefully consider the details of how the master transcoder derives the path through the pool of available transcoders. One can imagine many situations in which more than one chain of transcoders might satisfy a request. How does the master transcoder decide which path to take? Dynamic measurements of the past performance of each transcoder could be considered, or information as to whether type or other conversions are known to be lossy.

In addition, many enhancements can be made to the language that describes transcoder capabilities and requests. A more expressive way of describing patterns might be useful, for instance, one that enables the system

to match ranges of numbers. Currently, each transcoder is free to choose any name it wishes for the properties specified in its input and output formats, leading to the possibility of name space collisions. A mechanism for avoiding such collisions is needed, such as creating a standard ontology for properties.

2.5.5 Deployment Models of Transcoding

Transcoding must be interposed at some point between the generation of the initial content and the final rendering on the client device. Three locations are available in the current Web architecture:

1. The client device;
2. An intermediate point in the network, such as a proxy server;
3. The source of the content.

Figure 2.7 shows a simplified picture of a network configuration and the potential transcoding placement locations. Each has its strengths and weaknesses. Therefore, combinations of the approaches may be useful in some situations.

The Web experience today is still largely oriented toward a client model that assumes a high-bandwidth network connection and workstation capabilities. Content that is designed based on these assumptions is not usually appropriate for display on lower-capability devices or for delivery on low-bandwidth or high-cost connections. This is the crux of the problem when performing client-side transcoding. When we first started looking at

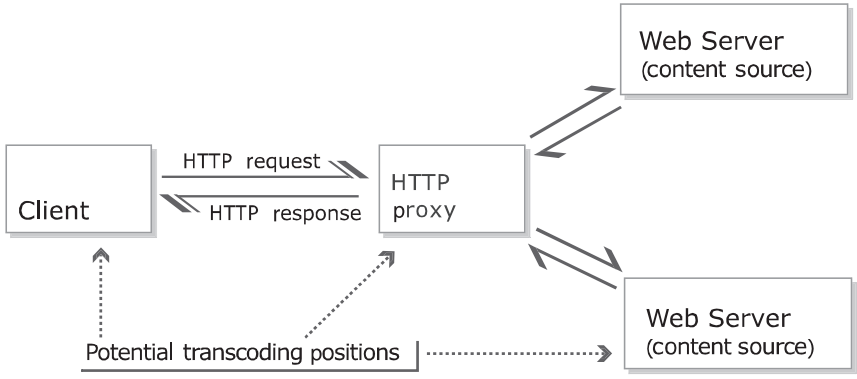


Figure 2.7 Potential transcoding positions in network. (*From*: [7]. © 2001 IBM.)

client-side transcoding options for WTP, we tried visiting a variety of sites on the Web. It could and did take 20 minutes or more to view some pages using a dial-up or wireless connection. Even over faster links, the very limited rendering speeds and capabilities of handheld devices will usually make pure client-side transcoding an unworkable solution. Image transcoding can be particularly problematic for clients because of its computationally intensive nature, large image sizes, and the potentially very high compression rates.

Independent of the transcoding placement issue, there still needs to be a Web-capable presence on the client device. Additional client-side presence can be useful in providing transcoding-specific user options (such as image-specific transcoding options) or a controlled client environment. Such a client can also provide additional information about client device capabilities, although emerging standards such as *composite capability/preference profiles* (CC/PP) are expected to play this role.

CC/PP is a way to specify what exactly a user agent is capable of doing. This allows for sophisticated content negotiation techniques between Web servers and clients to produce optimized XML-based markup for display and use on a wide variety of user agents. CC/PP is an XML-based framework for describing and managing software and hardware profiles that include information on the user agent's capabilities (physical and programmatic); the user's specified preferences within the user agent's set of options; and specific qualities about the user agent that can affect content processing and display, such as physical location.

CC/PP is designed to work with a wide variety of Web-enabled devices, from PDAs to desktop machines to laptops to WAP phones to phone browsers to Web television units to specialized browsers for users with disabilities. Proxies may also be used to provide markup transformation, transmission, or caching services for CC/PP-enabled clients and servers. The CC/PP framework provides a way to describe generic profiles accessible via the Web—for example, from the hardware or software vendor—reducing the amount of information that must be directly sent from the user agent itself, an important consideration for limited-bandwidth cellular modem technologies.

WTP assumes only that there is Web-rendering software of some sort on the client device and that the HTTP headers generated by the device either specify the capabilities of the device or allow it to be inferred. Currently available Web-capable devices meet this requirement through the use of user agent and other descriptive headers. This allows WTP to effectively transcode for a wide variety of devices without additional client presence.

A second deployment option is at an intermediate point in the network as mentioned above, such as a proxy. In this model, all client requests are directed to the intermediate representative that forwards them on to the specified destination server. Most Web clients support use of proxies for at least HTTP. WTP can be configured to run as an HTTP proxy, and in this mode can transparently transcode content for these clients.

In the proxy model, all HTTP responses flowing to the client can be transcoded, regardless of the server producing the content. This approach, combined with the preference-based transcoding of WTP, allows new client devices to be easily added to a network. For example, a business that wishes to give employees with WAP phones access to internal Web content can do so by using a WAP gateway (for protocol conversion) and a WTP proxy (for content transcoding), without requiring content to be specifically authored for the phones, and without impacting servers generating content.

There are some drawbacks to the proxy approach for transcoding. The use of encryption such as *Secure Sockets Layer* (SSL) to protect data precludes transcoding via a proxy. This drawback could be addressed by allowing the proxy to act as both an endpoint and a source for SSL connections. However, this additional point of exposure may not be acceptable to some users. It also imposes a significant performance penalty at the proxy.

Another potential concern of the proxy-based approach is legal, rather than technical. Some providers of content also wish to control the presentation of this content, and so will not want it to be transcoded outside their control. Although this restriction will not be a concern in the corporate intranet, it can limit deployment in the external Internet environment.

This type of transcoding raises many intellectual property issues. For example, is it legal to change the format of content owned or copyrighted by others? It is possible that the formal type-transforming and property-transforming distinctions made in our transcoding framework can be used in determining whether copyrighted content has actually been modified. Though such legal issues are clearly beyond the scope of this book, we feel certain that these will be addressed by legislatures and courts in the near future, for the legal battle has already begun: a group of on-line publishers has sought to stop one transcoding service from modifying its copyrighted content.

A final option is to provide transcoding at the source of the data. This option, content source transcoding, may be attractive for a number of reasons. It requires no client configuration since there is now no intervening proxy. For clients that do not support HTTP proxies, such as some WAP phones, it may

be the only workable option. Since the transcoding is coresident with the content, the content provider has tighter control over what is transcoded and how it is presented. Content source transcoding allows transcoding to be done before content is encrypted, and so it can support secure connections. If transcoding is being done to control who sees what data (for example, only allowing users from certain locations full access to data), performing transcoding at the source enforces these views.

The major drawback to content source transcoding is its limited scope. Deploying transcoding in the content source only affects targeted content from that server. It is not appropriate when a client is viewing content from a range of servers. Performing transcoding on the content source server will also impose an additional load on the resources of the server, and so may require additional infrastructure enhancements.

2.6 More Advanced Transcoding

Transcoding is a good way to create more adaptive and accessible content by match-making between user's requirements with preferences and original content. Considering reuse of content, more advanced transcoding is needed such as gathering and integration of multiple content (e.g., multidocument summarization) and a hypertext generation from a set of semantically related content. To do that, semantic descriptions about content are required. Since there are still big obstacles to automatic analysis of semantic features of content, human power should be involved in this important task. The next chapter considers metalevel information of digital content called annotation. Using annotation, content can be extended to be provided some semantic information.

An advanced transcoding is represented as an operation that takes multiple input data objects $d_{in_1}, \ldots, d_{in_n}$ (n is a particular integer) with their corresponding formats $f_{in_1}, \ldots, f_{in_n}$ and converts them into an output data object d_{out} in the specified format f_{out}. Thus, we denote a transcoding operation by $(\{d_{in_1}(f_{in_1}), \ldots, d_{in_n}(f_{in_n})\}, f_{out}) \rightarrow d_{out}(f_{out})$.

Any two types of the input formats $f_{in_i}, f_{in_j} (i \neq j)$ and the type of the output format f_{out} may be identical.

An example of the output of the advanced transcoding is embedding or linking of some related content such as comments, notes, and dictionary definitions with the requested content.

Transquotation proposes a way for quoting hypertext documents [18]. In its framework, the materials appear on the Web content, but the quoter

does not deliver the materials at all, even though they look that way on the resulting document. While quoting means importing other documents as they are, the transcoding approach is to convert other documents in a way that they fit into the original context, preserving the semantics and improving the readability at the same time. As the result of embedding, there are no windows hiding any part of the original document, which makes the context easy to follow, and the new document is ready to be used for further transcoding.

The basic idea of transquotation is to make it possible to freely quote any kind of content while reading a hypertext document on-line. For example, when an author wants to create his or her own Web content, rather than directly including photographs or text from other content, you can make a document where these related contents can be shown. This idea was based on the inventor's desire to find a way of expediting the confusing situation relating to copyright. If there is a clear distinction between the original and the copy of the quoted material, there should not be any problem, but although there are things that can definitely be identified as originals, there are many kinds of data that one cannot prove to be one's own. These materials are currently mixed together on the Internet. It seems necessary to further discuss the fundamental issues of original and copyright.

Using the advanced transcoding system, users can automatically browse the kind of screen they require via the browser and there is no need for these quoted materials to be redistributed. Just by clicking on the screen, the users can also go to the authorization page, where the supplier of the material gives the user permission to transquote.

Transcoding is a starting point of creating a new architecture of on-line digital content. Later in this book, this topic will be revisited with more concrete and practical applications.

References

[1] W3C, "HyperText Markup Language Home Page," http://www.w3.org/MarkUp/, 2002.

[2] W3C, "XHTML 1.0: The Extensible HyperText Markup Language (Second Edition)," http://www.w3.org/TR/2001/WD-xhtml1-20011004/, 2002.

[3] Wireless Application Protocol Forum, "Wireless Markup Language Version 2.0," http://www1.wapforum.org/tech/documents/WAP-238-WML-20010626-p.pdf, 2001.

[4] W3C, "Voice Extensible Markup Language (VoiceXML) Version 2.0," http://www.w3.org/TR/2001/WD-voicexml20-20011023/, 2001.

[5] W3C, "Synchronized Multimedia," http://www.w3.org/AudioVideo/, 2002.

[6] W3C, "XSL Transformations (XSLT) Version 1.0," http://www.w3.org/TR/xslt, 1999.

[7] Britton, K. H., et al., "Transcoding: Extending E-Business to New Environments," *IBM Systems Journal*, Vol. 40, No. 1, 2001, pp. 153–178.

[8] The Apache Software Foundation, "Xalan-Java XSLT Processor," http://xml.apache.org/xalan-j/, 2002.

[9] W3C, "Web Accessibility Initiative (WAI)," http://www.w3.org/WAI/, 2002.

[10] Asakawa, C. and H. Takagi, "Annotation-Based Transcoding for Nonvisual Web Access," *Proc. the Fourth International ACM Conference on Assistive Technologies (ASSETS 2000)*, 2000, pp. 172–179.

[11] Brusilovsky, P., E. Schwarz, and G. Weber, "A Tool for Developing Adaptive Electronic Textbooks on WWW," *Proc. WebNet'96 - World Conference of the Web Society*, 1996, pp. 64–69.

[12] Milosavljevic, M., et al., "Virtual Museums on the Information Superhighway: Prospects and Potholes," *Proc. CIDOC'98, the Annual Conference of the International Committee for Documentation of the International Council of Museums*, 1998, pp. 10–14.

[13] Fink, J., A. Kobsa, and A. Nill, "Adaptable and Adaptive Information Provision for All Users, Including Disabled and Elderly People," *New Review of Hypermedia and Multimedia*, Vol. 4, 1998, pp. 163–188, http://zeus.gmd.de/kobsa/papers/1998-NRMH-kobsa.ps.

[14] BBC Education, "Betsie Home Page," http://www.bbc.co.uk/education/betsie/, 2002.

[15] ALPAS Solutions, "DiffWeb," http://www.diffweb.com/, 2002.

[16] Vorburger, M., "ALTifier Web Accessibility Enhancement Tool," http://www.vorburger.ch/projects/alt/, 1999.

[17] Ihde, S. C., et al., "Intermediary-Based Transcoding Framework," *IBM Systems Journal*, Vol. 40, No. 1, 2001, pp. 179–192.

[18] Nelson, T. H., "Transcopyright: Dealing with the Dilemma of Digital Copyright," *Educom Review*, Vol. 32, No. 1, 1997, pp. 32–35.

3

Annotation: A Technique to Extend Digital Content

This chapter gives a survey on representation and application of metalevel information of digital content including on-line text and video. In this book, such metainformation is called *annotation* in general.

In ordinary language, annotation means a sort of commentary or explanation, or the act of producing such a commentary. Like ''markup,'' this term's ordinary meaning plausibly covers the extra information of textual descriptions, such as an etymology of a word and a historical account of a context. Some speech and language engineers have begun to use the term ''annotation'' to mean a linguistic analysis of text, such as the annotation of syntactic structure or of coreference, but there is not yet a specific, widely accepted technical meaning.

It is reasonable to generalize this term to cover any kinds of metacontent of content, by thinking of annotation as the provision of any symbolic description of particular portions of a preexisting object. If the object is a speech recording, then an ordinary orthographic transcription is certainly a kind of annotation in this sense.

While the related term ''metadata'' is more commonly used in the information technology field, I use the term ''annotation'' because it means more than ''data about data'' in some restricted format and emphasizes that it is an ''extra information about meaning or context of content.'' This chapter begins with an introduction of some metadata-related activities.

Metadata is widely recognized as the most promising solution for making sense of the explosion of materials being made available on

the World Wide Web. Metadata tackles problems related to the description of content and improving the precision of information retrieval. Although the term "metadata" is often associated with Internet resources, libraries have created metadata about bibliographic materials for most of their existence, conveying rules for its creation through several generations of cataloguing codes.

Currently, Web authors insert metadata into their documents through the use of a "META" tag in the HTML. A META tag can have a name and a content attribute with the value for the content attribute determined by the Web author. In the example below, the META tag provides a general description of this chapter and lists keywords for retrieval by the search engines.

```
<META name="description" content="This document describes a
    comprehensive survey on annotation and metadata">
<META name="keywords" content="annotation, metadata, digital
    content, XML, WWW">
```

For an Internet user, metadata becomes most useful if it contains enough description of a resource to allow a user to assess its potential utility without having to retrieve it or click a link to it. Unlike a cataloguer, Web authors who insert META tags into their documents often will not be conversant with controlled subject schemes, nor will they be prepared to record the relationships between their creations and other resources. For this reason, there is tremendous interest in metadata formats that do not require specialist knowledge to create or maintain them.

3.1 Frameworks of Metadata

There are several frameworks of metadata that are working or being discussed. They will now be introduced since they are helpful to understand the concept of metadata, which will be called annotations later in this chapter.

3.1.1 Dublin Core

One of the most promising general metadata formats is the Dublin Core metadata set designed by a broadly representative group from the library, research, and academic communities, as well as from industry [1]. It takes its name from a workshop hosted by *Online Computer Library Center, Inc.* (OCLC) in Dublin, Ohio, in March 1995.

The *Dublin Core Metadata Initiative* (DCMI) is an open forum engaged in the development of interoperable on-line metadata standards that

support a broad range of purposes and business models. DCMI's activities include consensus-driven working groups, global workshops, conferences, standards liaison, and educational efforts to promote widespread acceptance of metadata standards and practices.

DCMI has defined the Dublin Core metadata set that consists of the following 15 elements:

1. *Title:* A name given to the resource. Typically, a title will be a name by which the resource is formally known.
2. *Creator:* An entity primarily responsible for making the content of the resource. Examples of a creator include a person, an organization, or a service. Typically, the name of a creator should be used to indicate the entity.
3. *Subject:* The topic of the content of the resource. Typically, a subject will be expressed as keywords, key phrases, or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.
4. *Description:* An account of the content of the resource. Description may include but is not limited to an abstract, table of contents, reference to a graphical representation of content, or a free-text account of the content.
5. *Publisher:* An entity responsible for making the resource available. Examples of a publisher include a person, an organization, or a service. Typically, the name of a publisher should be used to indicate the entity.
6. *Contributor:* An entity responsible for making contributions to the content of the resource. Examples of a contributor include a person, an organization, or a service. Typically, the name of a contributor should be used to indicate the entity.
7. *Date:* A date associated with an event in the life cycle of the resource. Typically, date will be associated with the creation or availability of the resource. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 and follows the YYYY-MM-DD format.
8. *Type:* The nature or genre of the content of the resource. Type includes terms describing general categories, functions, genres, or aggregation levels for content. Recommended best practice is to select a value from a controlled vocabulary (for example, the working draft list of Dublin Core types). To describe the physical

or digital manifestation of the resource, use the FORMAT element.

9. *Format:* The physical or digital manifestation of the resource. Typically, format may include the media type or dimensions of the resource. Format may be used to determine the software, hardware, or other equipment needed to display or operate the resource. Examples of dimensions include size and duration. Recommended best practice is to select a value from a controlled vocabulary—for example, the list of Internet media types (i.e., MIME) defining computer media formats.
10. *Identifier:* An unambiguous reference to the resource within a given context. Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Example formal identification systems include *Universal Resource Identifier* (URI) (including URL) [2], the *Digital Object Identifier* (DOI) [3], and *International Standard Book Number* (ISBN) [4].
11. *Source:* A reference to a resource from which the present resource is derived. The present resource may be derived from the source resource in whole or in part. Recommended best practice is to reference the resource by means of a string or number conforming to a formal identification system.
12. *Language:* A language of the intellectual content of the resource. Recommended best practice for the values of the language element is defined by RFC 1766 which includes a two-letter language code (taken from the ISO 639 standard). For example, ''en'' for English, ''fr'' for French, or ''ja'' for Japanese.
13. *Relation:* A reference to a related resource. Recommended best practice is to reference the resource by means of a string or number conforming to a formal identification system.
14. *Coverage:* The extent or scope of the content of the resource. Coverage will typically include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the *Thesaurus of Geographic Names*) and that, where appropriate, named places or time periods be used in preference to numeric identifiers such as sets of coordinates or date ranges.
15. *Rights:* Information about rights held in and over the resource. Typically, a rights element will contain a rights management

statement for the resource, or reference a service providing such information. Rights information often encompasses (intellectual property rights IPRs), copyright, and various property rights. If the Rights element is absent, no assumptions can be made about the status of these and other rights with respect to the resource.

These elements are straightforward enough to be generated by authors themselves, yet they can be expanded to include extra levels of information as desired. For example, here are the META tags for this article expanded to include the use of some Dublin Core elements.

```
<META name="dc.title" content="Digital Content Technology">
<META name="dc.author" content="Katashi Nagao">
<META name="dc.author" content="(TYPE=email)
                                nagao@nuie.nagoya-u.ac.jp">
```

3.1.2 Warwick Framework

The relative simplicity of the Dublin Core elements combined with their flexibility makes them well suited to a wide variety of network resources. It was recognized, however, that other metadata element sets may be appropriate for more specialized resources. In 1996 OCLC and the U.K. Office for Library and Information Networking hosted a second metadata workshop to broaden the scope of the Dublin Core. The workshop participants proposed an infrastructure called the Warwick Framework designed to support any metadata element set [5].

The Warwick Framework has two fundamental components. A *container* is the unit for aggregating the typed metadata sets, which are known as *packages*.

A container may be either transient or persistent. In its transient form, it exists as a transport object between and among repositories, clients, and agents. In its persistent form, it exists as a first-class object in the information infrastructure. That is, it is stored on one or more servers and is accessible from these servers using a globally accessible identifier (e.g., URI). Note that a container may also be wrapped within another object (i.e., one that is a wrapper for both data and metadata). In this case the wrapper object will have a URI rather than the metadata container itself.

Each package is a typed object; its type may be inferred after access by a client or agent. Packages are of three types:

1. *Metadata set:* These are packages that contain actual metadata. A potential problem is the ability of clients and agents to recognize and process the semantics of the many metadata sets. In addition, clients and agents will need to adapt to new metadata types as they are introduced. Initial implementations of the Warwick Framework will probably include a set of well known metadata sets, in the same manner that most Web browsers have native handlers for a set of well-known MIME types. Extending the framework implementations to handle extensible metadata sets will rely on a type registry scheme.
2. *Indirect:* This is a package that is an indirect reference to another object in the information infrastructure. While the indirection could be done using URLs, we emphasize that the existence of a reliable URI implementation is necessary to avoid the problems of dangling references. We note three possibly obvious but important points about this indirection. First, the target of the indirect package is a first-class object, and thus may have its own metadata and, significantly, its own terms and conditions for access. Second, the target of the indirect package may also be indirectly referenced by other containers (i.e., sharing of metadata objects). Finally, the target of the indirection may be in a different repository or server than the container that references it.
3. *Container:* This is a package that is itself a container. There is no defined limit for this recursion.

The mechanisms for associating a Warwick Framework container with an information resource depend on the implementation of the framework.

An example of association is the reverse linkage that ties a container to a piece of content. Anyone can, in fact, create descriptive data for a networked resource, without permission or knowledge of the owner or manager of that resource. This metadata is fundamentally different from that metadata that the owner of a resource chooses to link or embed with the resource. The designers of the Warwick Framework, therefore, informally distinguished between two categories of metadata containers, which both have the same implementation:

1. *An internally referenced metadata container:* This is the metadata that the author or maintainer of a resource has selected as the preferred description(s) for the content. This metadata is associated with the content by either embedding it as part of the structure

that holds the content or referencing it via a URI. An internally referenced metadata container referenced via a URI is, by nature, a first-class networked object and may have its own metadata container associated with it. In addition, an internally referenced metadata container may back-reference the content that it describes via a linkage metadata element within the container.

2. *An externally referenced metadata container:* This is metadata that has been created and is maintained by an authority separate from the creator or maintainer of a resource. In fact, the creator of the resource may not even be aware of this metadata. There may be an unlimited number of such externally referenced metadata containers. For example, libraries, indexing services, ratings services, and the like may compose sets of metadata for content objects that exist on the Internet. As we stated earlier, these externally referenced metadata containers are themselves first-class network objects, accessible through a URI and having some associated metadata. The linkage to the content that one of these externally referenced containers purports to describe will be via a linkage metadata element within the container. There is no requirement, nor is it expected, that the content object will reference these externally referenced containers in any way.

3.1.3 Resource Description Framework

The W3C is also actively involved in fostering the deployment of metadata for Web resources. W3C has been developing a specification called the *Resource Description Framework* (RDF), which draws on the Warwick Framework and is designed to support the use and exchange of metadata on the Web. RDF provides infrastructure for letting Web authors and others describe resources and collections of resources so that these resources can be better indexed and used in a wide variety of tools, from graphical overviews of document repositories to software agents traversing the Web for resources on particular topics.

RDF is an application of XML that was specifically designed for Web applications. XML gives RDF a tag-based syntax for describing metadata that is somewhat similar to HTML:

```
<?xml:namespace ns="http://www.w3.org/RDF/"prefix="rdf"?>
<?xml:namespace ns="http://purl.oclc.org/DC/"prefix="dc"?>
```

```
<rdf:RDF>
<rdf:Description
  rdf:HREF="http://www.artechhouse.com/newbooks/dct.xml">
<dc:Title>Digital Content Technology</dc:Title>
</rdf:Description>
</rdf:RDF>
```

In the example above, the RDF metadata conventions are used to describe the title of a document. The title is marked in by the tags <dc:title> and </dc:title>. The code "dc" specifies a particular schema or authority for the semantics and conventions of the property type or element employed in the description. In this case, dc represents a schema for Dublin Core metadata. The schema can be found at the Web location specified at the beginning of the metadata as the namespace, an XML mechanism for identifying the governing authority for a vocabulary. In this case, the location of the schema is http://purl.oclc.org/DC/. The schema is the source to refer to for the meaning and proper use of the element specified in this example, the element "Title."

Schemas can range from the very simple to the very complex. RDF also allows schemas to be combined. For example, when creating metadata for a Web resource on mathematics, we could use the Dublin Core elements along with a schema specifically designed for describing mathematical concepts.

3.1.3.1 RDF in Search Engines

Despite the interest in metadata within the Internet research community, the quality and quantity of metadata on the Web varies greatly. The chaotic nature of Web-based publishing makes almost any standard mechanism for describing Web resources difficult to implement. Web authors have also demonstrated little interest in creating metadata, due to the lack of enthusiasm that maintainers of Internet search engines have shown towards self-assigned descriptions of resources. Most search engines do very little with metadata, in part because experience has shown that Web authors occasionally cannot be trusted to describe their creations accurately and sometimes blatantly use misleading descriptions in order to increase the odds that they will be noticed.

RDF may have a dramatic impact on this situation. RDF, which provides a standard means for designating metadata, can be incorporated into many different kinds of Web-publishing tools. This would allow Web authors to create appropriate metadata as an integral part of putting resources on the Web. RDF also allows for metadata to be digitally certified.

Internet search engines, software agents, and other kinds of information discovery software could fully exploit verified metadata from a trusted source.

3.1.3.2 RDF Features

RDF, as its name implies, is a framework for describing and interchanging metadata. It is built on the following rules:

- A resource is anything that can have a URI; this includes all the Web documents, as well as individual elements of an XML document.
- A property is a resource that has a name, for example, author or title. In many cases, all we really care about is the name; but a property needs to be a resource so that it can have its own properties.
- A statement consists of the combination of a resource, a property, and a value. These parts are known as the subject, predicate, and object of a statement.

  An example statement is ''The Author of http://www.ice.nuie.nagoya-u.ac.jp/~nagao/ is Katashi Nagao.'' The value can just be a string (for example, ''Katashi Nagao'' in the example) or it can be another resource (for example, ''The main page of http://www.ice.nuie.nagoya-u.ac.jp/~nagao/ is http://www.nuie.nagoya-u.ac.jp/'').

There is a straightforward method for expressing these abstract properties in XML, for example:

```
<rdf:Description
    about="http://www.ice.nuie.nagoya-u.ac.jp/~nagao/">
<Author>Katashi Nagao</Author>
<MainPage rdf:resource="http://www.nuie.nagoya-u.ac.jp/"/>
</rdf:Description>
```

RDF is carefully designed to have the following characteristics.

1. *Independence:* Since a property is a resource, any independent organization (or even person) can invent them. I can invent one called Author, and you can invent one called Director, and someone else can invent one called Creator.
2. *Interchange:* Since RDF statements can be converted into XML, they are easy for us to interchange. This would probably be necessary.

3. *Scalability:* RDF statements are simple, three-part records (resource, property, value), so they are easy to handle and look things up by, even in large numbers. The Web is already big and getting bigger, and we are probably going to have (literally) billions of these floating around (millions even for a big intranet). Scalability is important.
4. *Properties are resources:* Properties can have their own properties and can be found and manipulated like any other resource. This is important because there are going to be lots of them; too many to look at one by one. For example, I might want to know if anyone out there has defined a property that describes the genre of a movie, with values like Comedy, Horror, Romance, and Thriller. I would need metadata to help with that.
5. *Values can be resources:* For example, most Web pages will have a property named Home-Page, which points at the home page of their site. So the values of properties, which obviously have to include things like title and author's name, also have to include resources.
6. *Statements can be resources:* Statements can also have properties. Since nobody can provide useful assertions for all the resources, and since the Web is way too big for us to provide our own, we are going to need to do lookups based on other people's metadata. This means that we will want, given any statement such as "The Subject of this book is content technology," to be able to ask "Who said so?" and "When?" One useful way to do this would be with metadata; so statements will need to have properties.

XML allows Web authors to invent tags, which may contain both text data and other tags. XML has a built-in distinction between element types, (e.g., the IMG element type in XHTML) and elements (e.g., an individual `<img src="portrait.jpg"/>`); this corresponds naturally to the distinction between properties and statements. So it seems as though XML documents should be a natural vehicle for exchanging general purpose metadata.

XML, however, falls apart on the scalability design goal. There are two problems:

1. The order in which elements appear in an XML document is significant and often very meaningful. This seems highly unnatural in the metadata world. Who cares whether a movie's director or title is listed first, as long as both are available for lookups?

Furthermore, maintaining the correct order of millions of data items is expensive and difficult, in practice.

2. When you represent general XML documents in computer memory, you get weird data structures that mix trees, graphs, and character strings. In general, these are hard to handle in even moderate amounts, let alone by the billion.

On the other hand, something like XML is an absolutely necessary part of the solution to RDF's interchange design goal. XML is unequalled as an exchange format on the Web. But by itself, it does not provide what is needed in a metadata framework.

3.1.3.3 Vocabularies

RDF provides a model for metadata, and a syntax so that independent parties can exchange it and use it. What it does not provide, though, is any properties of its own. RDF does not define Author or Title or Director or Business-Category. It is a job for everyone.

It seems unlikely that one property standing by itself is apt to be very useful. It is expected that these will come in packages; for example, a set of basic bibliographic properties like Author, Title, Date, and so forth. Then a more elaborate set from OCLC and a competing one from the Library of Congress. These packages are called vocabularies; it is easy to imagine property vocabularies describing books, videos, foods, drinks, and so on.

3.1.3.4 RDF Contributions

The Web is too big for any one person to stay on top of. In fact, it contains information about a huge number of subjects, and for most of those subjects (such as fine wines, home improvement, and cancer therapy), the Web has too much information for any one person to stay on top of and do much of anything else.

This means that opinions, pointers, indexes, and anything that helps people discover things are going to be commodities of very high value. Nobody thinks that everyone will use the same vocabulary (nor should they), but with RDF we can have a marketplace in vocabularies. Anyone can invent them, advertise them, and sell them. The good (or best-marketed) ones will survive and prosper. Probably most niches of information will come to be dominated by a small number of vocabularies, the way that library catalogs are today.

Even among people who are sharing the use of metadata vocabularies, there is no need to share the same software. RDF makes it possible to use

multiple pieces of software to process the same metadata and to use a single piece of software to process (at least in part) many different metadata vocabularies.

With many human efforts, this should make the Web more like a library, or a video store, a phone book, than it is today.

3.1.3.5 RDF Schema

RDF provides interoperability between applications that exchange machine-understandable information on the Web. RDF uses XML to exchange descriptions of Web resources, but the resources, being described can be of any type, including XML and non-XML resources. RDF emphasizes facilities to enable automated processing of Web resources.

Descriptions used by these applications can be modeled as relationships among Web resources. The RDF data model defines a simple means for describing interrelationships among resources in terms of named properties and values. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources. As such, the RDF data model can therefore resemble an entity-relationship diagram. The RDF data model, however, provides no mechanisms for declaring these properties, nor does it provide any mechanisms for defining the relationships between these properties and other resources. That is the role of RDF Schema.

Resource description communities require the ability to say certain things about certain kinds of resources. For describing bibliographic resources, for example, descriptive attributes including author, title, and subject are common. For digital certification, attributes such as checksum and authorization are often required. The declaration of these properties (attributes) and their corresponding semantics are defined in the context of RDF as an RDF Schema. A schema defines not only the properties of the resource (e.g., title, author, subject, size, and color) but may also define the kinds of resources being described (books, Web pages, people, and companies).

3.1.3.6 RDF Site Summary

RDF Site Summary (RSS) is a multipurpose extensible metadata description and syndication format [6]. RSS is an XML application, conforming to the RDF specification. RSS is extensible via XML namespace and/or RDF based modularization.

An RSS summary, at a minimum, is a document describing a channel consisting of URL-retrievable items. Each item consists of a title, link,

and brief description. While items have traditionally been news headlines, RSS has seen much repurposing in its short existence.

The following is a basic sample RSS 1.0 document, making use of only the core RSS 1.0 element set.

```
<?xml version="1.0"?>

<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\#"
     xmlns="http://purl.org/rss/1.0/">

<channel rdf:about="http://www.xml.com/xml/news.rss">
     <title>XML.com</title>
     <link>http://xml.com/pub</link>
     <description>
       XML.com features a rich mix of information and services
       for the XML community.
     </description>

<image
     rdf:resource="http://xml.com/universal/images/xml_tiny.
     gif"/>

 <items>
  <rdf:Seq>
     <rdf:li
       resource="http://xml.com/pub/2000/08/09/xslt/
       xslt.html"/>
     <rdf:li
       resource="http://xml.com/pub/2000/08/09/
       rdfdb/index.html"/>
  </rdf:Seq>
 </items>

</channel>

<image rdf:about="http://xml.com/universal/images/
     xml_tiny.gif">
      <title>XML.com</title>
     <link>http://www.xml.com</link>
```

```
     <url>http://xml.com/universal/images/xml_tiny.gif</url>
</image>

<item rdf:about="http://xml.com/pub/2000/08/09/xslt/
     xslt.html">
     <title>Processing Inclusions with XSLT</title>
     <link>http://xml.com/pub/2000/08/09/xslt/xslt.html</link>
     <description>
      Processing document inclusions with general XML tools can be
      problematic. This article proposes a way of preserving
      inclusion information through SAX-based processing.
     </description>
</item>

<item rdf:about="http://xml.com/pub/2000/08/09/rdfdb/
     index.html">
     <title>Putting RDF to Work</title>
     <link>http://xml.com/pub/2000/08/09/rdfdb/index.html
     </link>
     <description>
      Tool and API support for the Resource Description Framework
      is slowly coming of age. Edd Dumbill takes a look at RDFDB, one
      of the most exciting new RDF toolkits.
     </description>
</item>
</rdf:RDF>
```

As RSS continues to be repurposed, aggregated, and categorized, the need for an enhanced metadata framework grows. Channel- and item-level title and description elements are being overloaded with metadata and HTML. Some producers are even resorting to inserting unofficial ad hoc elements (e.g., `<category>`, `<date>`, `<author>`) in an attempt to augment the sparse metadata facilities of RSS.

One proposed solution is the addition of more simple elements to the RSS core. This direction, while possibly being the simplest in the short run, sacrifices scalability and requires iterative modifications to the core format, adding requested and removing unused functionality.

A second solution, and the one adopted here, is the compartmentalization of specific functionality into the pluggable RSS modules. This is one of the approaches used in this specification: modularization is achieved by using

XML namespaces for partitioning vocabularies. Adding and removing RSS functionality is then just a matter of the inclusion of a particular set of modules best suited to the task at hand. No reworking of the RSS core is necessary.

Advanced applications of RSS are demanding richer respresentation of relationships between intra- and interchannel elements (e.g., threaded discussions). While RDF provides a framework for just such rich metadata modeling, RSS provides an instance of RDF-based metadata language with which to layer further structure.

3.1.4 MPEG-7

MPEG-7 is an ISO/IEC standard developed by ISO/IEC JTC1 SC29/WG11, which is widely referred as the *Moving Picture Experts Group* (MPEG), the committee that also developed the standards known as MPEG-1, MPEG-2, and MPEG-4 [7]. MPEG-1 and MPEG-2 standards made possible interactive video on CD-ROM and digital television. MPEG-4 is the multimedia standard for the fixed and mobile Web enabling integration of multiple paradigms.

MPEG-7, formally named *Multimedia Content Description Interface*, is a standard for describing the multimedia content that supports some degree of interpretation of the meaning of the content, which can be passed onto, or accessed by, a device or a computer program. MPEG-7 is not aimed at any one application in particular; rather, the elements that MPEG-7 standardizes support as broad a range of applications as possible.

MPEG-7 addresses applications that can be stored or streamed (e.g., broadcast, push delivery models on the Internet), and can operate in both real-time and non real-time environments. A real-time environment in this context means that the description is generated while the content is being captured.

MPEG-7 makes searching the Web for multimedia content as easy as searching for text documents. The MPEG-7 standard will be particularly useful in large content archives, which the public can now access, and in multimedia catalogs in which consumers identify content for purchase.

Semantic information of content may also be used by agents, for the selection and filtering of broadcast material or for personalized advertising. Further, MPEG-7 descriptions will allow fast and cost-effective use of the underlying content by enabling semi-automatic multimedia presentation and editing.

MPEG-7 is the metadata standard, based on XML technology, for describing features of multimedia content and providing the most

comprehensive set of audio-visual description tools available. These description tools are based on Dublin Core-like metadata elements (i.e., title, creator, and rights): semantic (the who, what, when, and where information about objects and events); and structural (the measurement of the amount of color associated with an image or the timbre of a recorded instrument) features of the audio-visual content. They build on the audio-visual data formats defined by MPEG-1, 2, and 4.

MPEG-7 lets users find the content needed. It provides a flexible and extensible framework for describing audio-visual data, by defining a multimedia library of methods and tools. It standardizes:

1. *A set of descriptors:* A descriptor is a representation of a feature that defines the syntax and semantics of the feature representation.
2. *A set of description schemes (DSs):* A DS specifies the structure and semantics of the relationships between its components, which may be both descriptors and description schemes.
3. *A language that specifies description schemes, the description definition language (DDL):* It also allows for the extension and modification of existing DSs. MPEG-7 adopted a language for XML Schema [8] as the MPEG-7 DDL. XML Schema expresses shared vocabularies and allows machines to carry out rules made by Web authors. It provides a means for defining the structure, content, and semantics of XML documents. The DDL requires some specific extensions to XML Schema language to satisfy all the requirements of MPEG-7. These extensions are being discussed through liaison activities between MPEG and W3C.
4. *A binary representation to encode descriptions:* A coded description is one that has been encoded to fulfill relevant requirements such as compression efficiency, error resilience, and random access.

Here is an example of MPEG-7 description of an image content with a text annotation.

```
<mpeg7 xmlns:mpeg7="http://www.mpeg7.org/2000/
 MPEG7_schema"...>
   <ContentDescription xsi:type="ContentEntityType">
     <MultimediaContent xsi:type="ImageType">
        <Image id="img1">
          <MediaLocator>
            <MediaUri>
```

```
            http://www.nagao.nuie.nagoya-u.ac.jp/
            members/nagao/
            portrait.jpg
          </MediaUri>
        </MediaLocator>
        <TextAnnotation>
         <FreeTextAnnotation>Nagao's face
         </FreeTextAnnotation>
        </TextAnnotation>
        <VisualDescriptor xsi:type="DominantColorType">
          :
        <!--Dominant color description -->
          :
        </VisualDescriptor>
      </Image>
    </MultimediaContent>
  </ContentDescription>
</mpeg7>
```

3.1.4.1 MPEG-7 Features

MPEG-7 tools are able to detail deep levels of audio-visual content. Most content management tools describe audio-visual content at a shallow semantic level only, outlining features of audio-visual segments and subsegments such as news programs, sports, and politics, as well as titles, dates, actors, basic copyrights, and basic timing/frame information.

MPEG-7 tools, however, enable descriptions of deep-level features of audio-visual content: they break down a segment of audio-visual content into its constituent elements and into relationships between those elements. A moving region within a video segment can be indexed into a subset of moving regions by using MPEG-7. Then objects within a moving region can be further indexed. An object can also be indexed by its color, texture, and shape, and by its trajectory feature within the motion region. The spatial-temporal relationships of objects within frames of a stream can also be indexed using MPEG-7.

MPEG-7 aims for interoperability with other leading audio-visual standards groups and consortia. With a description language grounded in XML Schema, content creators, distributors, content management vendors, and start-up enterprises alike will find MPEG-7 an encouraging environment.

MPEG-7 will provide the most comprehensive tool set for managing audio-visual content. This promise may not be realized immediately; initial applications most likely will be designed using the high-level catalog and semantic description tools of MPEG-7. The structural tools for describing deeper levels of audio-visual content will at first be applied by professionals in the broadcast and digital studio industries. Search engine companies will follow once their customers start demanding better search and retrieval performances using audio-visual content.

A new breed of digital librarian, called a multimedia archivist, will appear, who, with indexing tools such as those provided by MPEG-7, will help store content for retrieval purposes. At first, this process will move slowly, because converting legacy content will create a bottleneck; however, the costs of this conversion process will eventually be covered by customers willing to pay for access to rich multimedia.

3.1.4.2 MPEG-7 Scope

Figure 3.1 shows a highly abstract block diagram of a possible MPEG-7 processing chain, included here to explain the scope of the MPEG-7 standard. This chain includes feature extraction (analysis), the description itself, and the search engine (application).

To fully exploit the possibilities of MPEG-7 descriptions, automatic extraction of features (or descriptors) will be extremely useful. It is also clear that automatic extraction is not always possible. However, the higher the level of abstraction, the more difficult automatic extraction is, and interactive extraction tools will be of great use. However useful they are, neither automatic nor semi-automatic feature extraction algorithms are inside the scope of the standard. The main reason is that their standardization is not required to allow interoperability, while leaving space for industry

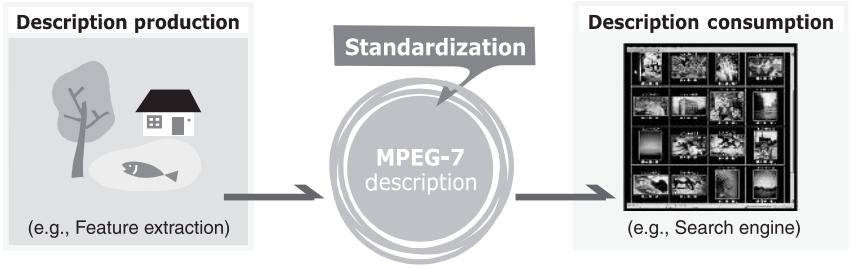


Figure 3.1 MPEG-7 scope.

competition. Another reason not to standardize analysis is to allow making good use of the expected improvements in these technical areas.

Also, the search engines, filter agents, or any other program that can make use of the description, are not specified within the scope of MPEG-7; again this is not necessary, and here too, competition will produce the best results.

3.1.4.3 Semantic Aspects in MPEG-7

For some applications, the viewpoint described above is not appropriate because it highlights the structural aspects of the content. For applications where the structure is of no real use, but where the user is mainly interested in the semantics of the content, an alternative approach is provided by the Semantic DS. In this approach, the emphasis is not on segments but on events, objects, concepts, places, time in narrative worlds, and abstraction.

Narrative world refers to the context for a semantic description—that is, it is the ''reality'' in which the description makes sense. This notion covers the world depicted in the specific instances of audio-visual content as well as more abstract descriptions representing the possible worlds described in the possible media occurrences. A description may involve multiple narrative worlds depicted in multiple instances of audio-visual content.

The SemanticBase DS describes narrative worlds and semantic entities in a narrative world. In addition, a number of specialized DSs are derived from the generic SemanticBase DS, which describe specific types of semantic entities, such as narrative worlds, objects, agent objects, events, places, and time, as follows:

1. The Semantic DS describes narrative worlds that are depicted by or related to the audio-visual content. It may also be used to describe a template for audio-visual content. In practice, the Semantic DS is intended to encapsulate the description of a narrative world.
2. The Object DS describes a perceivable or abstract object. A perceivable object is an entity that exists (i.e., has temporal and spatial extent) in a narrative world (e.g., ''Tom's piano''). An abstract object is the result of applying abstraction to a perceivable object (e.g., ''any piano''). Essentially, this generates an object template.

3. The AgentObject DS extends from the Object DS. It describes a person, an organization, a group of people, or personalized objects (e.g., ''a talking piano in an animated movie'').
4. The Event DS describes a perceivable or abstract event. A perceivable event is a dynamic relation involving one or more objects occurring in a region in time and space of a narrative world (e.g., ''Tom playing the piano'').

   An abstract event is the result of applying abstraction to a perceivable event (e.g., ''anyone playing the piano''). Here also, this generates a template of the event.
5. The Concept DS describes a semantic entity that cannot be described as a generalization or abstraction of a specific object, event, time, place, or state. It is expressed as a property or collection of properties (e.g., ''harmony'' or ''ripeness''). It may refer to the media directly or to another semantic entity being described.
6. The SemanticState DS describes one or more parametric attributes of a semantic entity at a given time or spatial location in the narrative world, or in a given location in the media (e.g., the piano's weight is 100 kg).
7. SemanticPlace and SemanticTime DSs describe, respectively, a place and a time in a narrative world.

3.1.4.4 Application Domains of MPEG-7

The increased volume of audio-visual content available in our everyday lives requires effective multimedia systems that make it possible to access, interact, and display complex and inhomogeneous information. Such needs are related to important social and economic issues and are imperative in various cases of professional and consumers applications, such as:

- Education;
- Journalism (e.g., searching speeches of a certain politician using his name, his voice, or his face);
- Tourist information;
- Cultural services (e.g., history museums, art galleries, and so on);
- Entertainment (e.g., searching a game, karaoke);
- Investigation services (e.g., human characteristics recognition, forensics);

- Geographical information systems;
- Remote sensing (e.g., cartography, ecology, and natural resources management);
- Surveillance (e.g., traffic control, surface transportation, and nondestructive testing in hostile environments);
- Biomedical applications;
- Shopping (e.g., searching for clothes that you like);
- Architecture, real estate, and interior design;
- Social (e.g., dating services);
- Film, video, and radio archives.

3.2 Creation of Annotations

While metadata is ''data about data,'' ''annotation'' means ''content about content.'' So annotation itself is a kind of content and is also capable of metadata functionality.

The annotations include text and voice commentaries or remarks, indices, markups, hyperlinks, and control messages for machines. The coverage of annotations is much broader than metadata. So a framework of annotations may be more complicated than that of metadata.

The main concern of this chapter is a creation of annotations as well as their applications. There are some tools that can associate remarks to existing documents and some application server systems that enable users to share these annotation data and extract a particular topic from documents using the annotations. Annotea is one of the most popular tools to create commentary annotations. It is based on RDF and an annotation repository architecture. There have been many previous approaches to creating and sharing external remarks on Web content. The major difference of Annotea and them is that Annotea uses a promising standard framework while others employ their proprietary data formats.

One of the major applications of annotations other than content retrieval is transcoding. As explained in the previous chapter, high-quality trancoding sometimes requires annotations—that is, extra information about content such as exact meanings and relations of elements in content. Accessibility transcoding that transcodes original content into a more readable or listenable one can achieve higher quality if the content has annotation. An IBM group has developed an annotation method

applicable to a whole Web site. Their annotation specifies a typical meaning of visual structure of content in the site. This is useful if content is updated day by day and it is impractical to create annotation individually.

3.2.1 Annotea

Annotea is a Web-based shared annotation system based on a general-purpose open RDF infrastructure [9]. This annotation is viewed as a statement made by an author about a Web document. The annotations are external to the documents and can be stored in one or more annotation servers. One of the goals of this project has been to reuse as much existing W3C technology as possible.

The developers of Annotea also implemented an instance of their system using the Amaya editor/browser and a generic RDF database, accessible through an Apache HTTP server. In this implementation, the merging of annotations with documents takes place within the client, not in an intermediary such as a proxy server.

3.2.1.1 Requirements

The following list gives the principal requirements that have shaped Annotea:

1. *Open technologies:* Many of the existing annotation systems are based on proprietary schemes or protocols. This makes it hard to extend them. Annotea is built on top of open standards to simplify the interoperability of Annotea with other annotation systems and to maximize the extensibility of the data this system can carry.
2. *Annotated documents are well-formed, structured documents:* Many Web annotation systems allow users to annotate any kind of resource that has a URI. To simplify the design, the developers decided to limit annotated resources to those that have a structure—that is, any HTML or XML-based document, including other annotations.
3. *Annotations are first-class Web resources:* As any other Web resource, each annotation should be associated with a URI.
4. *Annotations are typed:* At the same time that an annotation can be seen as metadata related to an annotated document, annotations themselves can have distinct properties. The type of an annotation is metadata about the annotation itself. It allows users to classify

the annotations as they are creating them (for example, saying this annotation is a comment or an erratum about a given document).

5. *Annotation types can be defined by users:* Different users have different views and needs. Annotea should make it possible for any user group to define their own annotation types.
6. *Annotation properties must be described with an RDF Schema:* Annotations are stored in generic RDF databases. Rather than making a specialized annotation server, the servers can be viewed as generic RDF databases. This is important as it will allow users to reuse the information stored in such databases without having to change them.
7. *Local (private) and remote (shared) annotations:* Annotations can be stored either locally in the user's client computer or in an annotation server. It is assumed that local annotations are private and remote ones are shared. An annotation server is responsible for controlling access to the annotations that it stores.
8. *Multiple annotation servers:* Having a centralized server may present both scalability and privacy problems. User groups must be able to easily set up an annotation server and define who can consult it. Thus, in an Annotea instantiation, there may be a number of annotation servers.

3.2.1.2 Annotea Operation

In Annotea, annotations are described with a dedicated RDF Schema and are stored in annotation servers. The annotation server stores the annotations in an RDF database. Users can query a server to either retrieve an existing annotation, post a new annotation, modify an annotation, or delete an annotation. All communication between a client and an annotation server uses the standard HTTP operations as shown in Figure 3.2.

The annotations handled by Annotea are collections of various statements about a document. They may be comments, typographical corrections, hypotheses, or ratings, but there is always an author that makes a statement about the document or some part of it at a certain time. An annotation is represented as a set of metadata and an annotation body.

The metadata of an annotation is modeled according to an RDF Schema and gives information such as the date of creation of the annotation, name of the author, the annotation type (e.g., comment, query, correction, and so forth) the URI of the annotated document, and an XPointer [10] that specifies

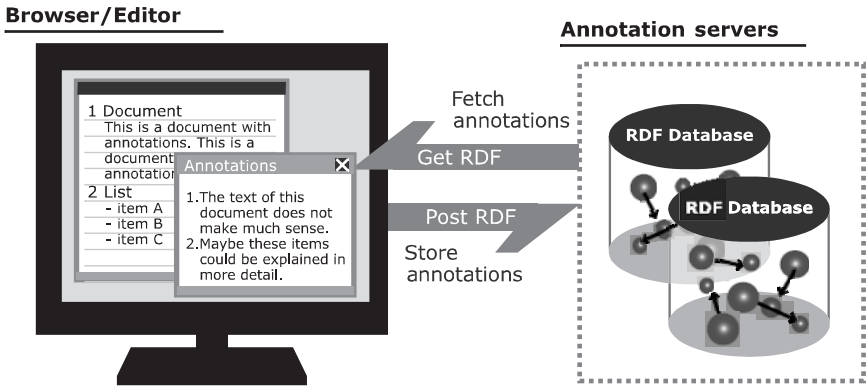


Figure 3.2 Annotea basic architecture. (*From*: [11]. © 2002 ACM Inc.)

what part of the document was annotated. The metadata also includes a URI to the body of the annotation, which we assume to be an XHTML document. The annotation metadata does not say how the annotations must be presented to the user. This choice is left open to the developer of the client.

Annotations are stored in generic RDF databases, which are accessible through an HTTP server. The following scenario explains the interaction between these components when a user creates a new annotation document. For simplicity, the annotations are supposed to be displayed by highlighting the annotated text in the document.

The following is a typical scenario that a user creates and submits an annotation:

1. The user browses a document.
2. The user selects some text on the document and tells the browser that he or she wants to annotate this text.
3. The browser pops up a new window, where the user can type the text of his annotation and choose the type of the annotation.
4. The user then publishes the annotation to a given annotation server. To do this, the browser generates an RDF description of the annotation that includes the metadata and the body and sends it to the server, using the HTTP POST method. The server assigns a URI to the annotation and to the body and replies with an RDF statement that includes these URIs.
5. If the user further modifies the annotation, he will publish it directly to the URI that was assigned.

Note that the first time that a user publishes an annotation, this annotation does not have any URI. It is the server that assigns the URI. When the user requests the URI from the server later, the server will reply with the annotation metadata.

The next scenario describes how the user browses an annotated document. Suppose that this user has previously configured his or her browser with the list of annotation servers that he or she wants to query.

1. The user browses a document.
2. The browser queries each of the annotation servers, requesting via an HTTP GET method the annotation metadata that is associated with the document's URI.
3. Each server replies with an RDF list of the annotation metadata. If the server is not storing any related annotations, it replies with an HTTP 404 Not Found message.
4. For each list of annotations that it receives, the browser parses the metadata of each annotation, resolves the XPointer of the annotation and, if successful, highlights the annotated text.
5. If the user clicks on the highlighted text, the browser will use an HTTP GET method to fetch the body of the annotation from the URI specified in the metadata.
6. Finally, the browser will open up a window showing the metadata and the body.

In the above scenario, there were two stages for downloading annotations. First, the browser downloads the metadata of an annotation. Next, and only if the user requests it explicitly, the browser downloads the body of the annotation. The motivation for this choice is to reduce the amount of data that is being sent back to the browser. In a heavily annotated document, sending the complete annotations will consume resources and the user may not actually be interested in seeing the body of all the annotations.

Note that once that an annotation is published to a server, it becomes a shared annotation. That is, any user with the correct access rights may retrieve the annotations from the server. For the moment, we expect that the HTTP server will enforce the access control to the annotations, using the standard HTTP authentication mechanisms.

It is also possible to store annotations locally in the client computer of the user, provided that the client simulates the reply to the first query of the server.

3.2.1.3 RDF Schema Application to Annotations

The most important feature of the annotation in the Annotea architecture is that it supports the evolving needs of the collaborating groups. For instance, an annotation system for classifying new technologies will need to extend their annotation types to classify specific characteristics of the technologies they are reviewing. Another working group may start with a set of annotation types and then modify this set according to the evolution of their work.

Annotea users may wish to define new types of annotations as they use the system more. The group may also add relationships to other objects that are specific to the group's work. RDF provides support for these needs (e.g., by allowing the expression of new relationships, by allowing new annotation types, and by supporting the transformations from one annotation type to another).

The type of an annotation is defined by the user or the group by declaring additional annotation classes. These classes are a part of the RDF model and may be described on the Web in an RDF Schema.

The general annotation super class is called Annotation (more precisely, its name is http://www.w3.org/2000/10/annotation-ns#Annotation, which is a URI about which an application can expect to ask the Web for more information) and the developers of Annotea have defined several sample subclasses based on it:

- *Annotation:* a super class describing the common features ofannotations;
- *Advice:* a subclass of Annotation representing advice to the reader;
- *Change:* a subclass of Annotation describing annotations that document or propose a change to the source document;
- *Comment:* a subclass of Annotation describing annotations that are comments;
- *Example:* a subclass of Annotation representing examples;
- *Explanation:* a subclass of Annotation representing explanations of content;
- *Question:* a subclass of Annotation representing questions about the content;
- *SeeAlso:* a subclass of Annotation representing a reference to another resource.

These subclasses are defined in a separate RDF Schema whose namespace is http://www.w3.org/2000/10/annotationTypes#. Likewise, other user

groups can easily create new subclasses. Users can also easily add new properties to the annotation classes; for instance, they could add a property that defines an annotation set. This property can then be queried with general RDF mechanisms and also presented as text.

The annotations in Annotea are user-made statements that consist of these main parts: the body of the annotation (which contains the textual or graphical content of the annotation) the link to the annotated document with a location within the document, an identification of the person making the annotation, and additional metadata related to the annotation. By using RDF, Annotea can take advantage of other work on Web metadata vocabularies wherever possible. Specifically, the Dublin Core element set can be used to describe some of the properties of annotations.

The RDF Schema that defines the annotation properties consists of the property name and the natural language explanation. The ''type'' is one of the above-mentioned basic classes or some other type of annotation defined elsewhere. The ''annotates'' property stores the link to the annotated document, ''body'' is a link to the content of the annotation, and ''dc:creator'' (imported from Dublin Core) to the author making the annotation.

The context defines where exactly inside the document the annotation is attached. Annotea uses XPointer for defining positions within XML documents. This works well for static (unchanging) documents, but with documents that go through revision, such as working group drafts, we may end up with orphan annotations or annotations pointing to wrong places. To prevent unnecessary loss of pointers, we can search for the nearest ID to a parent of the object and use it as the starting point for the XPointer path. Fortunately, many documents usually have IDs at least at their main levels. Pointing to finer details after the ID can be done by other means, such as using text matching.

The additional annotation metadata includes date for the creation and last modified time, and related resources for adding relationships to other objects. Other metadata can be added to the annotation when the working group needs that. For instance, the working group will probably add their own properties directly as shown in the following list:

- *rdf:type:* an indication of the creator's intention when making an annotation; the value should be of rdf:type Annotation or any of its subclasses;
- *annotates:* the relation between an annotation resource and the resource to which the annotation applies;
- *body:* the content of the annotation;

- *context:* context within the resource named in annotates to which the annotation most directly applies; eventually will be an XPointer and may include a location range, too;
- *dc:creator:* the creator of the annotation;
- *created:* the date and time on which the annotation was created;
- *dc:date:* the date and time on which the annotation was last modified;
- *related:* a relation between an annotation and a (collection of) resource(s) that augment the resource that is the body of the annotation. This may point to related issues and discussion threads.

3.2.1.4 Amaya

Amaya is a full-featured Web browser and editor developed by W3C for experimenting and validating Web specifications at an early stage of their development [12]. Amaya supports CSS, MathML, XHTML, HTML, and also provides a basic implementation of XLink [13] and XPointer. Amaya can also show different views of a document. In particular, Amaya has a formatted view, which shows the interpreted document, and a links view, which gives a list of all the links in the document.

Amaya's prototype implementation is able to interpret the RDF Schema for Annotation and supports all of the Annotea protocols. It is also possible to specify additional annotation types (subclasses) as an RDF Schema that can be downloaded at runtime. The namespaces for these additional types are specified to Amaya in a local configuration file that is read at startup. Amaya will use the namespace name to try to retrieve an RDF Schema from the Web or the schema content can be cached in a local file and specified with the same startup configuration.

The following are the most important features of the implementation of Amiya: creating, browsing, and filtering annotations.

3.2.1.5 Creating Annotations

The user has three choices for creating an annotation: annotate a whole document, annotate the position where the caret is, annotate the current selection. After making this choice, a pop-up annotation window appears. The annotation window shows the metadata of the annotation, inside a box and the body of the annotation.

Three items are active. If the user clicks on the Source document field, Amaya will scroll to the annotated text and highlight it if it is a selection. Clicking on the Annotation type field allows the user to change the type of annotation. Finally, each time that the user saves the annotation, Amaya updates the value of the Last modified field. The body of the annotation can be edited as any other XHTML document. Users can cut and paste fragments from other documents, add links to other documents, and so on.

Amaya supports both local (private) and remote (shared) annotations. When a user creates an annotation, it is considered a local one and will be stored in the user's Amaya directory. When the user decides to post it to an annotation server, the local annotation will be deleted and subsequent saves will be sent to the server. Both local and remote annotations are represented with the same schema format. The only difference is that for local ones, we emulate the annotation server's query response by using an index file that associates URIs with annotation metadata.

3.2.1.6 Browsing Annotations

By means of a setup menu, the user can specify the URIs of the annotation servers he wants to query, as well as the local annotation repository. The user can also say if he or she wants annotations to download automatically or only on-demand. In order to avoid hampering performance, Amaya separated the downloading process into two steps. Once a document is downloaded, the annotations metadata is downloaded asynchronously, just like images, and merged into the document. The body of an annotation is only downloaded when the user opens an annotation. The motivation for this choice is that metadata may be relatively smaller than the body of an annotation. Moreover, if the user does not open all the annotations, Amaya saves time by not downloading the body.

For showing annotations, Amaya defined an active XML element, which is called A-element, that has an XLink pointing to the body of the annotation and a special icon (currently, a pencil). When the user clicks once on the A-element, Amaya highlights the target of the annotation. Clicking on it twice will open the annotation window and show both the metadata and the body of the annotation. The A-element is visible in both the formatted document and links views and it is ignored when saving or printing an annotated document. Clicking on the A-element on any view has the same effect.

In the formatted view, Amaya positions the A-element to the location to which the XPointer of the annotation resolves. Amaya made an exception for MathML documents, as it would be disturbing to add it anywhere in

the expression. Instead, Amaya places it as the the beginning of the Math expression. Clicking on the A-element will highlight the target of the annotation, even if this target is not close to the A-element.

If an annotated document is edited, some of its annotations may become orphan. That is, the XPointer will not resolve anymore to a point in the document. In this case, Amaya will warn the user and make the orphan annotation visible from the links view. The user may then open the orphan annotation and reposition its XPointer or delete it.

3.2.1.7 Filtering Annotations

For a heavily annotated document, seeing the A-element icon can make reading the document bothersome. To avoid this problem, Amaya developers defined a local filter that allows the user to hide the annotations according to one of three criterion: by author name, by annotation type, and by annotation server. It is also possible to hide all the annotations in the formatted view. Using this menu, the user can hide all but the annotations that interest him. This filter menu does not have any effect on the links view.

As an alternative to hiding annotations, the user can also temporarily disable some annotation servers using the configuration menu. Amaya also has an experimental customized query feature, where the user can describe his own query, using a language named Algae [14]. This customized query interface makes it possible to start filtering the annotations on the server side, for example, by only requesting those done in the past week by a given author and belonging to a given annotation type.

3.2.1.8 Related Systems

Annotea concentrates on document-centered approaches where users are browsing documents and examining annotations related to them. There are also discussion-centered approaches to annotations, such as HyperNews [15], where users browse discussion messages and threads and follow a link to a document that these messages annotate.

Web annotations first appeared in version 1.2 of Mosaic, almost 10 years ago, and many other Web annotation–aware tools or servers have seen the light since then, such as CritLink and ThirdVoice. Due to the lack of existing annotation standards, most of these proposals are proprietary or closed.

The two main categories to Web annotation systems are proxy-based approaches and browser-based approaches. In a proxy-based approach,

annotations are stored and merged with a Web document by a proxy server. The browser user agent only sees the result of the merge, typically with some semantic content removed. In a browser-based approach, the browser is enhanced (either by an external application or by a plug-in) to merge the document and the annotation data just prior to presenting the content to the user. The annotation data is stored in the proxy or a separate annotation server. It is also possible to store annotations locally or provide site-specific annotations, but these are less interesting to us because of their limitations.

The CritLink annotation tool uses the proxy approach where a Web page and its annotations are served through a different URI address than the original document [16]. This approach works with any existing browser. However, the user must use different addresses for the document depending on which annotation proxy server is used. This is a limitation when a user wants to use more than one annotation server. The proxy approach also inherently restricts the types of content that can be annotated and the presentation styles that can be used for the annotations. Typically, presentation of the annotations is limited to the presentation styles available through HTML. Finally, as the browser does not have any knowledge about annotations, it makes it harder to filter the annotations locally, without having to send a new request to the proxy server.

ThirdVoice uses plug-ins to enhance Web browsers so that they understand annotations [17]. The users can annotate the page or some text on the page with discussions on selected topics. The discussions can be closed to a group of participants or open to anybody. Unfortunately, users cannot host their own servers.

iMarkup is an Internet Explorer annotation tool that has an interesting user interface [18]. Users have a wide variety of palettes for annotation markers and can even circle parts of the text with something akin to a real marker. Annotations can be placed anywhere on the browser's document window, without taking into account the markup of the document itself. All the annotations are local. A menu entry allows annotations to be mailed to other users and to be imported. The format used for describing annotations is proprietary and too related to the browser's API, making their use with other tools practically impossible.

Another possibility for presenting the annotations on a Web document is to use internal DOM events without actually changing the markup of the page. Yawas is an annotation tool that uses this approach [19]. It codes the annotations into an extended URI format and uses local files similar to bookmark files to store and retrieve the annotations. A modified browser can

transform the URI format into DOM events. The local annotation files can be sent to other users only by mail or copied by other means. There is no provision for having active links or filtering options. This kind of approach is limited by the API provided by the browser.

XLink, an XML linking technology currently under development in W3C, has some built-in features in the markup for creating annotations. For instance, it is possible to store XLink arcs (connecting elements) in an external document that can be loaded with another document. The content defined by the end locator of an XLink arc may be embedded to the location in a document defined by a starting locator of the arc. Using XLink provides the means to easily present the annotations in predefined ways in any browser implementing XLink. A more detailed discussion on XLink is presented in Section 3.4.3.

3.2.2 Web Site–Wide Annotation for Accessibility

As presented in the previous chapter, the Web has become a new information resource for handicapped people. Web accessibility, however, is becoming worse, since Web authors tend to care only for the visual appearance. Transcoding described in the previous chapter can solve this problem. The system has the ability to transcode documents on annotated sites into totally accessible ones without changing the original content. However, Web site–wide annotation authoring is an extremely tedious and time-consuming task. This prevents us from applying transcoding for accessibility to a wide variety of Web sites.

In order to overcome this difficulty, a new algorithm called *Dynamic Annotation Matching* was developed by Asakawa's group at IBM [11]. By utilizing this algorithm, the accessibility transcoding system can automatically determine appropriate annotations based on each document's layout. An authoring tool for Web site–wide annotation called *Site Pattern Analyzer* was also proposed by the researchers.

When a user accesses a page through this system, it converts the page into an accessible one by applying several transcoding methods, such as automatic insertion of missing ALT text, rearrangement of the document layout based on its logical structure, insertion of an index, delimiters between visual groupings, and so forth. This system allows visually challenged users to get more auditory information, and to explore each content logically by accessing an index and to easily recognize the role of each section by using textual delimiters. This system has been publicly available since March 2001, and already more than 100,000 users have tried this system.

The accessibility transcoding employs the *Accessibility Annotation* (A-Annotation). The A-Annotation is a kind of external metainformation, which describes the visual fragments in a rendered document. In other words, an A-Annotation data is an information on content layout over the Web site.

When a user requests a content, the transcoder redirects this request to the target Web server and gets the target resource. Then, the transcoder retrieves corresponding annotations from the annotation database. The database returns layout structure information, and the transcoder processes the target page based on this layout information. In this way, the transcoder can get the necessary information for optimum transcoding.

The workload of annotation authoring was greatly reduced by the proposed method. A-Annotation information has to be created manually. Human annotators need to create A-Annotations and register them in the annotation database. This authoring was extremely tedious and time-consuming, since annotators needed to understand the layout patterns of entire pages in a target Web site, and repeatedly create, check, and modify annotations adapted to dynamically changing Web content. Therefore, the Dynamic Annotation Matching Algorithm for annotation matching was proposed to reduce workload. The Site Pattern Analyzer was also developed to efficiently create reusable annotation data.

3.2.2.1 A-Annotation

As mentioned, representing visually fragmented groupings on a rendered document is crucial for improving nonvisual access. The rearrangement functions can represent groupings in three ways. First, they rearrange the groupings in a page based on roles and importance to allow users to access the content based on its importance. Second, they insert delimiter text at each boundary between groupings, such as ''Group 1 What's New.'' Third, they insert page indices at the bottom of the page to allow users to access each group directly.

Each numbered item is a link to a group, so users can access each group directly. After the content of a selected group is read, the next delimiter would be read as ''Group 2 Newsletters.'' This informs users of the end of one group and introduces the content of the next group. In this way, the system allows users to access Web documents logically. Therefore, the rearrangement function can provide drastically more accessible pages to visually challenged users.

For rearrangement, the system needs to understand not only the target content's layout structure, but also accurately understand each group's title

and role. A-Annotation is designed to support the transcoder by providing the necessary information as created by a human annotator. For this purpose an annotation authoring GUI tool called the Annotation Editor for A-Annotations was developed. Using this tool, an annotator can create annotations for one page within 15 minutes.

Figure 3.3 shows relationships between the visual appearance of groups and the XML encoding of A-Annotations. The editor software does this encoding process automatically. When an annotator selects a group in a browser view, the editor looks for a corresponding node in the DOM tree structure and describes the path from the root node to that node using XPath notation [20].

For the gray node in the DOM tree in the figure, the root node is ''body,'' the second level is the third ''table,'' this ''tr'' is the first child of the table, and the ''td'' is the second ''td'' child of ''tr.'' This path is represented as ''/body/table[3]/tr[1]/td[2]'' in XPath notation. After an area is specified, The editor asks an annotator to input a title and importance. The XPath, title and importance are combined and converted into XML format as an A-Annotation. The XML annotations are then registered in the annotation database, and the new annotation is then available for transcoding.

3.2.2.2 Annotation Matching

There are two major technical challenges for annotation technology, in general:

1. *Reducing the authoring workload:* One obvious answer is by adapting generalized annotations to similar Web documents.
2. *Adapting annotations to dynamically changing documents:* Web documents are always changing their content and layout and new documents are created ceaselessly.

Annotation matching is the key technology to solve these problems. An annotation database selects a corresponding annotation file by calculating the fit between a target page and an annotation file based on some matching algorithm.

3.2.2.3 Regular Expression Matching of URLs

Documents with similar layouts tend to have similar URLs in many cases. For example, for *The New York Times* Web site (www.nytimes.com), URLs involve date, category, and part of the title such as http://www.nytimes.com/2001/11/21/international/21CND-KUND.html.

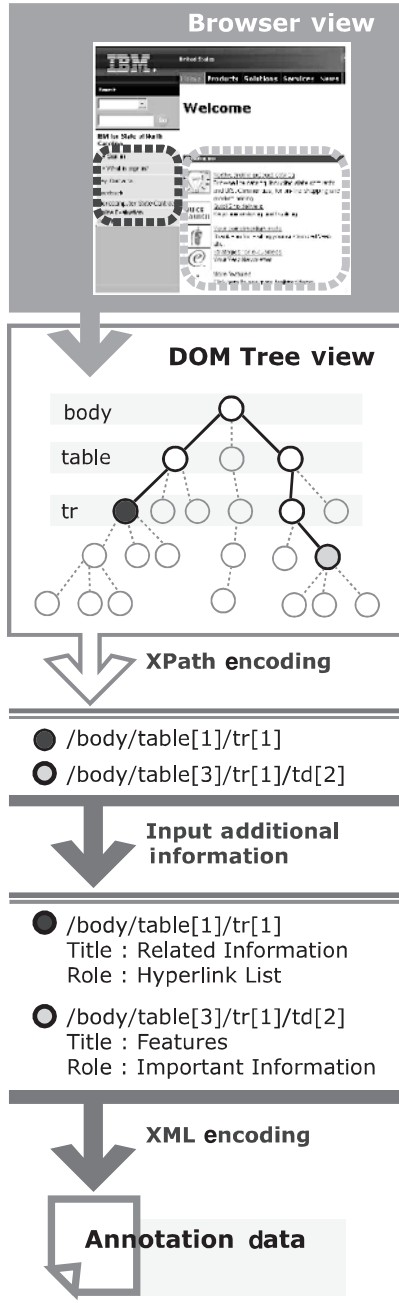


Figure 3.3 XPath notation in the Annotation data. (*From*: [11]. © 2002 ACM Inc. Reprinted with permission.)

Therefore, we can expect that URLs in this Web site may follow a specific rule such as http://www.nytimes.com/[year]/[month]/[date]/[category]/[acronym of title].html.

This rule can be converted into a regular expression format, such as http://www.nytimes.com/200[0-9]/[0-1][0-9]/[0-3][0-9]/(international | politics | business | national)/+.html. If all article documents in this Web site have exactly the same structure, one annotation using this regular expression will cover all article documents on this site. In this case, the transcoder only sends the target page's URL to retrieve a corresponding annotation data. The database looks for a regular expression matching the target URL, and the annotation file associated with the matched regular expression will be returned to the transcoder as a result of the query.

Researchers, however, found that this method can only be used for static HTML documents with simple layouts, since the returned annotations are still unable to handle dynamically changing Web documents. These days, content in major commercial Web sites is not static, but dynamically generated, and their layouts are continuously changing.

Many Web sites modify their layout and contents based on users' cookies, location, time, and browsing history in order to provide personalized and user-centric pages. For example, CNN's top page has a function to provide one selected edition among three editions based on a user's cookie. Most shopping sites change their pages based on a user's purchase history. These changes make the annotation matching difficult, and the regular expression-based matching method cannot deal with these problems. If an annotation database uses regular expressions, it is very hard to create annotations covering most of the pages in major sites, since there are huge amount of dynamic content and an annotator is required to rewrite and adjust each annotation data.

3.2.2.4 Dynamic Annotation Matching

The annotation database should take the layout of the target page into account, instead of focusing on the URLs. In other words, a layout-based annotation matching algorithm is necessary to realize an annotation over a Web site. However, creating additional description for layouts may be more tedious than writing XPaths or regular expressions.

In order to overcome this potential problem, a new matching algorithm called Dynamic Annotation Matching was developed. The fundamental idea of this algorithm is to regard the annotation description itself as a pattern for the layout.

Each annotation is associated with one layout pattern. Layout patterns are essentially lists of XPaths involved in each annotation file. Each XPath must start with the `<body>` element, and therefore each layout pattern takes the form of a subtree with one root node and several paths extending from the root to specific nodes. XPath patterns typically have 10 to 30 branches. Subtrees are assumed to consist of layout-related tags, such as `<table>`, `<div>`, `<center>`, `<font>` and so forth, and each branch points to page-specific nodes. Therefore, the groups of XPaths can be considered as layout patterns.

When an annotator submits an annotation data, the database automatically generates a layout pattern by listing the XPaths involved in each annotation file and compiling them. The transcoder sends a target's DOM tree structure to the database, and the database evaluates each layout pattern with the target's DOM. If some pattern matches, the database returns the associated annotation to the transcoder.

The point of this method is that it releases the annotators from the loop of checking and fixing XPaths and regular expressions, since the database can take over by automatically retrieving corresponding annotations. The workload for maintenance can decrease. If an unmatched document is found, an annotator can simply create an annotation data for the document and submit it, without concerning about XPaths and regular expressions. Therefore, the workload of the annotation authors may be drastically reduced by this algorithm. Another point is that annotators do not need to input too much information for layout matching. They can create annotations as usual, and the patterns can be automatically generated from the annotation files themselves.

3.2.2.5 Site Pattern Analyzer

Further problems to realize accessible Web site–wide annotation are as follows:

- *Duplicate matching:* Ideally, one page should have only one annotation, but sometimes multiple annotations match one target content. In this case, the annotation database needs to select the most appropriate one by some extra rules, such as assigning higher priority to the annotation with the longer Xpath; of course, such a rule is insufficient.
- *Missing annotation:* For documents without annotation, the annotators still need to work on them one-by-one.

The Site Pattern Analyzer was developed to solve these problems. This system can support annotation authoring by visualizing the status of annotation matching, so an annotator can easily recognize duplicate matching or missing annotation. Figure 3.4 shows an example screen of the Site Pattern Analyzer.

The annotator will find and fix these problems in four steps: analysis, summarization, visualization, and correction.

1. *Analysis:* At start-up, the system loads thousands of cached documents and annotation data and analyzes the matches for all of the annotations and documents in a particular Web site. This step is done automatically by the system, but may take a lot of time. Therefore, the annotator usually restricts the analysis only to targeted documents.
2. *Summarization:* After analysis, the system creates a summary report of matching status as an HTML document. The annotator can browse an overview of the current authoring status by checking this report, reviewing how many documents matched with each annotation data, how many documents do not have any matched annotation, and how many documents duplicately matched with annotation data.
3. *Visualization:* After providing the summary report, the system provides a graphical interface for checking and fixing the annotation database. The system has four panels in its screen, providing a Web site structure tree view, a matching status view, a matching detailed report view, and a browser view. Figure 3.4 shows an example of the matching status view, which indicates the matching status for documents. The annotator can see candidate annotation data in the matching status panel. A dark-colored bullet icon means unmatched XPaths, and light-colored shows a match. The annotator can see that the database will select the annotation data with green bullets for the target document, which is indicated by the ''sun'' icon.
4. *Correction:* When the annotator finds an unmatched document, he or she can create annotation data using the Annotation Editor and submit it to the database. When the annotation is submitted, the system automatically reloads the newly created annotation data and analyzes the matches. This correction is very simple and easy. When there are duplicate matches, there are several options. If all of the matched annotations are inappropriate for the document,

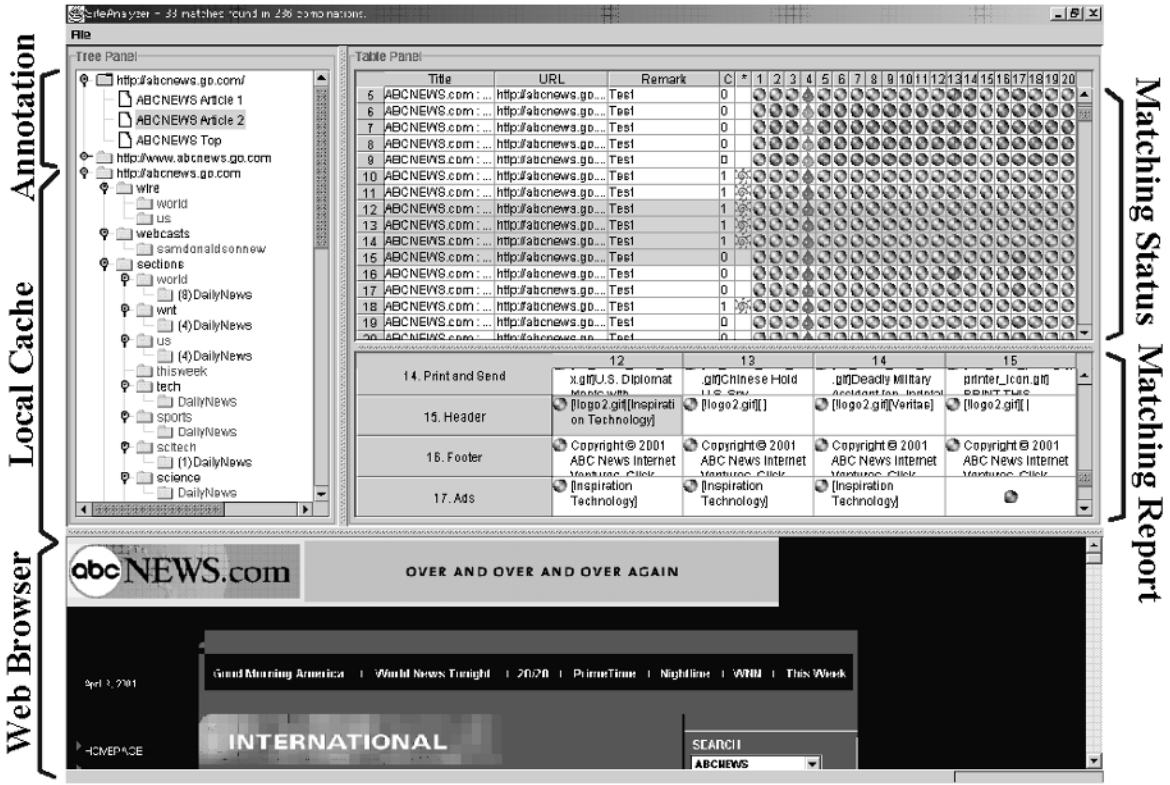


Figure 3.4 Screen of the site pattern analyzer. (Courtesy of Chieko Asakawa.)

new annotation data should be created. If an XPath points at incorrect content, a semi-automatic correction is useful. As mentioned above, an XPath in an annotation indicates a branch to a specific tag from a root node. This means that content in the tag being pointed at cannot be taken into account. The semi-automatic correction adds an additional conditional description to an XPath based on the content.

For example, this XPath might point at a headline for political news such as /body/table[1]. In another document, the same XPath might point at a headline for world news. In this case, this mismatch can be resolved by adding the following content-related condition:
/body/table[1][child::tr[1]/td[1]/text()[1] = 'Politics'].

The additional part means "the first text node in the first table cell should be 'Politics'." This XPath will reject a table involving "World" instead of "Politics."

3.2.2.6 Effectiveness and Further Study

The researchers evaluated the feasibility of Web site–wide annotation by creating annotations for an existing large Web site. In order to discuss applicability of this method to other Web sites, they defined a calculation method to estimate the work time.

They used the Common Layout Analysis method for this purpose, a kind of document clustering method. They developed this method to help with manual annotations by automatically detecting documents having the same layout. The output is a list of layout groups based on the layout-related nodes in the DOM tree structures. The number of layout groups is a reasonable index of the site's complexity. The estimated total work time T can be calculated with following equation:

$$T\ [min] = K\ [min/group] \times G$$

where K is the workload, minutes per group and G is the number of groups.

They analyzed the results of the current experiment using this equation. The target *USA Today* Web site was classified into 2,732 groups based on the algorithm. By measuring the total workload, the value of the coefficient K became 0.67 [minutes/group]. Now, the work time for general Web sites such as popular articles on CNN.com (3,965 pages) may take 24 hours, and *The Washington Post* Web site (10,122 pages) may take 36 hours.

The researchers considered that these work times should be feasible. While the work time estimation for creating and maintaining annotations has not been discussed in any annotation-related research, it is worth considering for real applications of annotation.

Their future plans to develop additional algorithms and tools are as follows:

- *Automatic detection of group titles:* If the annotation database can reliably detect the title of a group, the annotator will not have to create a redundant annotation data to adapt the title to the content. They expect that 10% of the annotation data could be eliminated and the work time could be reduced by 20% to 30%.
- *Triage of content to be annotated:* If the system can effectively classify documents into high-priority pages that nonvisual users want to read and documents that will be rarely accessed, then the work time could be reduced by 10% to 20%.
- *Parallel annotation authoring:* In the experiment, waiting time took 21% of the authoring time. The Site Pattern Analyzer could be used simultaneously with other operations, but analysis loads a machine heavily. Therefore, the annotator would benefit from having multiple machines to more effectively use the analysis time. With all of these improvements, they can eliminate 80% to 90% of work time required. If the system can reduce the effort to 10% of the current level, the work time for a half million pages would only be 168 hours. This means that a single annotator can create all of the annotations in 21 days.

3.3 Applications of Annotation

There are many applications of metadata or annotations beyond content retrieval and transcoding. Details of more advanced transcoding based on semantic annotation are described in the next chapter.

Here, two major applications of annotations are introduced: multimedia content distribution and content access control.

3.3.1 Multimedia Content Transcoding and Distribution

Today, many elements exist to build an infrastructure for the delivery and consumption of multimedia content. There is, however, no big picture to

describe how these elements, either in existence or under development, relate to each other.

Work on the new standard MPEG-21 Multimedia Framework started in June 2000 [21]. MPEG-21 aims at describing how these various elements fit together. Where gaps exist, MPEG-21 will recommend which new standards are required. ISO/IEC JTC 1/SC 29/WG 11 (MPEG) will then develop new standards as appropriate while other relevant standards may be developed by other bodies. These specifications will be integrated into the multimedia framework through collaboration between MPEG and these bodies.

The result is an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points. This open framework provides content creators and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumer, providing them access to a large variety of content in an interoperable manner.

The vision for MPEG-21 is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities. MPEG-21 introduces the concept of *digital item,* which is an abstraction for a multimedia content item, including the various types of relating data. For example, a music album may consist of a collection of songs, provided in several encoding formats to suit the capabilities (bit-rate, CPU, and so on) of the device on which they will be played. Furthermore, the album may provide the lyrics, some bibliographic information about the musicians and compositors, a picture of the album, information about the rights associated to the songs and pointers to a Web site where other related material can be purchased. All this aggregation of content is considered by MPEG-21 as a digital item, where descriptors (e.g., the lyrics) are associated to the resources (i.e., the songs themselves).

Conceptually, a digital item is defined as a structured digital object with a standard representation, identification, and description. This entity is also the fundamental unit of distribution and transaction within this framework.

Seven key elements within the MPEG-21 framework have already been defined. One of these elements is Terminals and Networks. The goal of the MPEG-21 Terminals and Networks work item is to achieve interoperable transparent access to advanced multimedia content by shielding users from network and terminal installation, management, and implementation issues. This will enable the provision of network and terminal resources on demand

to form user communities where multimedia content can be created and shared, always with the agreed/contracted quality, reliability, and flexibility, allowing the multimedia applications to connect diverse sets of users, such that the quality of the user experience will be guaranteed.

3.3.1.1 Digital Item Adaptation

MPEG-21 promotes the universal multimedia access that is concerned with access to any multimedia content from any type of terminal or network, and thus, it is closely related to the target mentioned above of achieving interoperable transparent access to (distributed) advanced multimedia content. Toward this goal, and in the context of MPEG-21, the adaptation of digital items is targeted. A digital item is subject to a resource adaptation engine, as well as a descriptor adaptation engine, which together produce the modified digital item.

The adaptation engines themselves are nonnormative tools of digital item adaptation. However, tools that provide support for digital item adaptation in terms of resource adaptation, descriptor adaptation, and/or *quality of service* (QoS) management are within the scope of the requirements.

With this goal in mind, it is essential to have available not only the description of the content but also a description of its format and of the usage environment in order that content adaptation may be performed to provide the user the best content experience for the content requested with the conditions available. The dimensions of the usage environment are defined in more detail below, but it essentially includes the description of terminal and networks resources, as well as user preferences and characteristics of the natural environment. While the content description problem has been addressed by MPEG-7, the description of content format and usage environments has not been addressed and it is now the target of MPEG-21 digital item adaptation.

3.3.1.2 Requirements for Multimedia Adaptation

High-level requirements for adaptation, and specific requirements for usage environment description and content format description have been specified. The usage environment description shall consider all the dimensions that may influence the transparent access to the multimedia content, notably terminals, networks, users, and the natural environment where users and terminals are located. Additionally, the MPEG-21 digital item adaptation tools may include MPEG-7 or any external standard tools by reference whenever these tools fulfill requirements that have been specified in this

document. The content format description shall use generic tools to assist the adaptation engines.

General requirements on descriptions have been developed. These include generic properties of the description itself, such as compactness, extensibility, human readability, and low complexity. Additionally, the ability to efficiently access partial descriptions and update existing description has been targeted. With regards to digital rights management, there is also a requirement to support associations of rights data and expressions with descriptions for digital item adaptation.

The specific requirements on usage environment description aim to describe a variety of dimensions, including terminal, network, delivery, user, and natural environment capabilities. Terminal capabilities include hardware properties, such as processor speed and memory capacity, software properties such as operating system, display properties such as screen resolution, and device profiles, which may indicate the media formats supported (e.g., MPEG profile/level). Physical network conditions specify delay characteistics, such as end-to-end delay, one-way delay, or delay variation, error characteristics, such as bit error rate, packet loss, or burstiness, and bandwidth characteristics, such as amount of available bandwidth or bandwidth variation. Delivery capabilities specify the type of transport protocols supported, such as MPEG-2 Systems, TCP/IP and *Real-Time Transport Protocol* (RTP), as well as the types of connections supported (e.g., broadcast, unicast, multicast). User preferences include filtering and search preferences, browsing preferences, display preferences, and QoS preferences, as well as demographic information, such as gender and age. Natural environment characteristics include location, such as *Global Positioning System* (GPS) coordinates and locale, the type of location (e.g., indoor, outdoor, home, or office), the velocity of a user or terminal, as well as the illumination properties affecting a user or terminal.

In addition to the above, the digital item adaptation descriptions may also specify service capabilities. Service capabilities include a particular user's role (e.g., content creator, service provider, rights owner, billing party, or end consumer), as well as the type of service that a particular user provides, such as content creation, rights negotiation, billing, content adaptation and transcoding, use of the network, and content consumption. Assuming that a particular user is the rights owner or content creator, digital item adaptation descriptions may also include the permissible types of adaptations that are allowed (e.g., the bit rate should not be less that 2 Mbps or spatial resolution of a video should not be reduced by more than a factor of two).

One important requirement regarding the usage environment description is that MPEG-21 shall provide mechanisms that support vocabularies defined by established capability description frameworks and be semantically compatible with such vocabularies. This requires MPEG-21 to be semantically compatible with CC/PP defined by W3C, UAProf defined by WAP Forum [22], Content Negotiation (CONNEG) defined by IETF [23], and any emerging works.

Beyond the usage environment description, the content format description shall provide information on how the resource itself can be adapted in a generic way. For example, several multimedia coding formats such as JPEG2000 for still images or MPEG-4 for videos feature scalable properties. By retrieving and/or decoding a part of the bit stream, it is then possible to render a degraded version of the content in terms of SNR quality, frame-rate or size without having to decode and reencode it. The content format description shall enable generic tools to work in the compressed domain, and hence facilitate the adaptation of scalable multimedia content to terminal and network capabilities.

3.3.2 Content Access Control

XML is a promising standard for describing semi-structured data and documents on the Internet. When XML comes to be a widespread data encoding format for Web applications, safeguarding the accuracy of the information represented in XML documents will be indispensable.

Kudo and Hada of IBM Research proposed *XML access control language* (XACL) to annotate documents with security features such as authorization, nonrepudiation, confidentiality, and an audit trail for documents [24]. Their implementation can be used as an extension of a Web server for e-business applications.

3.3.2.1 Security of Content

The XACL approach extends XML documents in terms of security. The basic idea for securing an XML document is the notion of provisional authorization, which adds an extended semantics to a traditional authorization model, and the notion of security transcoding, which protects data from possible threats by transforming the original content into another encoding format.

W3C is working on XML Digital Signature and element-wise encryption, and their implementations are already available [25].

Early authorization models have assumed that the system either authorizes the access request or denies it. The proposed model enables the authorization system to make more flexible authorization decisions by incorporating the notion of provisional actions into traditional authorization semantics. Damiani et al. proposed an access control model for XML documents and schema definition [26]. While they lay stress more on the semantics for reading XML documents, and they only present a behavioral model, XACL deals not only with the read function but also with write, create, and delete capabilities and also defines an access control specification language as an application of XML.

KeyNote is a unified approach to specifying and interpreting security policies, credentials, and relationships [27]. The access control rules in KeyNote are termed assertion that consist of an issuer, subjects to whom trust is delegated, and a predicate, which specifies the class of actions delegated, together with conditions under which the delegations apply. The KeyNote system returns a text string, called a compliance value. The model semantics and assumptions, however, are different from XACL. The XACL approach has more comprehensive semantics since it applied the syntax of the traditional access control language that consists of an object, a subject, and an action, while in KeyNote there are no distinct primitives except for subject. On the contrary, the KeyNote system has no relation with the semantics of the compliance value and only sends it to each application, which in turn handles it independently.

The provisional actions proposed in XACL can be viewed as triggering actions in the sense that they are triggered by access requests. Although they have already been investigated on the context of intrusion detection systems and some database products, it has not been explored in the domain of access control.

3.3.2.2 Authorization Architecture

Almost all studies in access control and authorization systems have assumed the following model: A user makes an access request of a system in some context, and the system either authorizes the access request or denies it. The notion of a provisional authorization tells the user that his or her request will be authorized provided the person (and/or the system) takes certain security actions such as signing a statement prior to authorization of the request.

The following are examples of security policy:

- You are allowed to access confidential information, but the access must be logged.

- You are allowed to read sensitive information, but you must sign a terms and conditions statement first.
- If unauthorized access is detected, a warning message must be sent to an administrator.

The following elements are component primitives that consist of the provisional authorization model:

- *SUBJECT (sbj):* Initiator information such as user ID that is an output from an authentication processor. Any information regarding the subject such as group name and role name can be used in this model.
- *OBJECT (obj):* Resource pointer of target object such as a directory path expression. Since XML documents are handled, XPath notation is used to specify the target objects.
- *ACTION (act):* Action is an access mode permitted on the target object. The following generic actions are permitted on XML documents: read, write, create, and delete. The read is defined as the subject can read any element-related data such as tag name, attribute type/value pairs, and the contents of the elements. The write is defined as allowing the subject to update the content of the elements or the values of the attributes. The create is defined as allowing the subject to create a new tree structure under the target element. The delete action is defined as permitting the subject to delete the target element and any subtree below it, or the attribute type/value pairs.
- *CONTEXT (cxt):* Context is any data such as the time at which the request occurred, the location from which the request was initiated, and any arguments for the specific action.
- *PROVISIONAL ACTION (prv):* Provisional action is a set of functions used for adding extended semantics on the authorization policy. This primitive has not been introduced in past authorization models. An initial set of provisional actions are: log, verify, encrypt, transform, write, create, and delete.
- *REQUEST (req):* obj × sbj × act × cxt. req refers to a query as to whether or not the subject of sbj is permitted to perform the action of act on the object of obj under the context of cxt.

- *PERMISSION (pms):* {*grant*, *deny*}. pms is a flag indicating whether access can be granted or denied.
- *FORMULA (fml):* A formula expression that returns a binary value, true or false. This expression is evaluated in the evaluate function that selects the authorization rule that fits the authorization request. The formula is defined as logical conjunction of equalities and/or inequalities.

An evaluation function using above components is defined as follows:
Evaluate: req $\rightarrow$ pms $\times$ prv

This function takes the authorization request as input, evaluates authorization rules with the formula expression, and returns the access permission of an appropriate authorization rule, along with any required provisional actions.

The researchers defined the provisional authorization model based on the above model components.

Authorization policy is defined as a 7-tuple

```
<obj, sbj, act, pms, prv, cxt, fml>.
```

This means that if the access flag pms is "grant" and the fml holds, then the initiator sbj can perform the act on the obj under the cxt, provided that all of the prvs are executed. If the pms is "deny" and the fml holds, then the sbj cannot perform the act on the obj under the cxt; however, all of the prvs must still be executed.

Authorization decision is defined as a 6-tuple

```
<obj, sbj, act, pms, prv, cxt>.
```

This means if the pms is "grant" and the prv is not empty, the sbj is permitted to perform the act on the obj under the cxt, provided all of the prvs are executed. If the pms is "deny" and the prv is not empty, the sbj is denied to perform the act on the obj with the cxt but all of the prvs must still be executed.

Consider that we have two authorization policies:

```
<account, Manager, read, grant, _ , _ , _> and

<contractor, Registered_ Client, read, deny, log, 'today is
    holiday', _>
```

The first rule says that the manager can read the element `account` and the second says that if today is holiday, the registered client cannot read the contractor element but the access must be logged. When the authorization request of `<account, Manager, read, _>` is submitted, the authorization system returns `<account, Manager, read, grant, _, _>`, which says that the manager is permitted to read element `account` if the evaluate function selects the first rule as an appropriate authorization policy. When the authorization request of `<contractor, Registered Client, read, _>` is submitted and if the time at which the request occurred is a holiday, the system returns `<contractor, Registered Client, read, log, _, _>`, whcih says that the registered client is not allowed to read the contractor element but the access must be logged for auditing purposes if the evaluate function selects the second rule as an appropriate policy. The latter example shows that the provisional authorization model works differently from the traditional authorization model.

3.3.2.3 XACL

XACL is an access control language based on the provisional authorization model. The syntax and semantics of XACL are now briefly explained. The primary purpose is to enable security policy programmers to write flexible authorization policies in XML. Another advantage of using the XML format for policy specification is that the XACL language can easily implement the policy management authorization rules. In other words, authorization rules or the authorization policy itself can be defined by metarules also described in XACL. There are two fundamental approaches to binding a set of policies written in XACL with target documents. One is at the schema definition level such as DTDs, and the other is at the level of each specific document. In the former approach, a set of policies is bound to all documents valid according to a DTD. Therefore, one needs to maintain the mapping between a particular DTD and the associated policy. In the latter, a policy is bound to each specific document. In this case, an associated policy, which is encoded as a `<policy>` element, may be an element contained within the target document. XACL can be applied to both approaches.

A policy consists of multiple XACLs, an `<xacl>` element consists of multiple `<rule>` elements, and a `<rule>` consists of multiple `<acl>` elements. The target object is specified for each `<xacl>` element. This means you can specify multiple rules and acls for the same target element. In an `<acl>` element, zero or more subjects, one or more actions, and an optional condition are specified. An `<acl>` says that all specified subjects are

to be authorized to execute all specified actions (although they may be associated with provisional actions) on all specified subelements under <object> elements if the specified condition is satisfied.

A simple authorization policy says the authenticated user Nagao, who is assigned an employee role, has write privilege on contractor elements below the document, provided the access is logged.

Here is an example of the authorization policy specified in XACL.

```
<policy>
  <xacl>
   <object href="/document/contractor"/>
     <rule>
       <acl>
         <subject><uid>Nagao</uid>
         <roles><role>Employee</role></roles></subject>
         <action name="write"permission="grant">
          <provisional_action timing="before"name="log"/>
         </action>
       </acl>
     </rule>
  </xacl>
</policy>
```

A subject is specified by a triple: uid, role-set, and group-set, where role-set and group-set are a set of role names and group names, respectively. Semantically, it represents a user who has the specified identity, performs all specified roles and belongs to all specified groups. Both roles and groups have a hierarchical structure. When a user belongs to a group, he or she is a member of all its super groups. The same is true of roles. Therefore, all authorizations of a group and a role propagate to all its subgroups and subroles, respectively.

The DTD of subject is as follows:

```
<!ELEMENT subject (uid?,role*,group*)>
<!ELEMENT uid (#PCDATA)>
<!ELEMENT role (#PCDATA)>
<!ELEMENT group (#PCDATA)>
```

An object represents an element or a set of elements in a target XML document. A single XPath expression is employed to identify it. The ''href'' attribute is used to specify it. Objects have an element-based hierarchical

structure. This means that in a tree structure of XML documents, all nodes except for the element nodes are ignored with respect to hierarchy. In other words, authorizations specified for an element are intended to be applicable to all its nodes (attributes, texts, and so forth) but not to any subelements. However, policies can propagate on this hierarchy.

The DTD of object is as follows:

```
<!ELEMENT object EMPTY>
<!ATTLIST object href CDATA #REQUIRED>
```

An action is specified as an <action> element. XACL supports both grants and denials. The "permission" attribute is used to indicate whether it is a grant or a denial. The "name" attribute is used to specify the name of the action: read, write, create, or delete.

The DTD of action is as follows:

```
<!ELEMENT action (parameter*, provision*)>
<!ATTLIST action name (read|write|create|delete) #REQUIRED
          permission (grant|deny) #REQUIRED>
```

Provisional actions can be associated with an action. The "name" attribute is used to specify the name of a provisional action, and the "timing" attribute is used to specify the timing constraints on when the provisional action is executed (i.e., before or after the specified action is executed). A provisional action may succeed or fail. Provisional actions may take zero or more input parameters as predicates and functions. Provisional actions are application-specific.

The DTD of provisional action is as follows:

```
<!ELEMENT provisional_action (parameter*)>
<!ATTLIST provisional_action name CDATA #REQUIRED
          timing (before|after) "after">
```

3.3.2.4 Security Transcoding

The security system protects the target content by transforming the original content into a secure representation using XACL descriptions about the authorization policies. For example, if an authorization decision generated by the security system says that the initiator is permitted to read a contract document except for the identity of the contractor, the transcoding system

transforms an original contract document into that without contractor's identity field. Another example is that the system transforms a content of the element from plaintext format to ciphertext by applying some cryptographic operation and stores it in the authorization information repository in order to satisfy confidentiality.

The researchers call these transformational operations security transcoding. Basically, the security transcoding implies two types of transformations: transformation of the secure document to retrieve from the repository, and transformation of the secure document to modify in the repository.

Consider that there is an electronic contract document represented in XML and there are two roles: the business owner who offers business to clients, and the registered client who makes the contract with the business owner.

The target XML contract document and the subject relation are stored in the authorization information repository. In the following examples, an element of the document is considered as a target object.

An XML document has a rooted tree structure. An element specified by a tag is a node. Each element can have attributes that consist of names and the associated values, and can have some content. The text string at the leaf indicates the content of the element. An example of the document is as follows:

```
<?xml version="1.0"encoding="UTF-8"?>
<document>
   <contractor level="1">
     <contract class="A">
       <account>Purchase of $1M</account>
       <representative>Katashi Nagao</representative>
     </contract>
     <comments/>
   </contractor>
   <status>
     <log>
      account, Business_Owner, write,
       "Purchase of $1M", April 10, 2002.
     </log>
   </status>
</document>
```

The "contract" is the element name, "class" is the attribute name, and "A" is the attribute value both attached to the "contract" element, and "Purchase of $1M" indicates the text content of the "account" element. XPath represents a path expression in a document tree by a sequence of element names or predefined functions separated by the "/" character. For example, "/document/contractor/contract/account" denotes the account element that is a child of the contract element that is a child of the contractor element.

3.3.2.5 Read Transcoding

Consider the following set of authorization decisions coming from the security system:

1. `<document, Registered_Client, read, deny, _, _>;`
2. `<contractor, Registered_Client, read, deny, _, _>;`
3. `<status, Registered_Client, read, deny, _, _>;`
4. `<log, Registered_Client, read, deny, _, _>;`
5. `<comments, Registered_Client, read, deny, _, _>;`
6. `<contract, Registered_Client, read, grant, _, _>;`
7. `<account, Registered_Client, read, grant, _, _>;`
8. `<representative, Registered_Client, read, grant, _, _>.`

The decisions are generated in response to the authorization request such as `<document, Registered_Client, read, _>` with the authorization policy of `<contract, Registered_Client, read, grant, _, _, _>` and a default-policy of denial.

The first authorization decision says that Registered Client is not allowed to read the document element, while the sixth decision says that Registered Client is permitted to read the contract element. The read transcoding module receives the above authorization decisions and generates a corresponding set of elements according to the type of permissions. If the permission is grant, it generates an element with any related information such as attributes and the content text string. If the permission is deny, it generates only an element name. As a result, the target document is transcoded into the document that does not include the status and comments elements and the attribute "level=1" as shown in the following:

```
<?xml version="1.0"encoding="UTF-8"?>
      <document>
        <contractor>
           <contract class="A">
```

```
      <account>Purchase of $1M</account>
      <representative>Katashi Nagao</representative>
    </contract>
  </contractor>
</document>
```

3.3.2.6 Update and Log Transcoding

Consider the following authorization decision:

```
<account, Business_Owner, write, grant, log, Purchase
    of $2M">
```

This says that Business Owner is allowed to update the account element with the value of "Purchase of $2M," provided the access is logged. Assume that this is generated in response to the authorization request of

```
<account, Business_Owner, write, "Purchase of $2M">
```

with the authorization policy of

```
<account, Business_Owner, write, grant, log, _,
    "comments field is empty">.
```

First, the log transcoding module adds a new log entry under the status element of the target XML document. If it succeeds, the write transcoding module updates the account element. The sequence of calling transcoding modules conforms to the semantics of the provisional authorization. The result of transcoding should be the following:

```
<?xml version="1.0"encoding="UTF-8"?>
  <document>
    <contractor level="1">
      <contract class="A">
        <account>Purchase of $2M</account>
        <representative>Katashi Nagao</representative>
      </contract>
      <comments/>
    </contractor>
```

```
    <status>
         <log>
          account, Business_Owner, write,
          "Purchase of $1M", April 10, 2002.
         </log>
          <log>
            account, Business_Owner, write,
             "Purchase of $2M", April 20, 2002.
          </log>
    </status>
</document>
```

3.3.2.7 Encryption Transcoding

Consider the following authorization decision:

```
<account, guest, read, grant, encrypt, key1234>
```

This says that guest user is allowed to read the account element, provided the value is encrypted with the cryptographic key of key1234. This means that the guest who knows the value of the key1234 can understand the account element. Assume that this is generated in response to the authorization request of `<account, guest, read, _>` with the authorization policy of `<account, guest, read, grant, encrypt, key1234>`.

First, the encryption transcoding module is called to encrypt the account. Next, the read transcoding module generates a corresponding element and its content as an authorization result.

3.3.2.8 Signature Verification Transcoding

It is reasonable that the contractor is required to put his or her signature on the statements if he or she makes the contract. Assume that the statement and the signature value are sent from the client as the context parameters.

Consider the following authorization decision:

```
<comments, Registered_Client, write, grant, log&verify,
     "Contractor accepts the contract">
```

This says that Registered Client is allowed to write ''Contractor accepts the contract'' in the comments element, provided the value is logged then and successfully verified using the signer's public key.

First, the log transcoding module is called, and next, the signature verification module is called to verify the client's signature. If both succeed, the write transcoding module updates the comments element with the verified statement of "Contractor accepts the contract." The result of transcoding should be the following:

```
<?xml version="1.0"encoding="UTF-8"?>
 <document>
   <contractor level="1">
    <contract class="A">
      <account>Purchase of $2M</account>
           <representative>Katashi Nagao</representative>
    </contract>
    <comments>Contractor accepts the contract.</comments>
</contractor>
<status>
   <log>
     account, Business_Owner, write,
     "Purchase of $1M", April 10, 2002.
   </log>
   <log>
      account, Business_Owner, write,
     "Purchase of $2M", April 20, 2002.
   </log>
   <log>
      comments, Registered_Client, write,
      *ENCRYPTED TEXT*, April 21, 2002.
   </log>
</status>
</document>
```

3.4 More on Annotation

3.4.1 Annotation by Digital Watermarking

Digital watermarking is a technique by which imperceptible digital code is embedded in content such as audio, video, and images, or printed material such as magazine pages, financial documents, or identity cards.

Digimarc provides content owners with a range of solutions for copyright communication, asset management, as well as enhanced

e-commerce capabilities to manage, track, and protect digital image, audio, and video assets [28].

Digimarc and Philips developed and provide a high-quality video and audio watermarking systems called WaterCast that can be integrated and used in a wide variety of applications. The WaterCast system includes video watermarking embedders and detectors that add identifying data to the video signal prior to transmission by terrestrial, cable, and satellite broadcasters. These identifiers are imperceptible to television audiences and survive all common video-processing operations. Broadcasts can then be monitored using WaterCast detectors to verify that program and advertisement transmissions comply with contractual requirements and do not occur without the permission of the broadcast owner.

Digital watermarking technology enables users to embed invisible and inaudible messages, called watermarks, into still images, video data, and audio data. The watermark does not increase the size of the data file. It is not affected by format conversions, signal processing, lossy compression, or analog transmission. The detected messages are use to prevent unauthorized copy and alteration by specifying ownership, monitoring unauthorized use, filtering illegal copies, and detecting modified locations.

The digital watermark can be considered as a device of unbreakable connection of multimedia content with the annotations. Whenever the content is edited, modified, or converted into a different format, the watermark continues to exist and remains unchanged. So if the watermark data includes a location or an identifier of the annotation data, the annotation created for the original content will never be orphan.

Integrating with the transcoding technology, the watermark can be a trigger to initiate transcoding of the content according to viewer's or listener's preferences. This topic is presented in more detail in the following chapter.

3.4.2 Semantic Interoperability by Ontology Annotation

To achieve real data interoperability on the Web, Web application systems must be able to exchange data in such a way that the precise meaning of the data is readily accessible and the data itself can be translated by any system into a form that it understands.

There are numerous benefits of semantic interoperability. For example, searches can often be frustrating because of the limitations of

keyword-based matching techniques. Users frequently experience one of two problems: they either get back no results or too many irrelevant results.

The problem is that words can be synonymous (that is, two words have the same meaning) or polysemous (a single word has multiple meanings). However, if the languages used to describe Web documents were semantically interoperable, then the user could specify a query in the terminology that was most convenient, and be assured that the correct results were returned, regardless of how the data was expressed in the sources.

Intelligent software agents could also benefit from semantic interoperability. Currently, such agents are very sensitive to changes in the format of Web resources. Although these agents would not be affected by presentation changes if the content was available in XML, they would still break if the XML representation of the data was changed slightly (for example, if the element `<price>` was renamed to `<cost>`). Semantic interoperability would allow agents to be more robust. A useful function for Internet agents would be the ability to automatically integrate information from multiple sources. For example, a travel-planning agent might need to access data from airlines, hotels, rental car companies, weather sites, and tourist sites, each of which may have different ways of representing the relevant data. Such an agent would be faced with the problem of translating the different vocabularies and representations for this data into a common format that it could work with.

An ontology-based knowledge representation language called *Simple HTML Ontology Extensions* (SHOE) was designed by Hendler's group at the University of Maryland [29]. SHOE uses knowledge-oriented elements and associates meaning with content by making each Web document commit to one or more ontologies. The term ''ontology'' is used by AI researchers to mean a formal specification of the concepts and relations of some domain. Thus, an RDF Schema can be considered an ontology, although it is more limited than most AI ontologies. SHOE ontologies permit the discovery of implicit knowledge through the use of taxonomies and inference rules, allowing content providers to encode only the necessary information on their Web content, and to use the level of detail that is appropriate to the context. Interoperability is promoted through the sharing and reuse of ontologies.

Once we have some ontologies to work with, users can embed some semantic knowledge into their Web page using SHOE. Here is the sample HTML document with SHOE annotation.

```
<HTML>
<HEAD>
<TITLE>Nagao's Page</TITLE>
</HEAD>
<BODY>
 <P>
  This is Katashi Nagao's Web page.
  I am a professor at Nagoya University.
 </P>
 <P>Also, I'm 40 years old.</P>
 <P>
  Here is a pointer to my
<A HREF="http://www.nagao.nuie.nagoya-u.ac.jp/members/
     index.xml">
  lab members.
 </A>
</P>
<P>
 And a paper I recently wrote is
<A HREF="http://www.nagao.nuie.nagoya-u.ac.jp/papers/
     coling2002.pdf">
  here.
</A>
</P>

<h3>Kazue Nagao</h3>

<P>
 Kazue is a visiting lecturer here who doesn't have her own Web
 page. I have agreed to let part of my Web page space belong to her.
</P>
<INSTANCE KEY="http://www.nagao.nuie.nagoya-u.ac.jp/
     members/nagao/">
   <USE-ONTOLOGY
     ID="cs-dept-ontology"
     URL="http://www.cs.umd.edu/projects/plus/
        SHOE/onts/cs.html"
     VERSION="1.0"
     PREFIX="cs">
    <CATEGORY NAME="cs.professor">
```

```
    <RELATION NAME="cs.name">
      <ARG POS="TO" VALUE="Katashi Nagao">
    </RELATION>
    <RELATION NAME="cs.age">
       <ARG POS="TO" VALUE="40">
    </RELATION>
    <RELATION NAME="cs.labmembers">
       <ARG POS="TO"
         VALUE="http://www.nagao.nuie.nagoya-u.ac.jp/
         members/">
    </RELATION>
    <RELATION NAME="cs.paper">
      <ARG POS="TO"
         VALUE="http://www.nagao.nuie.nagoya-u.ac.jp/
         papers/coling2002.pdf">
    </RELATION>
<INSTANCE KEY="http://www.nagao.nuie.nagoya-u.ac.jp/
     members/nagao/#kazue">
    <CATEGORY NAME="cs.lecturer">
    <RELATION NAME="cs.name">
      <ARG POS="TO" VALUE="Kazue Nagao">
    </RELATION>
  </INSTANCE>
</INSTANCE>

</BODY>
</HTML>
```

This tells us that:

- The Web page is about Katashi Nagao.
- The person's name is Katashi Nagao.
- The person is a professor.
- The person is 40 years old.
- The person's laboratory members are on http://www.nagao.nuie.nagoya-u.ac.jp/members/index.xml.
- The person is the author of the paper on http://www.nagao.nuie.nagoya-u.ac.jp/papers/coling2002.pdf.

Further, some additional facts about Kazue Nagao are also described:

- Her name is Kazue Nagao.
- She is a lecturer.

It so happens that we want to tell these things to intelligent agents and other knowledge gatherers. To do this, we first need to tell the robot that we are using SHOE and uniquely define our document as an instance. An instance is similar to an ''entity'' as defined in the database field. Web authors should begin by declaring that their Web documents use SHOE 1.0-compliant tags. To do this, add the following in the HEAD section of the document:

```
<META HTTP-EQUIV="SHOE" CONTENT="VERSION=1.0">
```

We need to tell the agent which ontology we are using to structure those facts. Without an ontology, agents would have no idea what ''professor'' means when we claim that that is what we are. Web authors will use the CS Department ontology they partially described previously. Let us suppose that the ontology is stored at http://www.cs.umd.edu/projects/plus/SHOE/onts/cs.html and it is called cs-dept-ontology version 1.0. To indicate that the author is using this particular ontology, he or she should declare:

```
<USE-ONTOLOGY
     ID="cs-dept-ontology"
     URL="http://www.cs.umd.edu/projects/plus/SHOE/onts/cs.
       html"
     VERSION="1.0"
     PREFIX="cs">
```

The PREFIX indicates that all references the author makes to elements declared in the cs-dept-ontology ontology will be prefixed with a ''cs.'' prefix. Web authors can use as many ontologies as they like, as long as each has a unique prefix.

XML can revolutionize the Web, but semantic interoperability will be necessary to achieve the Web's true potential. The researchers of SHOE have discussed the limitations of XML and RDF with respect to semantic interoperability and explained how it is better suited for semantics on the Web than either XML DTDs or RDF. Using the ontology framework, Web authors can associate semantics of information with the original content. Since ontology-based approaches seem top-down, there may be a big gap

between the current ill-defined human-centered Web content and the well-defined machine-centered Web content. In Chapter 4, another approach to semantic annotation will be presented. This approach is bottom-up and allows ordinary people to annotate their Web documents with some helpful information for machines to infer their meanings.

3.4.3 Resource Linking and Meta-Annotation

The very nature of the success of the Web lies in its capability for linking resources. However, the unidirectional, simple linking structures of today's Web are not enough for the growing needs of an XML world. XLink as the W3C solution for resource linking in XML can remarkably improve this situation [30].

The linking model that has been employed in the traditional Web model is very simplistic. Essentially, the standard `<A HREF="http://...">` link is a simple static, directional, single-source, single-destination link that is embedded into the source document. It is static because the link never changes (at least not once it has been coded into an HTML page). It is directional because the link has an explicit direction of association (and hence, usually, an explicit direction of navigation). It is single-source because the link only has one point from which it can be triggered. It is single-destination because the link only has one resource to which the client traverses when the user activates the link. The link is embedded into the source document (indeed, it is embedded within the source anchor) because the description of the connection between the source and destination anchors exists within the source anchor.

Even this simple model highlights a number of different linking concepts, which in turn indicate aspects that, if changed, can lead to more sophisticated linking. To start with, we have identified an addressable unit of information or service. In other words, basically anything that we are able to request, including Web pages, XML documents, images, audio clips, program outputs, etc. A resource (in the URI sense of the word) is not necessarily a computer-related thing but could, at least conceptually, be anything you would like to deal with, such as paper documents or animals or whatever. Indeed, by utilizing standards for addressing fragments of these things, we can also treat fragments as resources. It is worth noting that Web standards distinguish between addressing a whole resource (referred to as an identifier, and typically implemented using URIs) and addressing a particular subresource (implemented using resource-type-specific fragment identifiers).

The linking model for XML revolves around the complementary technologies of XLink and XPointer. XPointers provide the mechanism for identifying resource fragments, and XLink provides the mechanism for collecting these together into links.

3.4.3.1 XPointer

XPointer provides a general way to select fragments of an XML document, essentially by writing a set of expressions [10]. An expression is evaluated with respect to the current context (which includes aspects such as bindings between variables and values, and a context element within the given XML document), and usually results in a set of locations (not surprisingly, referred to as a location set). An expression can be used to select children, siblings, nodes with given attributes, arbitrary text strings, and so on. For example, the XPointer

```
xpointer(//child::body[position()=1]/child::p)
```

selects all p children elements of the first body element in the document.

As another example, the XPointer expression

```
xpointer(/descendant::*[attribute::name='book'])
```

selects all elements (i.e., all descendants of the document root) that have an attribute called name with a value of book. In effect, the selection mechanisms can be concatenated to progressively identify a specific subset of nodes.

It is also worth briefly pointing out that XPointer is actually an application of the XPath standard. XPath was developed specifically to be a foundation for other standards such as XPointer. Another example application of XPath is as part of XSLT. In XSLT, XPath expressions allow the selection of specific nodes that can then be transformed into an alternative form (often for presentation).

Since XPath is intended to be relatively generic (to suit multiple applications), there are certain sections of documents that cannot be specified using XPath expressions. For example, both of the XPointer fragments shown above are constructed from valid XPath expressions. XPath cannot, however, select an arbitrary string that crosses several nodes. It is in areas such as this that XPointer has extended XPath. For example, the following expression defines the set of all occurrences of the string "content" within all para elements in the http://www.nagao.nuie.nagoya-u.ac.jp/members/nagao.xml resource (this could not be achieved using just XPath expressions):

```
http://www.nagao.nuie.nagoya-u.ac.jp/members/nagao.xml#xpointer
(string-range(//para, 'content'))
```

As one further example, the following URI defines a range that extends from the beginning of the element with an ID of sect-3, to the end of the element with an ID of sect-4:

```
http://www.nagao.nuie.nagoya-u.ac.jp/members/nagao.xml#xpointer
(id('sect-3')/range-to(id('sect-4')))
```

3.4.3.2 Linkbases

An interesting issue of XLink is that of adding links from read-only material. This is a rather unusual concept for people who are only familiar with the Web hypertext, where all links must be embedded into the source content, which is very restrictive. For example, an author might want to be able to annotate material that does not belong to him or her, or the material may be stored on read-only media, or the author may want to use different sets of links at different times (depending upon what he or she is trying to do). In each case, the author cannot add links directly into the underlying content. Instead, he or she would like to be able to specify links separately and somehow have them used. Using XLink, this is relatively straightforward. Consider the following relatively simple example:

```
<?xml version="1.0"?>
<!DOCTYPE dictionary SYSTEM "dictionary.dtd">
<dictionary>
   <entry word="anchor">
     <pronunciation>...</pronunciation>
     <definition>
      An identified region of a node that can be explicitly
      addressed and identified within the presentation of
      a node.
     </definition>
   </entry>
   <entry word="link">
     <pronunciation>...</pronunciation>
     <definition>
      A connection between multiple anchors (and nodes,
      where there is an implied anchor that encompasses
```

```
        the entire node) that represents an association
        between the concepts captured by the anchors.
      </definition>
    </entry>
    <!-- Further entries go here -->
</dictionary>
```

In this example, the document contains words and definitions but no links. The Web author can then create a separate file (often called a linkbase) that contains links from words to the definitions of these words.

```
<?xml version="1.0"?>
<!DOCTYPE xrefs SYSTEM "xrefs.dtd">
<xrefs>
 <xref xlink:type="extended">
  <word xlink:type="locator"
    xlink:href="#xpointer(string-range(//definition,
    'anchor'))"
    xlink:label="src"/>
  <defn xlink:type="locator"
    xlink:href="Dict.xml#xpointer(//entry[@word=
    'anchor')"
    xlink:label="dest"/>
  <go xlink:type="arc"xlink:from="src" xlink:to="dest"/>
 </xref>
 <! -- Further cross references go here -->
</xrefs>
```

This linkbase file contains a series of XLinks, which link any occurrence of specific words in the definitions to the definition of those words. For example, the word "anchor" appearing in the definition of the word "link" would be the starting point for a link to the definition of anchor. In this case, the links are termed third-party links. This is because they are not embedded into any of the anchors that are participating in the link.

3.4.3.3 Multiended Links

Another aspect supported by XLink but not supported by HTML linking is multiended links. This type of link has a number of important applications essentially wherever we have more than two resources that participate in

a relationship. There are, however, several ways in which we can create links of this type. First, consider the following example:

```
<family xlink:type="extended">
   <loc xlink:type="locator" xlink:label="parent"
    xlink:href="p1.xml"/>
   <loc xlink:type="locator" xlink:label="parent"
    xlink:href="p2.xml"/>
   <loc xlink:type="locator" xlink:label="child"
    xlink:href="c1.xml"/>
   <loc xlink:type="locator" xlink:label="child"
    xlink:href="c2.xml"/>
   <loc xlink:type="locator" xlink:label="child"
    xlink:href="c3.xml"/>
   <go xlink:type="arc" xlink:from="parent"
    xlink:to="child"/>
</family>
```

In this example, there are five participating resources. There is also a single arc specification which results in six arcs (p1-c1, p1-c2, p1-c3, p2-c1, p2-c2, p2-c3). Therefore, there are three arcs from "p1.xml." If the author initiates traversal of the arcs that originate from this resource, then XLink does not specify what should happen though a typical behavior might be to provide the user with a list of the possible destinations and allow them to select the appropriate arc to traverse. It is also worth noting that XLink does not specify how to trigger a traversal, which is largely left to XLink applications (except for the standard behavior attributes).

XLink also supports a second form of multiended link, though it is a little more subtle than the above example. Consider the following:

```
<family xlink:type="extended">
   <loc xlink:type="locator" xlink:label="parents"
     xlink:href="family.xml#xpointer(//person
     [@type='parent')"/>
   <loc xlink:type="locator" xlink:label="children"
     xlink:href="family.xml#xpointer(//person
     [@type='child')"/>
   <go xlink:type="arc" xlink:from="parents" xlink:
     to="children"/>
</family>
```

In this example, there is a single arc specification, which results in just a single arc. The key is in the XPointer. Essentially, the XPointer selects a location set, which in the case of the parents locator, can potentially be a set of multiple noncontiguous elements. The same applies to the children locator. In effect, we have a single arc from one subresource to another subresource, but both subresources can potentially be location sets with more than one member.

Again, XLink does not specify how this situation, once specified in XLink, should be interpreted in terms of navigational behavior. In particular, the standard states: Some possibilities for application behavior with noncontiguous ending resources might include highlighting of each location, producing a dialog box that allows the reader to choose among the locations as if there were separate arcs leading to each one, concatenating the content of all the locations for presentation, and so on. Application behavior with noncontiguous starting resources might include concatenation and rendering as a single unit, or creating one arc emanating from each contiguous portion.

It is also worth pointing out that XLink supports annotating arcs with additional semantics such as titles and roles. The authors can define two arcs between the same two resources, but with different purposes as indicated by their roles.

3.4.3.4 Generic Links

A link that had the same anchor text can be created by simply adding a new anchor (and associated link) for each occurrence of the relevant phrase, but this could become extremely cumbersome and difficult to maintain if the word occurred often.

The use of generic links solves this problem. By defining a link that has as its source anchor any occurrence of the relevant text, Web authors can effectively create a link that is much easier to maintain. For example, the following XPointer refers to any occurrence of the text ''transcode'' within element content in a document (though it will not match attribute values and any XML markup such as element or attribute names):

```
xpointer(string-range(//*, 'transcode'))
```

This XPointer can then be used to create a link from all occurrences of this text to the relevant definition of this text in a file containing a list of definitions. For example, the following third-party extended link provides a link from all occurrences of the words ''transcode'' and ''transcoding'' within the link.xml file to an appropriate definition in the defs.xml file (note that this link can be stored in a separate file unrelated to either):

```
<extendedlink xlink:type="extended">
   <loc xlink:type="locator"
     xlink:href="links.xml#xpointer
    (string-range(//*, 'transcode'))
    "xlink:role="phrase"/>
   <loc xlink:type="locator"
     xlink:href="links.xml#xpointer(string-range
    (//*, 'transcoding'))"xlink:role="phrase"/>
   <loc xlink:href="defs.xml#xpointer(//defn
    [phrase='transcode']) "xlink:role="definition"/>
    <go xlink:type="arc"xlink:from="phrase"
     xlink:to="definition" xlink:show="new"
     xlink:actuate="onRequest"/>
</extendedlink>
```

3.4.3.5 Meta-Annotation

Using XLink, Web authors can define annotation about annotation called *meta-annotation*. Meta-annotation is needed when an annotation includes something wrong such as misunderstanding comments or descriptions, and the annotator is not aware of his or her misunderstanding. In this case, another annotator will create a note to correct the wrong comments and associate it with them. The relationships between annotations are also defined by XLink if these annotations are described as XML (or RDF) documents.

A set of content, annotation, and meta-annotation constructs a super-structure on the Web as illustrated in Section 1.3. Annotations including meta-annotations are not necessarily a human-readable content, but they are extremely usable for machines to understand meanings of directly or indirectly annotated content. The next chapter will discuss transcoding of semantically annotated content. The proposed framework is based on annotation schemes and resource linking models presented in this chapter.

References

[1] Dublin Core Metadata Initiative, "Dublin Core," http://dublincore.org/, 2002.

[2] W3C, "Naming and Addressing: URIs, URLs, ...," http://www.w3.org/Addressing/, 2002.

[3] The International DOI Foundation, "The Digital Object Identifier," http://www.doi.org/, 2002.

[4] The International ISBN Agency, "International Standard Book Number (ISBN) System," http://www.isbn-international.org/, 2002.

[5] Lagoze, C., "The Warwick Framework: A Container Architecture for Diverse Sets of Metadata," http://www.dlib.org/dlib/july96/lagoze/07lagoze.html, 1996.

[6] W3C, "RDF Site Summary (RSS) 1.0," http://groups.yahoo.com/group/rss-dev/files/specification.html, 2002.

[7] MPEG, "MPEG-7 Context and Objectives," http://drogo.cselt.stet.it/mpeg/standards/mpeg-7/mpeg-7.htm, 2002.

[8] W3C, "XML Schema," http://www.w3.org/XML/Schema, 2002.

[9] Kahan, J., et al., "Annotea: An Open RDF Infrastructure for Shared Web Annotaions," *Proc. 10th International World Wide Web Conference (WWW-10)*, 2001.

[10] W3C, "XML Pointer Language (XPointer) Version 1.0," http://www.w3.org/TR/xptr/, 2001.

[11] Takagi, H., et al., "Site-Wide Annotation: Reconstructing Existing Pages to Be Accessible," *Proc. the Fifth International ACM Conference on Assistive Technologies (ASSETS 2002)*, 2002.

[12] W3C, "Amaya Home Page," http://www.w3.org/Amaya/, 2002.

[13] W3C, "XML Linking Language (XLink) Version 1.0," http://www.w3.org/TR/xlink/, 2001.

[14] W3C, "Algae HOWTO," http://www.w3.org/1999/02/26-modules/User/Algae-HOWTO.html, 2002.

[15] HyperNews.org, "HyperNews Home Page," http://www.hypernews.org/, 2001.

[16] Foresight Institute, "CritLink: Better Hyperlinks for the WWW," http://crit.org/, 2001.

[17] ThirdVoice, "ThirdVoice Home Page," http://www.thirdvoice.com/, 2001.

[18] iMarkup Solutions, "iMarkup," http://www.imarkup.com/, 2002.

[19] Denoue, L., and L. Vignollet, "An Annotation Tool for Web Browsers and Its Applications to Information Retrieval," *Proc. RIAO 2000*, http://www.univ-savoie.fr/labos/syscom/Laurent.Denoue/publications/riao2000.pdf, 2000.

[20] W3C, "XML Path Language (XPath) Version 1.0," http://www.w3.org/TR/xpath/, 1999.

[21] Vetro, A., and S. Devillers, "Delivery Context in MPEG-21," http://www.w3.org/2002/02/DIWS/submission/sdevillers-phillips-position.html, 2002.

[22] Wireless Application Protocol Forum, "WAG UAProf," http://www1.wapforum.org/tech/documents/WAP-248-UAProf-20010530-p.pdf, 2001.

[23] HTTP Working Group, "Transparent Content Negotiation in HTTP," http://gewis.win.tue.nl/koen/conneg/draft-ietf-http-negotiation-01.html, 1997.

[24] Kudo, M., and S. Hada, "XML Document Security Based on Provisional Authorization," *Proc. 7th ACM Conference on Computer and Communication Security (CCS'00)*, 2000, pp. 87–96.

[25] W3C, "XML-Signature Syntax and Processing," http://www.w3.org/TR/xmldsig-core/, 2002.

[26] Damiani, E., et al., "Securing XML Documents," *Proc. EDBT 2000,* (Lecture Notes in Computer Science, Vol. 1777), 2000, pp. 121–135.

[27] Blaze, M., J. Feigenbaum, and A. Keromytis, "KeyNote: Trust Management for Public-Key Infrastructures," *Proc. Cambridge Security Protocols International Workshop,* (Lecture Notes in Computer Science Vol. 1550), 1999, pp. 59–63.

[28] Digimarc Corporation, "Digimarc Home Page," http://www.digimarc.com/, 2002.

[29] Heflin, J., and J. Hendler, "Semantic Interoperability on the Web," *Proc. Extreme Markup Languages 2000*, 2000, pp. 111–120.

[30] Lowe D., and E. Wilde, "Improving Web Linking Using XLink," *Proc. Open Publish 2001*, http://www.binarything.com/binarything/openpublish/DavidLowe1.pdf, 2001.

4

Semantic Annotation and Transcoding: Towards Semantically Sharable Digital Content

This chapter presents an annotation-based approach to an emerging digital content that is semantically sharable between humans and machines. In order to create such content, we have to realize ''information grounding,''—that is, grounding information on the human life world, which is an essential process for machines to share meaning of information with humans. It is discussed in more detail in Section 4.1.2.

Recently, a project aimed at this type of grounding on the Web has been promoted by W3C. It is called the *Semantic Web* [1]. One of the basic milestones in the road to the Semantic Web is the association of well-defined descriptions to content. The descriptions allow the Web developers to extract and process properties about some given content, even if the medium of the content does not directly provide the necessary means to do so.

The Semantic Web is not designed as a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The first steps in weaving the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will lead to significant new functionality as machines become much better able to process and understand the data that they merely display at present.

The Semantic Web is mainly advocating an ontology-based approach to a machine-understandable Web that requires formal descriptions of concepts contained in Web content. It seems to be top-down, because such formal descriptions should cover more than one content and must be universal.

Another approach, an annotation-based approach, is the main topic in this chapter. Since an annotation concerns one particular content and can be modified in an incremental manner by meta-annotations, the annotation-based approach must be bottom-up. In this chapter, annotations have a definite role, which is to provide for machines to autonomously infer the semantics of the target content. Annotated content is easier for computers to understand and process, allowing personalized content to be created with much less effort and greater quality. This permits content providers to reach a much wider audience with minimal overhead.

The annotation presented in this chapter has three categories. One is linguistic annotation, which helps the transcoder understand the semantic structure of textual elements. The second is commentary annotation, which helps the transcoder manipulate both textual and nontextual elements such as images and sounds. The third category is multimedia annotation, which is a combination of the first two types.

My group has developed a system for semi-automatic and interactive Web content annotation, allowing users to annotate any element of any Web document or multimedia data with additional information. We have also developed a proxy server that transcodes requested content using information from annotations assigned to them. I call the entire process *semantic transcoding*, because we provide means for easily transcoding annotated documents using information about their deep semantic content [2]. The current semantic transcoding process handles mainly text and video summarization, language translation, and speech synthesis of documents containing text and images.

The ontology-based and annotation-based approaches are not at all contradictory but rather complementary to each other for information grounding. Their descriptions should be closely aligned with an original content (annotation-based approach) if the original content is available, and formal semantics that the Semantic Web is pursuing should be taken into more serious consideration (ontology-based approach) if there is no original content. Of course, further study is necessary on how to integrate the two approaches.

More about the semantics of content and the ontology in the Semantic Web is discussed in the following section.

4.1 Semantics and Grounding

For the Semantic Web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning. Artificial intelligence researchers have studied such systems since long before the Web was developed. *Knowledge representation*, as this technology is called, is clearly a good idea, and some very nice demonstrations exist, but it has not yet changed the world. It contains the seeds of important applications, but to realize its full potential it must be linked into a single global system.

Semantic Web researchers, in contrast, accept that inconsistencies and unanswerable questions are a price that must be paid to achieve versatility. They are developing the language for the rules as expressive as needed to allow the Web to reason as widely as desired. This philosophy is similar to that of the conventional Web: early in the Web's development, detractors pointed out that it could never be a well-organized library; without a central database and tree structure, one would never be sure of finding everything. They were right. But the expressive power of the system made vast amounts of information available, and search engines (which would have seemed quite impractical a decade ago) now produce remarkably complete indices of a lot of the material out there. The challenge of the Semantic Web, therefore, is to provide a language that expresses both data and rules for reasoning about the data and that allows rules from any existing knowledge representation system to be exported onto the Web.

4.1.1 Ontology

In the Semantic Web, meaning of information is expressed by RDF [3](see Section 3.1.3), which encodes it in sets of triples, each triple being rather like the subject, verb, and object of an elementary sentence. These triples can be written using XML tags. In RDF, a document makes assertions that particular things (people, Web resources, or whatever) have properties (such as "is a kind of," "is a part of") with certain values (another person, another Web resource). This structure turns out to be a natural way to describe the vast majority of the data processed by machines. Subject and object are each identified by a URI [4], just as used in a link on a Web document. The verbs are also identified by URIs, which enables anyone to define a new concept, a new verb, just by defining a URI for it somewhere on the Web.

Of course, this is not the end of the story, because two databases on the Web may use different identifiers for what is in fact the same concept.

A program that wants to compare or combine information across the two databases has to know that these two terms are being used to mean the same thing. Ideally, the program must have a way to discover such common meanings for whatever databases it encounters.

A solution to this problem is provided by the third basic component of the Semantic Web: collections of information called *ontologies*. In philosophy, an ontology is a theory about the nature of existence, of what types of things exist; ontology as a discipline studies such theories. Artificial intelligence researchers have been using the term to mean a document or file that formally defines the relations among terms. The most typical kind of ontology for the Web has a taxonomy and a set of inference rules.

The taxonomy defines classes of objects and relations among them. For example, an address may be defined as a type of location, and city codes may be defined to apply only to locations, and so on. Classes, subclasses, and relations among entities are a very powerful tool for Web use. We can express a large number of relations among entities by assigning properties to classes and allowing subclasses to inherit such properties. If city codes must be of type city and cities generally have Web sites, we can discuss the Web site associated with a city code even if no database links a city code directly to a Web site.

Inference rules in ontologies supply further power. An ontology may express the rule, ''If a city code is associated with a country code, and an address uses that city code, then that address has the associated country code.'' A program could then readily deduce, for instance, that a Nagoya University address, being in Nagoya, must be in Japan, and therefore should be formatted to Japanese address standards. The computer does not truly understand any of this information, but it can now manipulate the terms much more effectively in ways that are useful and meaningful to the human user.

With ontology data on the Web, solutions to terminology (and other) problems begin to emerge. The meaning of terms or XML data used on a Web document can be defined by pointers from the document to an ontology. Of course, the same problems as before now arise if I point to an ontology that defines addresses as containing a zip code and you point to one that uses a postal code. This kind of confusion can be resolved if ontologies provide equivalence relations: one or both of our ontologies may contain the information that my zip code is equivalent to your postal code.

The scheme of the Semantic Web for sending in the clowns to satisfy my customers is partially solved when the two databases point to different definitions of address. The program, using distinct URIs for different

concepts of address, will not confuse them and in fact will need to discover that the concepts are related at all. The program could then use a service that takes a list of postal addresses (defined in the first ontology) and converts it into a list of physical addresses (the second ontology) by recognizing and removing post office boxes and other unsuitable addresses. The structure and semantics provided by ontologies make it easier for an entrepreneur to provide such a service and can make its use completely transparent.

Ontologies can enhance the functioning of the Web in many ways. They can be used in a simple fashion to improve the accuracy of Web searches—the search program can look for only those pages that refer to a precise concept instead of all the ones using ambiguous keywords. More advanced applications will use ontologies to relate the information on a page to the associated knowledge structures and inference rules.

4.1.2 Information Grounding

In order to make information technology useful for humans at all, we essentially need *information grounding*[1] of digital data on the human life world, which is to allow machines to share meaning with humans [6].

Then, we can get *intelligent content* from digital content structured for the sake of information grounding in this sense (i.e., structured so that humans and machines can share the meaning of the original content). Here, the original content may be natural language documents, visual/auditory content, or any other data in any modality. Information grounding should allow retrieval, translation, summarization, and presentation of such content, with very high quality and accuracy. Initiatives to promote intelligent content include MPEG-7 [7] (see Section 3.1.4), *Global Document Annotation* (GDA) [8][2], which is explained later (Section 4.3.4.1), and the Semantic Web.

For intelligent content, there ar two major approaches to information grounding, which are complementary to each other.

1. The symbol grounding problem [5] was originally posited as a problem about how digital information could have inherent meaning of the real world, rather than meaning externally assigned through interpretation by the human theoretician or programmer. The original argument on grounding tended to regard the real world as the physical world, but in the present discussion the real world is the human life world, which encompasses not just the physical aspects but also conceptual, social, and other aspects.
2. A simplified version of the GDA tag set, called the Linguistic DS (Description Scheme), has been proposed to MPEG-7 and is scheduled to be a part of the amendment of MPEG-7 in 2003.

The ontology-based approach, mainly advocated by the Semantic Web, aims at grounding in the classical sense that machines can autonomously carry out sound inferences reflecting the real world, based on formal ontologies associated with the formal descriptions as part of the information content. Information grounding is guaranteed here by the validity of the ontology prescribed by human developers.

GDA and MPEG-7, on the other hand, are regarded as putting more emphasis on the annotation-based approach to information grounding. This approach aims at grounding by fine-grained association between the semantic description and the original content. In GDA, for instance, XML tags are embedded in text data, so that the formal description in terms of the tags is closely aligned with the raw text data.

Similarly, MPEG-7 can align XML-based descriptions with the original multimodal data, though the descriptions are external to the original data.

Information grounding is maintained here by the interaction between machines and humans, rather than autonomous inferences by machines, as illustrated in Figure 4.1.

The lines in the figure represent various types of interactions, where the double-lined ones encode more minute and meaningful interactions. That is, humans can readily understand and thus meaningfully interact with the original content. Machines can understand and manipulate semantic descriptions more readily than the original content. Meaningful interactions

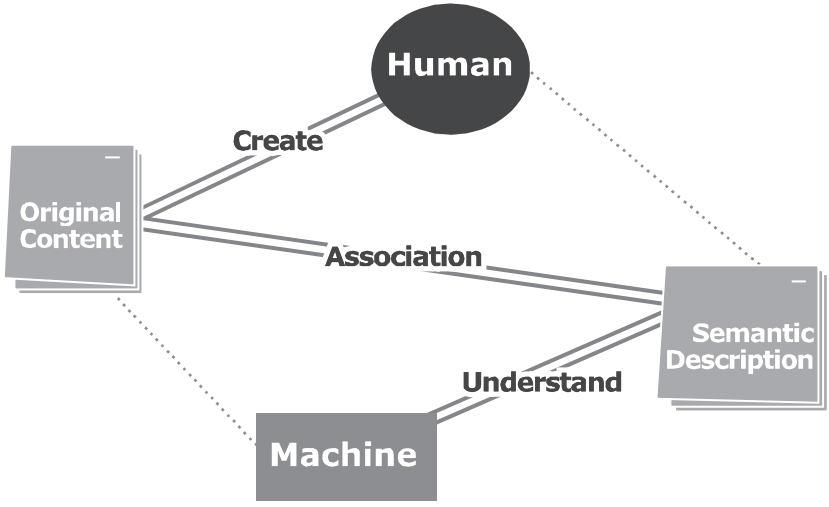


Figure 4.1 Content-description association and information grounding.

between humans and machines are hence supported by associating the semantic descriptions and the original content.

The association realizes information grounding in this sense, and a number of merits follow.

First of all, the association is obviously useful and necessary for summarization, translation, and retrieval. Summarization/translation of the original content using the formal description is possible owing to the association, provided that the description is first summarized/translated and the resulting summarization/translation is mapped to the original content to generate a desired result.

Another major merit of the association is that we can dispense with formal ontology to a substantial extent. Namely, automatic inferences using formal ontologies may be replaced by interactions with humans, because they understand the meaning of the original content and this understanding is communicated to machines due to the association. For instance, inferences involved in information retrieval can be carried out by interactions between the human user and the machine, using a linguistic thesaurus rather than a full-fledged ontology. Information retrieval potentially involves very intricate inferences to plug the gaps between the query and the database. For several decades to come, it will be impossible to fully automate such inferences, even with a very nice formal ontology. The only feasible means of such inferences is some human-machine interaction, which may not require any formal ontology.

A third important merit is that the association between the original content and the description facilitates the authoring of the description, because it allows humans to produce the description while closely referring to the original content, rather than totally from scratch. Also, a fine-grained association makes it easy to keep track of which parts of the original content have already been described and which others have not. Furthermore, the association allows humans and machines to effectively collaborate in the authoring process. For instance, a human annotator could first partially describe the content and the machine could add descriptions to the other parts of the content.

In this chapter, an annotation-based approach to intelligent content is mainly presented. This approach is bottom-up and our annotation system allows ordinary people to annotate their Web documents with some additional and useful information by using user-friendly tools. Before presenting annotation techniques, I want to explain some underlying technologies of automated analysis of digital content that can accelerate humans' creation of semantic descriptions.

4.2 Content Analysis Techniques

Annotating content with semantic information strongly depends on automatic analysis of digital content such as natural language text, speech data, and video clips. Here I explain some basic techniques on automatic content analysis including sentence analysis and disambiguation, speech recognition, and video scene/object detection. These techniques have been integrated with semantic annotation systems explained later. In these annotation systems, the most of humans' efforts will be effectively applied to fixing the remaining problems that machines could not solve.

4.2.1 Natural Language Analysis

Natural language text is generally analyzed in four steps: morphological, syntactic, semantic, and discourse analyses. For each step, there are problems of misrecognition and ambiguity.

4.2.1.1 Morphological Analysis

Individual words are analyzed into their components, and nonword tokens (such as punctuation) are separated from the words. For example, in the phrase "Hanako's room," the proper noun "Hanako" is separated from the possessive suffix "'s."

A meaningful linguistic unit that contains no smaller meaningful parts is called a morpheme. Morphological analysis is used to identify morphemes according to the rules of inflection and derivation of words.

In the case of agglutinative language such as Japanese, the morphological analysis becomes a little bit more complicated because different separation patterns can be applied to a sentence to identify morphemes. For instance, generally, more than one word-division patterns may be applied to a longer sequence of hiraganas (syllabic characters).

4.2.1.2 Syntactic Analysis

Linear sequences of words are transformed into structures that show how the words relate to one another. This parsing step converts the flat list of words of the sentence into a structure that defines the units represented by that list. Constraints imposed include word order, number agreement, and case agreement.

For example, "secretary the key" is an illegal constituent in the sentence "I gave the secretary the key" bacause of the word order constraint. In the noun phrase "the pages of the book which are written by him," the relative

clause ''which are written by him'' must modify ''the pages'' not ''the book'' because of the number agreement constraint. In the sentence ''I love her hair,'' both ''I love her (accusative case)'' and ''her (genitive case) hair'' satisfy the case agreement constraint, but ''her hair'' can be an objective case of ''love,'' so the latter interpretation is selected.

Syntactic analysis of a sentence is often called sentence parsing. Syntactic analysis involves parsing the sentence to extract whatever information the word order contains such as a syntactic structure and a word category by applying grammar rules. Grammar is a declarative representation that defines the syntactic facts of a language. The most common way to represent grammars is as a set of production rules, and the simplest structure for them is to build a parse tree that records the rules and how they are matched.

Sometimes backtracking is required (e.g., ''The horse raced past the barn fell''), and sometimes multiple interpretations may exist for the beginning of a sentence (e.g., ''Have the students who missed the exam'').

Syntactic processing also interprets the difference between ''John hit Mary'' and ''Mary hit John.''

4.2.1.3 Semantic Analysis

The structures created by the syntactic analyzer are assigned meanings. In most contexts, the sentence ''Colorless green ideas sleep furiously''[3] would be rejected as semantically anomalous. This step must map individual words into appropriate objects in the world knowledge, and must create the correct structures to correspond to the way the meanings of the individual words combine with each other.

After (or sometimes in conjunction with) syntactic processing, a natural language system must still produce a representation of the meaning of a sentence, based upon the meanings of the words in it. The following steps are usually taken to do this:

1. *Lexical processing:* Look up the individual words in a dictionary. It may not be possible to choose a single correct meaning, since there may be more than one. The process of determining the correct meaning of individual words is called word sense disambiguation or lexical disambiguation. For example, ''I'm going to the bank'' can be understood since it requires either a time or a location. This usually leads to preference semantics when it is not clear which definition we should prefer.

3. Famous linguist Noam Chomsky used this example in his book [9].

2. *Sentence-level processing:* There are several approaches to sentence-level processing. These include semantic grammars, case grammars, and conceptual dependencies.

Semantic processing determines the differences between such sentences as ''The dog is in the pen'' and ''The ink is in the pen.''

4.2.1.4 Discourse Analysis

The meaning of an individual sentence may depend on the sentences that precede it and may influence the sentences yet to come. The entities involved in the sentence must either have been introduced explicitly or they must be related to entities that were. The overall discourse must be coherent. The structure representing what was said is reinterpreted to determine what was actually meant.

Discourse analysis should resolve anaphora in sentences—that is, the use of a word to refer to previously mentioned entities. To determine a discourse structure of a text is also a target of discourse analysis.

Rhetorical structure theory (RST) models the discourse structure of a text by means of a hierarchical tree diagram that labels relations between propositions expressed in text spans (typically clauses or larger linguistic units) [10]. The relations between nodes are of two kinds: symmetric and asymmetric. A symmetric relation involves two or more nodes, conventionally labeled nuclei, each of which is equally important in realizing the writer's communicative goals. An asymmetric relation involves exactly two nodes: a nucleus, the more important of the two in realizing the writer's communicative goals, and a satellite, a node in a dependency relation to the nucleus, modifying the nucleus in ways specified in the definition of the particular relation. Hierarchical structure arises from the fact that a nucleus or satellite node may have internal structure. Examples of the relations defined in RST, called rhetorical relations, are listed in Table 4.1. However, RST lacks an explicit method for constructing the discourse structure.

To understand most sentences, it is necessary to know the discourse and pragmatic context in which it was uttered. In general, for a program to participate intelligently in a dialog, it must be able to represent its own beliefs about the world, as well as the beliefs of others (and their beliefs about its beliefs, and so on). The context of goals and plans can be used to aid understanding. Plan recognition has served as the basis for many understanding programs.

Table 4.1
Examples of Rhetorical Relations

Relation Name	Nucleus	Satellite
Background	Text whose understanding is being facilitated	Text for facilitaing understanding
Circumstance	Text expressing the events or ideas occurring in the interpretive context	An interpretive context of situation or time
Concession	Situation affirmed by author	Situation that is apparently inconsistent but also affirmed by author
Condition	Action or situation whose occurrence results from the occurrence of the conditioning situation	Conditioning situation
Elaboration	Basic information	Additional information
Enablement	An action	Information intended to aid the reader in performing an action
Evaluation	A situation	An evaluative comment about the situation
Evidence	A claim	Information intended to increase the reader's belief in the claim
Interpretation	A situation	An interpretation of the situation
Justify	Text	Information supporting the writer's right to express the text
Motivation	An action	Information intended to increase the reader's desire to perform the action
Preparation	Text to be presented	Text that prepares the reader to expect and interpret the text to be presented
Purpose	An intended situation	The intent behind the situation
Restatement	A situation	A reexpression of the situation
Solutionhood	A situation or method supporting full or partial satisfaction of the need	A question, request, problem, or other expressed need
Summary	Text	A short summary of that text

Speech acts can be axiomatized just as other operators in written language, except that they require modal operators to describe states of belief and knowledge.

4.2.1.5 Ambiguities in Natural Language Analysis

Some intermediate representations are generated in each step of sentence analysis. Figure 4.2 shows intermediate states and problems in natural language analysis [11]. The morphological analysis phase is omitted in this figure.

Each analysis is considered as a mapping from an input data to one or more output representations. Multiple output structures are caused by ambiguities in natural language. Such ambiguities present major problems in the development of natural language systems.

I previously developed a technique for the semantic analysis of sentences, including an ambiguity-packing method that generates a packed representation of individual syntactic and semantic structures [12]. This packed representation is based on a dependency structure with constraints that must be satisfied in the syntax-semantics mapping phase. Since the ambiguities in sentences sometime cause a combinatorial explosion (this occurs when a huge number of possible combinations are created by increasing the number of

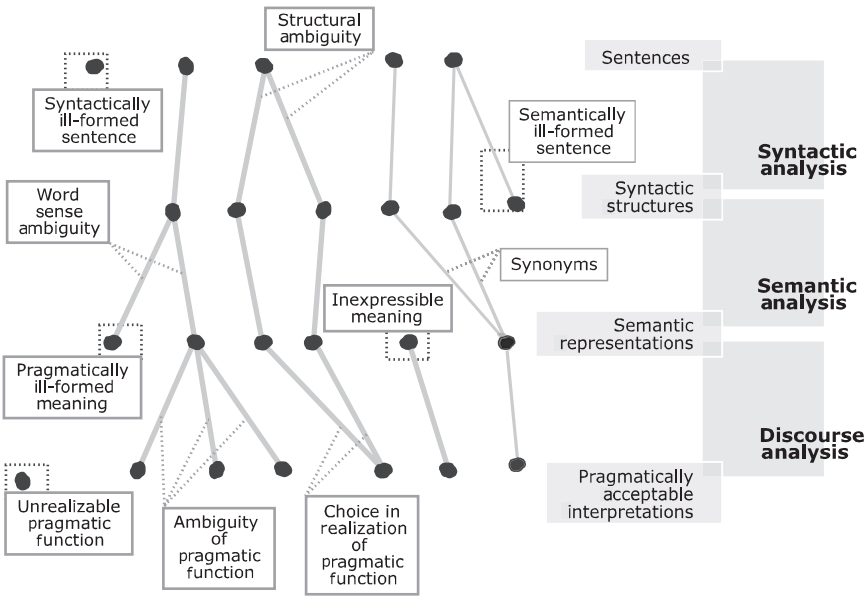


Figure 4.2 Layers of natural language analysis. (*From:* [11]. © 1989 Addison-Wesley Inc.)

entities that can be combined—forcing us to consider a constrained set of possibilities when we consider related problems), complete syntax-semantics mapping should not be performed until all ambiguities have been resolved without explicitly generating semantic structures.

A *preferential constraint satisfaction technique* is used for disambiguation and semantic analysis. Constraints are symbolic or combinatorial and restrict sentence structures, while preferences are numerical and compare the candidates of structures. Preferences are acquired by using both the taxonomies of a conceptual lexicon, which are called natural language classes, and examples of dependency structures.

4.2.1.6 Dependency Structures

A dependency structure is an ambiguity-packed syntactic representation of sentence structures. It is similar to a syntactic graph [13]. It explicitly represents modifier-modifiee relationships between words.

A dependency structure is a tree structure that consists of nodes and arcs. A node includes some syntactic features (e.g., `WORD`, `MOOD`, `TENSE`, `DET`, `NUM`, and `PREP`). `WORD` has some attributes such as `cat`, `subcat`, and `person`. Arcs correspond to syntactic relationships between sentence constituents (e.g., `SUBJECT`, `DOBJECT`, and `PPADJUNCT`). For example, the dependency structure for the sentence "Insert a diskette into the drive" is represented in an XML format as follows:

```
<PRED>
   <WORD cat="v" subcat="trans">insert</WORD>
   <MOOD>imperative</MOOD>
   <TENSE>present</TENSE>
   <DOBJECT>
     <WORD cat="n" person="3">diskette</WORD>
     <DET>a</DET>
     <NUM>singular</NUM>
   </DOBJECT>
   <PPADJUNCT>
     <WORD cat="n" person="3">drive</WORD>
     <PREP>into</PREP>
     <DET>the</DET>
     <NUM>singular</NUM>
   </PPADJUNCT>
</PRED>
```

This structure <PRED> is the root node of this dependency structure. <WORD>, <MOOD>, <TENSE>, <DOBJECT>, and <PPADJUNCT> are child nodes of <PRED> node. <WORD>, <DET>, and <NUM> are subnodes of <DOBJECT> and <PPADJUNCT>. <PREP> is another subnode of <PPADJUNCT>.

4.2.1.7 Natural Language Classes

For expressing the meaning of a sentence, an object-oriented conceptual representation, called the *natural language* (NL) *class* is employed [14]. NL objects, which are particular instances of NL classes, define a set of word senses. A domain ontology of a terminology can also be defined in the framework of NL class.

The set of NL classes consists of two special classes, *TOP and *BOTTOM, and two disjoint subsets of classes, open classes and closed classes. We have "is-a" relationships defined over NL classes. For any NL class *x, "*x is-a *TOP" and "*BOTTOM is-a *x" hold. Open classes correspond to entities which are expressed by nouns, verbs, adjectives, etc. Closed classes are used to represent attribute values of open classes. Typically, closed classes represent information conveyed by auxiliary verbs (aspects, tense, modals, etc.), determiners, inflections of verbs and nouns (number, gender, definiteness, etc.), and prepositions (relationships between two objects). The closed classes are stable and are common to almost all domains and natural languages. The open classes are comprehensive, but are dependent on both the domain and language.

A class is defined in terms of a *frame* format, as shown below. By convention, an open class name is preceded by an asterisk, a closed class name is preceded by an asterisk and a hyphen, and an instance has a hyphen and a number following its class name.

```
<defclass type="open">
  <concept>*insert</concept>
  <!-- definition of a class *insert -->
  <is-a>
    <value>*action</value>
    <!-- superclass is *action -->
  </is-a>
  <actor>
    <or>
      <sem>*human</sem>
      <sem>*system</sem>
```

```
    </or>
    <!-- actor filler must be *human or *system -->
  </actor>
  <theme>
    <sem>*physical-object</sem>
    <!-- theme filler must be *physical-object -->
  </theme>
  <goal>
    <or>
      <sem>*physical-object</sem>
      <sem>*location</sem>
    </or>
   <!-- goal filler must be *location or *physical-object -->
  </goal>
</defclass>
```

In the above example, the *value* facet of the `is-a` slot shows a filler (actual attribute value) of the slot. The *sem* facet of other slots shows "selectional restrictions" on their potential fillers. Actual fillers for these slots, except for the `is-a` slot, are not given in the class definition. The `is-a` slot is the only system-defined slot for open classes. All other slots are user-defined (i.e., domain-dependent). The `is-a` slot defines generalization relationships among NL classes, which roughly correspond to the taxonomy of words. A class can inherit each slot definition from its superclasses, unless the slot is redefined. Our actual NL hierarchy consists of several thousand classes [14] and the *Longman Dictionary of Contemporary English* (LDOCE) [15] word senses [16–18].

4.2.1.8 Syntax-Semantics Mapping

Since NL classes constitute the implementation of lexical word senses, the composition of NL objects can represent the meanings of phrases and sentences. Mapping rules are thus introduced to specify the structural composition of NL objects from a given dependency structure. The attribute `type="lexical"` indicates that the rule defines lexical mapping, including structural mapping, and the attribute `type="structural"` indicates that the rule defines structural mapping that has no association with a specific word.

The following two examples are lexical and structural mapping rules:

```
<map type="lexical">
   <concept>*insert</concept>
   <WORD cat="v" subcat="trans">insert</WORD>
   <!-- a class *insert is associated with
        a transitive verb ‘‘insert’’ -->
   <actor/>
   <SUBJECT/>
   <!-- actor filler corresponds to SUBJECT -->
   <theme/>
   <DOBJECT/>
   <!-- theme filler corresponds to DOBJECT -->
   <goal/>
   <PPADJUNCT><PREP>into</PREP></PPADJUNCT>
   <!-- goal filler corresponds to a PPADJUNCT with
        a preposition ‘‘into’’ -->
< /map>
 <map type="structural">
   <concept>*physical-action</concept>
   <WORD/>
   <!-- *physical-action has no association with
        a specific word -->
   <mood/>
   <MOOD/>
   <!-- mood filler corresponds to MOOD -->
   <time/>
   <TENSE/>
   <!-- time filler corresponds to TENSE -->
</map>
```

The first rule states lexical mapping between a syntactic word, or a node in dependency structure, and an NL object. The bodies of the first and second rules define structural mapping (i.e., the mapping between a subtree in the dependency structure and a semantic slot filler). The first rule defines that an instance of `*insert` is created to represent the meaning of the transitive verb "insert." Each of its semantic slot fillers is associated with a specific syntactic filler of the verb.

Structural mapping in the first type of rule is applicable only to a specific NL object that is mapped from a syntactic word, while the structural mapping for an NL class defined in the second type of rule is inherited by any of its subclasses.

4.2.1.9 Delayed Composition of NL Objects

During the sentence analysis process, the dependency structure is the primary structure to be built. Mapping from a dependency structure to NL objects is not applied immediately. Rather, it is stored within nodes of the dependency structure as semantic constraints, which makes it possible to pack lexical and structural ambiguities into one (or two) dependency structure(s) while keeping track of all possible semantic interpretations.

The disambiguation technique (described later) will determine the most probable interpretation. Once the dependency structure becomes unambiguous, the structural mapping is evaluated to obtain a full NL-expression that represents the meaning of the sentence. This evaluation process is fairly straightforward.

4.2.1.10 Disambiguation as Constraint Satisfaction

Constraints for disambiguation include syntactic and semantic constraints. Syntactic constraints can be considered as being declarative representations of syntactic rules. *Constraint dependency grammar* (CDG) formalizes parsing as a constraint satisfaction problem [19]. Our syntactic constraints are equivalent to CDG's syntactic rules. Semantic constraints are formed by the consistency between the semantic categories of case-slots and their fillers, which are called selectional restrictions.

From these constraints, a constraint network [20] is generated. Its nodes correspond to ambiguities that are represented as sets of candidates of slot-filler relationships. Its arcs correspond to two-dimensional constraint matrices that represent constraints on combinations of candidates. Each value in a constraint matrix takes 1 (when a combination satisfies the constraint) or 0 (when a combination does not satisfy the constraint). When the constraint network is generated, an algorithm for constraint propagation (filtering out inconsistent candidates) is applied to resolve any ambiguities [21]. I have developed a modified algorithm for constraint propagation that is controlled by preference orderings.

4.2.1.11 Disambiguation Algorithm

A brief overview of the disambiguation algorithm is as follows:

1. If all combinations of the most preferable candidates satisfy the constraints, then choose them and terminate. Otherwise, go to the next step.

2. Remove the least preferable candidate. Then perform constraint propagation.
3. If all candidate solutions of an ambiguity have been removed, undo the last propagation (restore the states that were modified in step 2).
4. If all ambiguities are singletons, then terminate.
5. Repeat steps 2 and 3 for the next least preferable candidate.

A detailed algorithm of constraint propagation under the control of preference orderings is as follows: Let A_i and A_j be ambiguities, a and b be each element respectively, $M(i, j)$ be a constraint matrix on A_i and A_j, and $M(i, a, j, b)$ be the value of the matrix when A_i takes a and A_j takes b. In addition, let $p(i, a)$ be a's preference value in ambiguity A_i. First, for all pairs of an ambiguity and its element (i, a), construct an ordering set PRF that is arranged (i, a) in ascending order of preference. Thus, for each element (i, a) and (j, b) of PRF, if (i, a) is ahead of (j, b), then $p(i, a)$ is equal to or less than $p(j, b)$.

The algorithm consists of the following steps:

1. Remove the front-most element (i, a) from PRF.
2. Using the following algorithm, for all ambiguities and their elements, construct their supported sets and the set of inactive elements IN.
   First, for each ambiguity A_i, its element a, and a constraint matrix $M(i, j)$, construct a supported set of (i, a), $S(i, j, a) = \{(j, b) | M(i, a, j, b) = 1$ (in this case, we say that (j, b) supports $(i, a))\}$. If $S(i, j, a) = \{\}$, then put (i, a) in IN, a set of inactive elements. Also, for each A_j and its element b, construct a supported set of (j, b), $S(j, i, b)$. If $S(j, i, b) = \{\}$, then put (j, b) in IN.
   Set the set CHG (the set containing the changed elements) to $\{\}$.
3. For each of the ambiguities A_j and A_k and their elements b and c, respectively, if $M(i, a, j, b) = 1$ or $M(k, c, i, a) = 1$, then set their values to 0 and put (i, a, j, b) or (k, c, i, a) in CHG.
4. Iterate the following substeps until IN becomes empty:
   - Remove element (j, b) from IN.
   - Set A_j to $A_j - \{b\}$.
   - If $A_j = \{\}$ then set A_j to $\{b\}$, for all elements (k, c, l, d) of CHG, set $M(k, c, l, d)$ to 1, then go to step 1.
   - For all elements (k, c) of $S(j, b)$, set $M(j, b, k, c)$ to 0 and $S(k, j, c)$ to $S(k, j, c) - \{(j, b)\}$, then put (j, b, k, c) in CHG.
   - If $S(k, j, c) = \{\}$ and (k, c) is not an element of IN, then put (k, c) in IN.
5. If all ambiguities are singletons, terminate. Otherwise, go to step 1.

The problem that this algorithm can solve is called the *consistent labeling problem* (CLP) [20]. CLP is a problem that determines the existence of an assignment that satisfies all the constraints, given a set of variables each of which can take any one of a set of values and constraints between these values. An example of CLP is a graph-coloring problem to assign colors to a graph with all adjacent vertexes in different colors. A CLP is *satisfiable* if there is an assignment that satisfies all the constraints simultaneously. Deciding the satisfiability of a CLP is NP complete in general. There are cases, however, in which unsatisfactory values (i.e., values that are not included in any solution) can be identified simply by constraint propagation that checks local inconsistencies. Such an algorithm has been shown to have polynomial complexity [21].

It is possible to achieve more global consistency by looking at multiple constraint matrices simultaneously, but once local (pair-wise) consistencies have been achieved, performing a backtrack search is usually more efficient than using higher-level consistency algorithms. Our algorithm combines constraint propagation with the preferential control rather than performing a backtrack search. I extended a constraint propagation algorithm in two senses. First, I added a control mechanism based on preferential orderings over candidate solutions. Second, I added an inconsistency checking and undoing process when local inconsistencies occur after constraint propagation. These mechanisms are invoked when the given constraints are not tight enough to narrow down the number of candidate solutions to one. If a CLP can have a solution, then our algorithm can find a solution, since the algorithm maintains local consistency (locally consistent value sets must contain a solution) and narrows down the number of candidates. However, since our preference compares candidates of each value set (i.e., ambiguity) independently, our algorithm may not find the optimum solution that is prior to all other solutions.

4.2.2 Speech Analysis

Speech recognition is the process of converting an acoustic signal, captured by a microphone, or a recorded audio data, to a set of words. The recognized words can be the final results, as for applications such as input commands, data entry, and document preparation. They can also serve as the input to further linguistic processing in order to achieve spoken language understanding.

Speech recognition systems can be characterized by many parameters. An isolated-word speech recognition system requires that the speaker pause

briefly between words, whereas a continuous speech recognition system does not. Spontaneous, or extemporaneously generated, speech contains disfluencies (e.g., stalls, hesitations, and self-repairs), and is much more difficult to recognize than speech read from script. Some systems require speaker enrollment—a user must provide samples of his or her speech before using them, whereas other systems are said to be speaker-independent, in that no enrollment is necessary. Some of the other parameters depend on the specific task. Recognition is generally more difficult when vocabularies are large or have many similar-sounding words. When speech is produced in a sequence of words, language models or grammars are used to restrict the combination of words.

The simplest language model can be specified as a finite-state network, where the permissible words following each word are given explicitly. More general language models approximating natural language are specified in terms of a context-sensitive grammar.

One popular measure of the difficulty of the task, combining the vocabulary size and the language model, is perplexity, loosely defined as the geometric mean of the number of words that can follow a word after the language model has been applied. Finally, there are some external parameters that can affect speech recognition system performance, including the characteristics of the environmental noise and the type and the placement of the microphone.

The linguistic elements handled by speech recognition are phonemes—a smallest distinguishable unit of speech. Figure 4.3 shows a typical process of speech analysis from a speech input to an output of a sequence of words.

An acoustic analysis module extracts a set of useful features in digitized speech signals at a fixed rate, typically once every 10 to 20 milliseconds. Then, a speech recognition module receives vector quantization codes that are compressed signal feature representations based on the result of acoustic analysis. The module identifies phonemes and assembles them into words using lexical and grammatical constraints.

Speech recognition is a difficult problem, largely because of the many sources of variability associated with the signal. First, the acoustic realizations of phonemes are highly dependent on the context in which they appear. Second, acoustic variabilities can result from changes in the environment as well as in the position and characteristics of the speech recognizer. Third, within-speaker variabilities can result from changes in the speaker's physical and emotional state, speaking rate, or voice quality. Finally, differences in sociolinguistic background, dialect, and vocal tract size and shape can contribute to across-speaker variabilities.

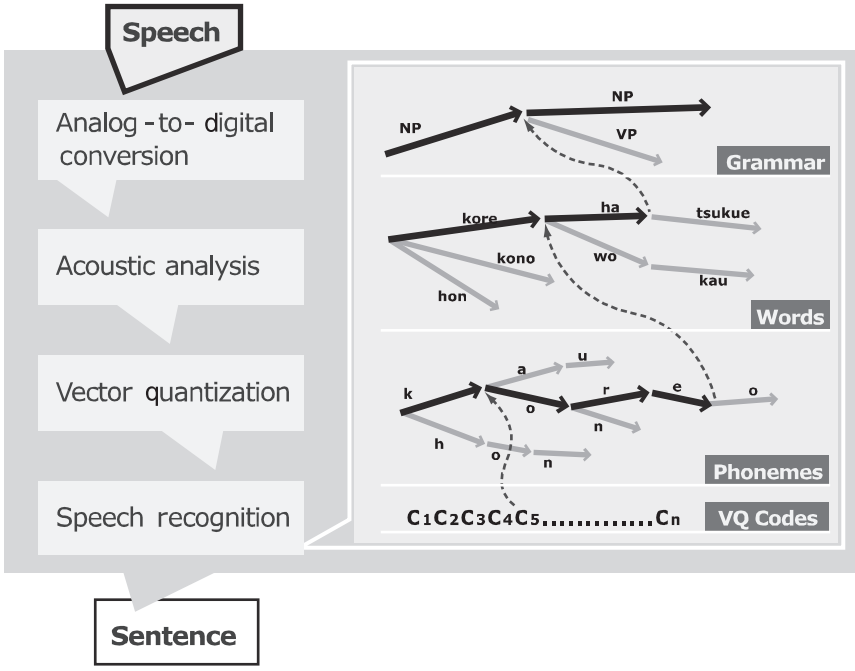


Figure 4.3 Process of speech analysis.

The dominant recognition paradigm in the past 15 years is known as the *hidden Markov model* (HMM) [22]. An HMM is a doubly stochastic model, in which the generation of the underlying phoneme string and the frame-by-frame, surface acoustic realizations are both represented probabilistically as Markov processes. Neural networks have also been used to estimate the frame based scores; these scores are then integrated into HMM-based system architectures, in what has come to be known as hybrid systems.

Integrating speech and natural language processing has been pursued to resolve several tough problems in understanding spoken language, such as treatment of ambiguous or ill-formed speech, prediction of subsequent utterances, revision of previous interpretation, and some speech-specific issues including phoneme insertion/deletion and word boundary recognition.

Some research has managed to integrate linguistic information into speech understanding, such as that by Erman et al. [23], Hayes et al. [24], Young et al. [25], and Chung et al. [26]. The methods they propose, however, are specific combinations of specific types of knowledge sources

with rather fixed information flows, and with flexible interactions disallowed among them.

While most conventional spoken language systems first construct phonological hypotheses, then apply higher level knowledge and generate semantic interpretations, my group has proposed a system that simultaneously applies all types of constraints, from acoustic to pragmatic [27]. Our method can flexibly control omnidirectional information flow in a very context-sensitive fashion.

A major advantage of our method over the previous ones is that it implements more fine-grained and diverse feedback loops integrating heterogeneous sorts of information, rather than constructing phoneme-level hypotheses in advance and then going on to the linguistic level as in most of the other approaches. Our system can, for instance, pick up an alleged phoneme depending upon linguistic or extralinguistic context, even if that phoneme looks so implausible on the acoustic ground alone that traditional bottom-up architecture would filter it out before entering the linguistic mode. Note that such diverse information flow does not cause computational intractability, because information processsing at each context is centered around dynamically determined focal points or the focus of attention in the network of constraints. This control mechanism gives rise to spoken language understanding without any specific procedures for specific tasks, such as matching between phonemes, parsing, and semantic and pragmatic inferences. The details of the algorithm are beyond the scope of this book.

4.2.3 Video Analysis

Video analysis deals with the images in the video stream. It is primarily used for the identification of scene breaks and to select static frame images that are representative of a scene. Primitive image features based on image statistics, such as color histograms, are used for indexing, matching, and segmenting images.

Figure 4.4 shows that several video characterizations are involved in video analysis [28]. Scene change detection is based on comparative difference of color histogram of consecutive video frames. Camera motion analysis uses optical flow vectors and detects a pan or zoom. Face and text detection recognizes human faces and text captions in video frames. More details of techniques to detect these characterizations are described below.

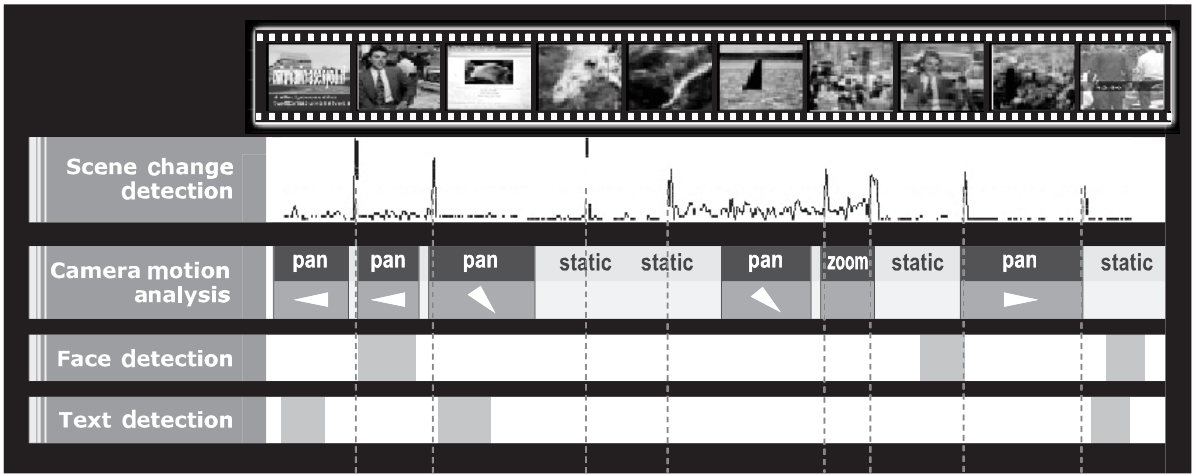


Figure 4.4 Video content characterization. (*From:* [28]. © 1995 Carnegie Mellon University.)

4.2.3.1 Scene Change Detection

To analyze each segment as an individual scene, the video analysis system must first identify frames where scene changes occur. Several techniques have been developed for detecting scene breaks.

Using color histogram analysis, video is segmented into scenes through the use of comparative difference measures [29]. Images with small histogram disparity are considered to be relatively similar. By detecting significant changes in the weighted color histogram of successive frames, image sequences can be separated into individual scenes. A comparison between cumulative distributions is used as a difference measure. This result is passed through a highpass filter to further isolate peaks, and an empirical threshold is used to select only those regions where scene breaks occur.

4.2.3.2 Camera Motion Analysis

One important method of video characterization is based on interpreting camera motion.

Optical flow (the direction and the speed of motion of the features in the image) analysis is an important method of visual segmentation and description based on interpreting camera motion. The video analysis system can identify camera motion as a pan or zoom by examining the geometric properties of the optical flow vectors. Using the Lucas-Kanade gradient descent method for measuring optical flow, the system can track individual regions from one frame to the next [30]. By measuring the velocity of individual regions over time, a motion representation of the scene is created. Sudden, widespread changes in this flow suggest random motion, and therefore, new scenes. Optical flow changes also occur during gradual transitions between images such as fades or special effects.

4.2.3.3 Object Detection

The video analysis system can identify significant objects by searching and matching known templates to individual regions in the frame. In most cases, the system deals with two of the more interesting objects in video, human faces and subtitle texts.

4.2.3.4 Face Detection

The talking human head image is common in interviews and news clips, and illustrates a clear example of video production focusing on an individual of interest. A human interacting within an environment is also a common theme in video. A human-face detection system was developed by researchers at Carnegie Mellon University [31]. Its current performance level is to detect

over 90% of more than 300 faces contained in 70 images, while producing approximately 60 false detections. While much improvement is needed, the system can detect faces of varying sizes and is especially reliable with frontal faces such as talking head images.

4.2.3.5 Text Detection

Text in the video provides significant information as to the content of a scene [32]. For example, statistical numbers are not usually spoken but are included in the captions for viewer inspection. Names and titles are attached to close-ups of people. A text region is a horizontal (or vertical in some cases of Japanese) rectangular structure of clustered sharp edges, due to characters with high contrast color or intensity, against the background. By detecting these properties, the video analysis system can extract regions from video frames that contain textual information. The system first extracts vertical edge features. Then smoothing is applied to eliminate extraneous fragments and to connect edge elements that may have been detached. Individual regions are identified by cluster detection and bounding rectangles are computed.

A cluster's bounding region must have a small vertical-to-horizontal aspect ratio and must also satisfy various limits in height and width. The fill factor of the region should be high to insure dense clusters. The cluster size should also be relatively large to avoid small fragments. Finally, the intensity histogram of each region is tested for high contrast. This is because certain textures and shapes are similar to text but exhibit low contrast when examined in a bounded region. This method works best with horizontal titles and captions.

As presented above, there are many different techniques to analyze semantics of digital content. These techniques are used in creation of semantic annotations. The main purpose of the rest of this chapter is to introduce the idea of semantic annotation in detail, and then demonstrate a number of useful applications that use these annotations with a transcoding proxy. The particular examples presented are text summarization, language translation, text paraphrasing, and multimedia transcoding.

4.3 Semantic Annotation

Since technologies of automatic analysis of content are not perfect, human support is needed for machines to understand semantics of content. Semantic annotations created by human-machine collaboration are helpful hints to

assist the machine's understanding. Such annotations do not just increase the expressive power of content but also play an important role in content reuse. An example of content reuse is, for example, transcoding of content depending on user preferences.

Content adaptation as presented in Chapter 2 is a type of transcoding that considers a user's environment—devices, network bandwidth, profiles, etc. In addition, such adaptation sometimes requires a good understanding of the original content. If the transcoder fails to analyze the semantic structure of content, then the transcoding results may not be accurate and may cause user misunderstanding.

Our technology assumes that semantic annotations help machines to understand content so that transcoding can have higher quality. I call such transcoding based on semantic annotation *semantic transcoding*. The overall configuration of the semantic transcoding system can be viewed in Figure 4.5.

My group has developed a simple method for associating semantic annotations with any element or segment of any Web document and multimedia data. The word "semantic" indicates that the annotations are helpful for machines to understand semantics of the original content.

We use URIs (actually URLs), XPointers (location identifiers in the document) [33], and document hash codes (digest values) based on the MD5 Message-Digest Algorithm [34] to identify particular elements in HTML or XML documents. We have also developed an annotation server that maintains the relationship between content and annotations and transfers requested annotations to a transcoding proxy.

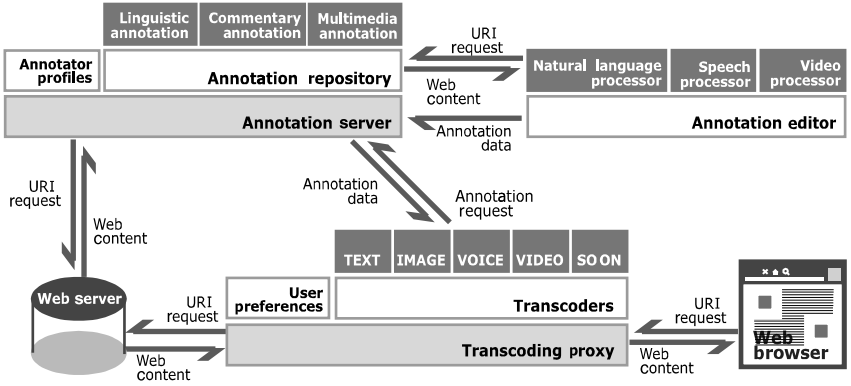


Figure 4.5 Semantic transcoding system.

4.3.1 Annotation Environment

Our annotation environment consists of a client side editor for the creation of annotations and a server for the management of annotations.

The annotation environment is shown in Figure 4.6.

The process flows as follows (in this example case, XML or HTML documents are processed):

1. The user runs the annotation editor and requests a URI as a target of annotation.
2. The annotation server (the server, hereafter) accepts the request and sends it to the Web server.
3. The server receives the Web document.
4. The server calculates the document hash code (digest value) and registers the URI with the code to its database.
5. The server returns the Web document to the editor.
6. The user annotates the requested document and sends the result to the server, along with some encrypted personal data (name, professional areas, and so on).

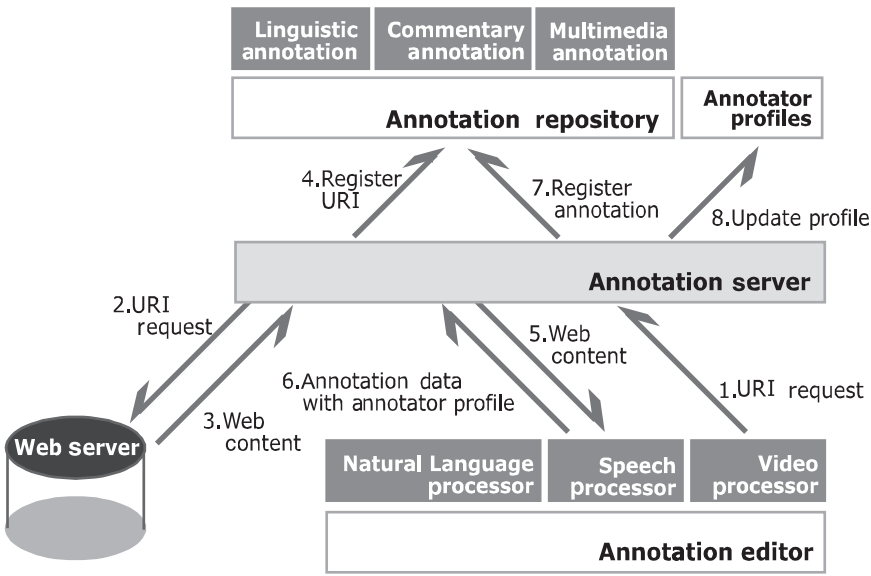


Figure 4.6 Annotation environment.

7. The server receives the annotation data and relates it with its URI in the database.
8. The server also checks a digital signature associated with the annotation data, decrypts the personal data, and updates the annotator profiles.

Below the annotation editor and the server are explained in more detail.

4.3.2 Annotation Editor

Our annotation editor, implemented as a Java application, can communicate with the annotation server, as explained below.

The annotation editor has the following functions:

1. Registering targets of annotation with the annotation server by sending URIs;
2. Specifying elements in the document using a Web browser;
3. Generating and sending annotation data to the annotation server;
4. Reusing previously created annotations when the target content is updated;
5. Associate an annotator's digital signature with his or her created annotation data.

An example screen of our annotation editor is shown in Figure 4.7.

The left top window of the editor shows the document object structure of the HTML document. The right window shows some text that was selected on the Web browser. The selected area is automatically assigned an XPointer. The left bottom window shows a linguistic structure of the sentence in the selected area.

Using the editor, the user annotates text with linguistic structure (grammatical and semantic structure, described later) and adds a comment to an element in the document. The editor is capable of basic natural language processing and interactive disambiguation. The user should modify the results of the automatically analyzed sentence structure as shown in Figure 4.8.

4.3.3 Annotation Server

Our annotation server receives annotation data from an annotator and classifies it according to the annotator's name. The server retrieves documents

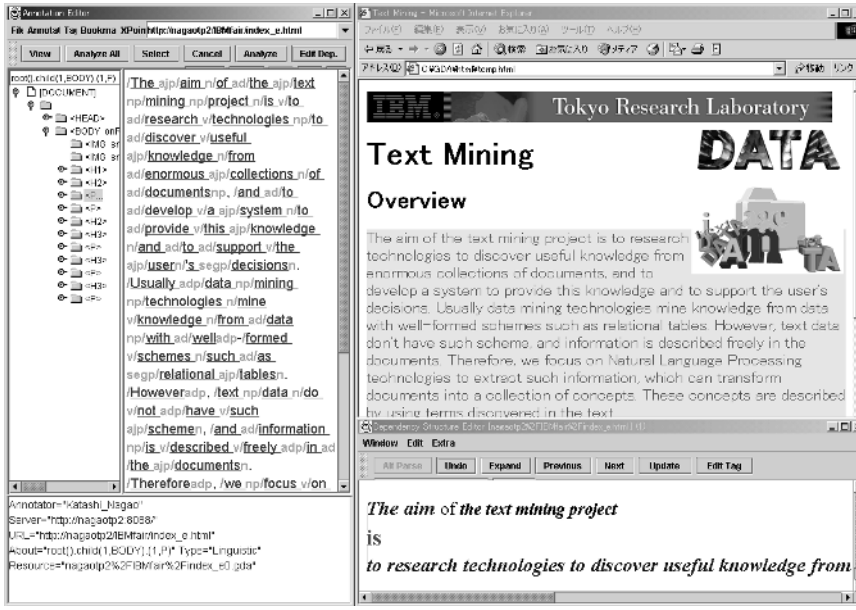


Figure 4.7 Screen of the annotation editor.

from URIs in annotation data and registers the document hash codes with their URIs in its annotation database. Since a document's author may modify the document after the initial retrieval, the hash code of a document's internal structure (i.e., DOM) enables the server to discover modified elements in the annotated document [35].

The annotation server makes a table of annotator names, URIs, XPointers, and document hash codes. When the server accepts a URI as a request from a transcoding proxy (described in Section 4.4.1), the server returns a list of XPointers with associated annotation data, their types (linguistic or commentary), and a hash code. If the server receives an annotator's name as a request, it responds with the set of annotations created by the specified annotator.

In the case of multimedia annotation, XPointers and hash codes cannot be calculated for the target content. Time stamps and data sizes are used instead of XPointers and hash codes, respectively.

My group is currently developing a mechanism for access control between annotation servers and ordinary content servers. If authors of original documents do not want to allow anyone to annotate their documents,

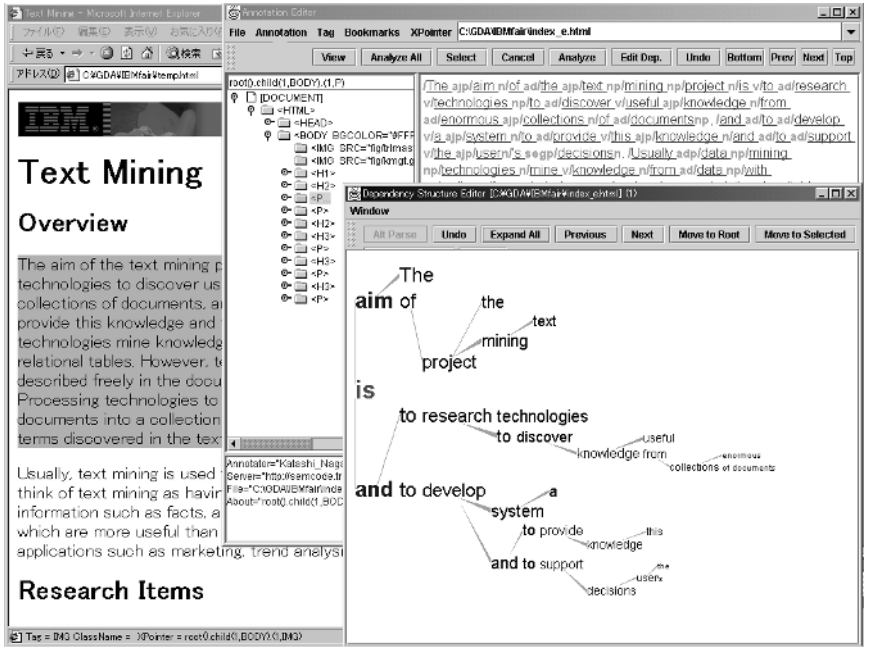


Figure 4.8 Annotation editor with the linguistic structure editor.

they can add a statement about it in the documents, and annotation servers will not retrieve such content for the annotation editors.

4.3.4 Linguistic Annotation

Linguistic annotation has been used to make digital documents machine-understandable, and to develop content-based presentation, retrieval, question-answering, summarization, and translation systems with much higher quality than is currently available.

4.3.4.1 Global Document Annotation

I have employed the GDA tag set as a basic framework to describe linguistic and semantic features of documents [8].

GDA is a challenging project to make digital documents machine-understandable on the basis of a new tag set, and to develop content-based presentation, retrieval, question-answering, summarization, and translation systems with much higher quality than before. GDA thus proposes an

integrated global platform for digital content authoring and annotation, presentation, and reuse.

The GDA tag set is based on XML and is designed to be as compatible as possible with *Text Encoding Initiative* (TEI) [36], *Corpus Encoding Standard* (CES) [37], and *Expert Advisory Group on Language Engineering Standards* (EAGLES) [38]. These projects aim at development standards that help libraries, museums, publishers, and individual scholars represent all kinds of literary and linguistic resources for on-line research and teaching, using encoding schemes that are maximally expressive and minimally obsolescent. The GDA specifies an encoding of modifier-modifiee relations, anaphor-referent relations, and word senses.

An example of a GDA-tagged sentence is as follows:

```
<su><np opr="agt" sem="time0">Time</np>
<v sem="fly1">flies</v>
<adp opr="eg"><ad sem="like0">like</ad>
<np>an <n sem="arrow0">arrow</n></np>
</adp>.</su>
```

The `<su>` element is a sentential unit. The other tags above, `<n>`, `<np>`, `<v>`, `<ad>` and `<adp>`, mean noun, noun phrase, verb, adnoun or adverb (including preposition and postposition), and adnominal or adverbial phrase, respectively.

The `opr` attribute encodes a relationship in which the current element stands with respect to the element that it semantically depends on. Its value denotes a binary relation, which may be a thematic role such as agent (actor), patient, recipient, or a rhetorical relation such as cause or concession. For instance, in the above sentence, `<np opr="agt" sem="time0">Time</np>` depends on the second element `<v sem="fly1">flies</v>`. `opr="agt"` means that *Time* has the agent role with respect to the event denoted by *flies*. The `sem` attribute encodes a word sense.

The GDA initiative aims at having many Web authors annotate their on-line documents with this common tag set so that machines can automatically recognize the underlying semantic and pragmatic structures of those documents much more easily than by analyzing traditional HTML documents. A huge amount of annotated data is expected to emerge, which should serve not just as tagged linguistic corpora but also as a worldwide, self-extending knowledge base, mainly consisting of examples showing how our knowledge is manifested.

GDA has three main steps:

1. Propose an XML tag set which allows machines to automatically infer the underlying structure of documents.
2. Promote development and spread of natural language processing and artificial intelligence applications to turn tagged texts to versatile and intelligent content.
3. Motivate the authors of Web documents to annotate their documents using the proposed tags.

The tags proposed in step 1 will also encode coreferences, rhetorical structure, and the social relationship between the author and the audience, in order to render the document machine-understandable.

Step 2 concerns AI applications such as machine translation, information retrieval, information filtering, data mining, consultation, and expert systems. If annotation with such tags as mentioned above may be assumed, it is certainly possible to drastically improve the accuracy of such applications. New types of applications for communication aids may be invented as well.

Step 3 encourages Web authors to present themselves to the widest and best possible audience by organized tagging. Web authors will be motivated to annotate their Web documents because documents annotated according to a common standard can be translated and retrieved with higher accuracy, and thus have a greater chance to reach more targeted readers. Thus, tagging will make documents stand out much more effectively than decorating them with pictures and sounds.

4.3.4.2 Thematic/Rhetorical Relations

A part of the `opr` attribute values is called a relational term. A relational term denotes a binary relation, which may be a thematic role such as agent, patient, or recipient, or a rhetorical relation such as cause or concession. Thus, we conflate thematic roles and rhetorical relations here, because the distinction between them is often vague. For instance, ''concession'' may be both intrasentential (e.g., ''I like this but not that.'') and intersentential (e.g., ''I like this. But you don't.'') relation.

Here is an example of a `opr` attribute:

```
<su><namep opr="agt">Tom</namep>
<v>came</v>.</su>
```

In this sentence, the first element `<namep opr="agt">Tom</namep>` depends on the second element `<v>came</v>`. `opr="agt"` means that Tom has the agent role with respect to the event denoted by *came*.

`The opr` is an open-class attribute, potentially encompassing all the binary relations lexicalized in natural languages. An exhaustive listing of thematic roles and rhetorical relations appears impossible, as widely recognized. The researchers of GDA are not yet sure about how many thematic roles and rhetorical relations are sufficient for engineering applications. However, the appropriate granularity of classification will be determined by the current level of technology.

4.3.4.3 Anaphora and Coreference

Each element may have an identifier as the value of the `id` attribute. Anaphoric and coreferential expressions should have the `eq` attribute with their antecedents' `id` values.

An example follows:

```
<namep id="j1">John</namep> beats
<ajp eq="j1">his</ajp> dog.
```

When the coreference is at the level of type (kind, sort, and so on), which the referents of the antecedent and the anaphor are tokens of, we use the `ctp` attribute as next:

```
You bought <np id="c1">a car</np>.
I bought <np ctp="c1">one</np>, too.
```

A zero anaphora is encoded by using the appropriate relational term as an attribute name with the referent's `id` value. Zero anaphors of compulsory elements, which describe the internal structure of the events represented by the verbs or adjectives, are required to be resolved. Zero anaphors of optional elements such as with reason and means roles may not. Here is an example of a zero anaphora concerning an optional thematic role `ben` (for *beneficiary*):

```
Tom visited <namep id="m1">Mary</namep>.
He <v ben="m1">brought</v> a present.
```

Linguistic annotation is generated by automatic morphological analysis, interactive sentence parsing, and word sense disambiguation by selecting the most appropriate item in the domain ontology. Some research issues on linguistic annotation are related to how the annotation cost can be reduced within some feasible levels. We have been developing some machine-guided annotation interfaces to simplify the annotation work. Machine

learning mechanisms also contribute to reducing the cost because they can gradually increase the accuracy of automatic annotation.

In principle, the tag set does not depend on language, but as a first step our group implemented a semi-automatic tagging system for English and Japanese.

Linguistic annotation covers syntactic and semantic annotations. Semantic annotation is divided into word sense and logical annotations. Word sense annotation is based on domain-specific ontologies, while logical annotation is related to formal semantics and inference.

4.3.4.4 Word Sense Annotation

In the computational linguistic field, word sense disambiguation has been one of the biggest issues. For example, to have a better translation of documents, disambiguation of certain polysemic words is essential. Even if an estimation of the word sense is achieved to some extent, incorrect interpretation of certain words can lead to irreparable misunderstanding.

To avoid this problem, we have been promoting annotation of word sense for polysemic words in the document, for example using WordNet [39], so that their word senses can be machine-understandable.

For this purpose, we need a dictionary of concepts, for which we use existing domain ontologies. An ontology is a set of descriptions of concepts—such as things, events, and relations—that are specified in some way (such as specific natural language) in order to create an agreed-upon vocabulary for exchanging information.

Annotating a word sense is therefore equal to creating a link between a word in the document and a concept in a certain domain ontology. We have made a word sense annotating tool for this purpose, which has been integrated with the annotation editor.

As mentioned, using the editor, the user annotates text with linguistic structure (syntactic and semantic structure) and adds a comment to an element in the document. The editor is also capable of word sense annotation as shown in Figure 4.9. The ontology viewer appears in the right middle of the figure. The user can easily select a concept in the domain ontology and assign a concept ID to a word in the document as a word sense.

4.3.5 Commentary Annotation

Commentary annotation is mainly used to annotate nontextual elements like images and sounds with some additional information. Each comment can

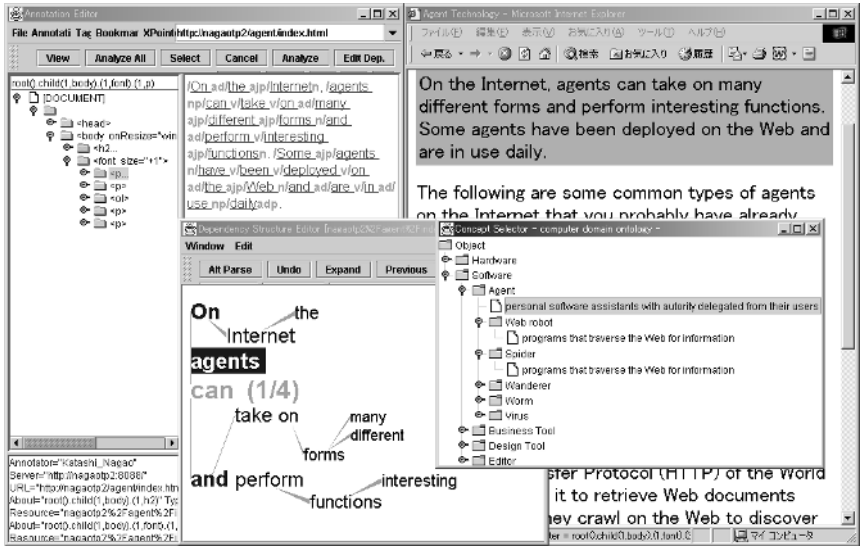


Figure 4.9 Annotation editor with the ontology viewer.

include not only tagged texts but also other images and links. Currently, this type of annotation appears in a subwindow that is overlayed on the original document window when a user positions a mouse pointer at the area of a comment-added element as shown in Figure 4.10.

Users can also annotate text elements with information such as paraphrases, correctly spelled words, and underlines. This type of annotation is used for text transcoding that combines such comments on texts and original texts.

Commentary annotaion on hyperlinks is also available. This contributes to quick introduction of target documents before clicking the links. If there are linguistic annotations on the target documents, the transcoders can generate summaries of these documents and relate them with hyperlinks in the source document.

Some research has been published concerning sharing comments over the Web. Annotea, mentioned earlier in Section 3.2.1, is a general metainformation architecture for annotating documents on the Web [40]. This architecture includes a basic client-server protocol, general metainformation description language (i.e., resource description framework [3]), a server system, and an authoring tool and browser with interface augmentations to provide access to its extended functionality. Annotea provides a general mechanism for shared annotations, which enables people to annotate

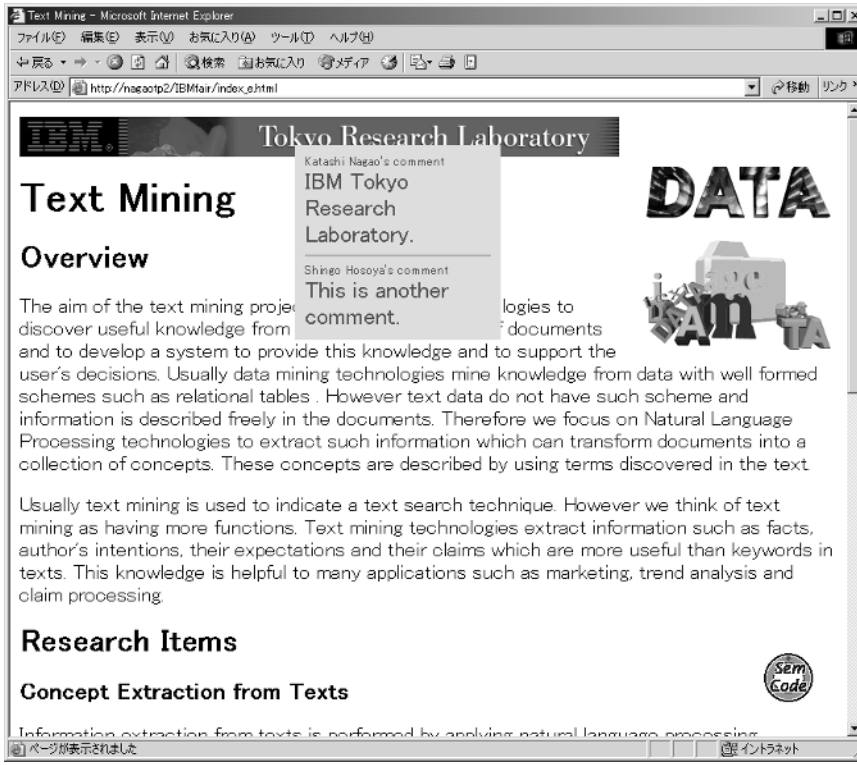


Figure 4.10 Comment overlay on the document.

arbitrary documents at any position in-place and share comments/pointers with other people. There are several annotation systems with a similar direction, such as *ComMentor* [41] and the *Group Annotation Transducer* [42].

These systems are often limited to particular documents or documents shared only among a few people. Our annotation and transcoding system can also handle multiple comments on any element of any document on the Web. Also, a community-wide access control mechanism can be added to our transcoding proxy. If a user is not a member of a particular group, then the user cannot access the transcoding proxy that is for group use only. In the future, transcoding proxies and annotation servers will communicate with some secured protocol that prevents some other server or proxy from accessing the annotation data.

My main focus is adaptation of on-line content to users, and sharing comments in a community is one of our additional features. I have been

applying both commentary and linguistic annotations to semantic transcoding.

Commentary annotations are also used in knowledge sharing and reuse in communities of several domain experts. This topic is discussed later in more detail.

4.3.6 Multimedia Annotation

Multimedia content such as digital video is becoming a prevalent information source. Since the volume of such content is growing to huge numbers of hours, summarization is required to effectively browse video segments in a short time without missing significant content. Annotating multimedia content with semantic information such as scene/segment structures and metadata about visual/auditory objects is necessary for advanced multimedia content services. Since natural language text such as a voice transcript is highly manageable, speech and natural language processing techniques have an essential role in our multimedia annotation.

My group has developed techniques for semi-automatic video annotation, integrating a multilingual voice transcription method, some video analysis methods, and an interactive visual/auditory annotation method [43]. The video analysis methods include automatic color change detection, characterization of frames, and scene recognition using similarity between frame attributes.

There are related approaches to video annotation. For example, MPEG-7 can describe indecis, notes, and so on, to retrieve necessary parts of content speedily. However, it takes a high cost to add these descriptions manually. The method of extracting them automatically through the video/audio analysis is vitally important. Our method can be integrated into tools for authoring MPEG-7 data. The linguistic description scheme, which will be a part of the amendment to MPEG-7, should play a major role in this integration.

Using such annotation data, we have also developed a system for advanced multimedia processing such as video summarization and translation. Our video summary is not just a shorter version of the original video clip, but an interactive multimedia presentation that shows keyframes of important scenes and their transcripts in Web documents and allows users to interactively modify summary. The video summarization is customizable according to users' favorite length and keywords. When a user's client device is not capable of video playing, our system tranforms video to a document that is the same as a Web document in HTML format.

Multimedia annotation can make delivery of multimedia content to different devices very effective. Dissemination of multimedia content will be facilitated by annotation on the usage of the content for different purposes, client devices, and so forth. Also, it provides object-level description of multimedia content, which allows a higher granularity of retrieval and presentation in which individual regions, segments, objects, and events in image, audio, and video data can be differentially accessed depending on publisher and user preferences, network bandwidth, and client capabilities.

Multimedia annotation is an extension of document annotation such as GDA. Since natural language text is more tractable and meaningful than binary data of visual (image and moving picture) and auditory (sound and voice) content, we associate text with multimedia content in several ways. Since most video clips contain spoken narrations, our system converts them into text and integrates them into video annotation data. The text is sometimes acquired from closed captions on television programs. The text in the multimedia annotation is linguistically annotated based on GDA.

4.3.7 Multimedia Annotation Editor

My group has developed an authoring tool called the Multimedia Annotation Editor capable of video scene change detection, multilingual voice transcription, syntactic and semantic analysis of transcripts, and correlation of visual/auditory segments and text.

An example screen of the editor is shown in Figure 4.11. The editor screen consists of three windows. One window (top) shows the video content, automatically detected keyframes in the video, and an automatically generated voice transcript. The second window (left bottom) enables the user to edit the transcript and modify an automatically analyzed linguistic markup structure. The third window (right bottom) shows graphically a linguistic structure of the selected sentence in the second window.

The editor is capable of basic natural language processing and interactive disambiguation. The user can modify the results of the automatically analyzed multimedia and linguistic (syntactic and semantic) structures.

4.3.7.1 Video Annotation

The linguistic annotation technique has an important role in multimedia annotation. Our video annotation consists of creation of text data related to video content, linguistic annotation of the text data, automatic segmentation

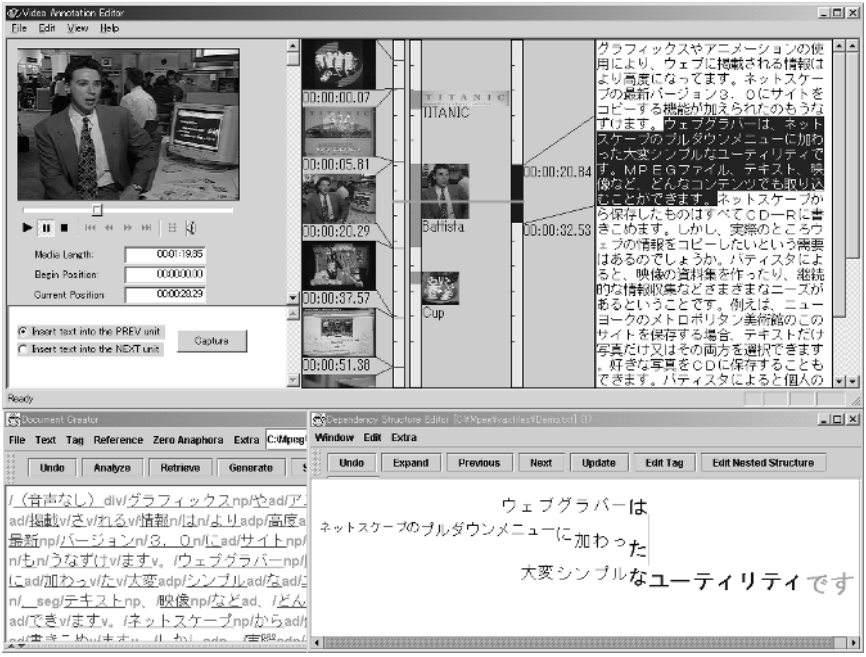


Figure 4.11 Multimedia annotation editor.

of video, semi-automatic linking of video segments with corresponding text data, and interactive naming of people and objects in video scenes.

To be more precise, video annotation is performed through the following three steps:

1. For each video clip, the annotation system creates the text corresponding to its content. We developed a method for creation of voice transcripts using speech recognition engines. It is called multilingual voice transcription and is described later.
2. Some video analysis techniques are applied to characterization of visual segments (i.e., scenes) and individual video frames. For example, by detecting significant changes in the color histogram of successive frames, frame sequences can be separated into scenes.

   Also, by matching prepared templates to individual regions in the frame, the annotation system identifies objects. The user can specify significant objects in some scene in order to reduce the time to identify target objects and to obtain a higher recognition accuracy.

The user can name objects in a frame simply by selecting words in the corresponding text.

3. The user relates video segments to text segments such as paragraphs, sentences, and phrases, based on scene structures and object-name correspondences. The system helps the user select appropriate segments by prioritizing them based on the number of the detected objects, camera motion, and the representative frames.

4.3.7.2 Multilingual Voice Transcription

The Multimedia Annotation Editor first extracts the audio data from a target video clip. Then, the extracted audio data is divided into left and right channels. If the average for the difference of the audio signals of the two channels exceeds a certain threshold, they are considerd different and transfered to the multilingual speech identification and recognition module. The output of the module is a structured transcript containing time codes, word sequences, and language information. It is described in XML format as shown here:

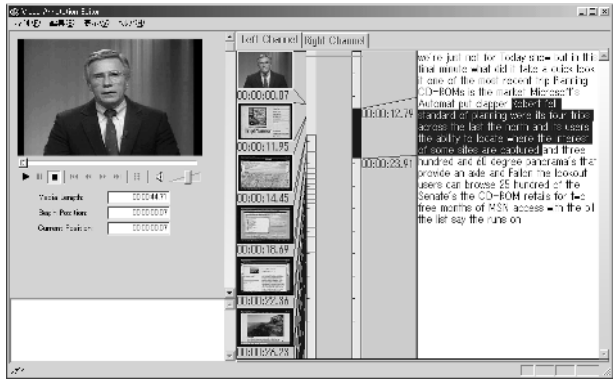
```
<transcript lang="en"channel="l">
   <w in="20.264000"out="20.663000">Web grabber </w>
   <w in="20.663000"out="21.072000">is a </w>
   <w in="21.072000"out="21.611000">very simple </w>
   <w in="21.611000"out="22.180000">utility </w>
   <w in="22.180000"out="22.778000">that is </w>
   <w in="22.778000"out="23.856000">attached to </w>
   <w in="23.856000"out="24.215000">Netscape </w>
   <w in="24.215000"out="24.934000">as a pull down menu </w>
   <w in="24.934000"out="25.153000">and </w>
   <w in="25.153000"out="25.462000">allows you </w>
   <w in="25.462000"out="25.802000">to take </w>
   <w in="25.802000"out="26.191000">your Web content </w>
   <w in="26.191000"out="27.039000">whether it's a </w>
   <w in="27.039000"out="27.538000">MPEG file </w>
   ...
</transcript>
```

Our Multilingual Voice Transcriptor automatically generates transcripts with time codes and provides their reusable data structure, which

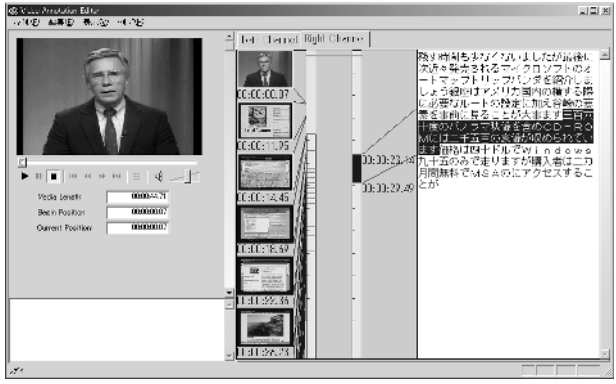
allows easy manual correction. An example screen of the Mulitilingual Voice Transcriptor is shown in Figure 4.12. The example video clip has two audio channels. One channel is for English and other for Japanese. The Mulitilingual Voice Transcriptor automatically identifies the language of each channel. The top screen in the figure shows the transcript in English and the bottom shows that in Japanese.

4.3.7.3 Multilingual Speech Identification and Recognition

The progress of speech recognition technology makes it comparatively easy to transform speech into text, but spoken language identification is needed for processing multilingual speech, because speech recognition technology



Left channel



Right channel

Figure 4.12 Multilingual voice transcriptor.

assumes that the language used is known. While researchers have been working on multilingual speech identification, few applications based on this technology have been actually used except a telephony speech translation system. In the case of the telephone translation system, the information of the language used is self-evident—at least, the speaker knows—so there are little needs and advantages of developing a multilingual speech identification system.

On the other hand, speech data in video does not always have the information about the language used. Due to the recent progress of digital broadcasting and signal compression technology, information about the language is expected to accompany the content in the future, but most of the data available now do not have it, so a large amount of labor is needed to identify the language. Therefore, multilingual speech identification has a large part to play with unknown-language speech input.

A process of multilingual speech identification is shown in Figure 4.13. Our method determines the language of input speech using a simple discriminant function based on relative scores obtained from multiple speech recognizers working in parallel.

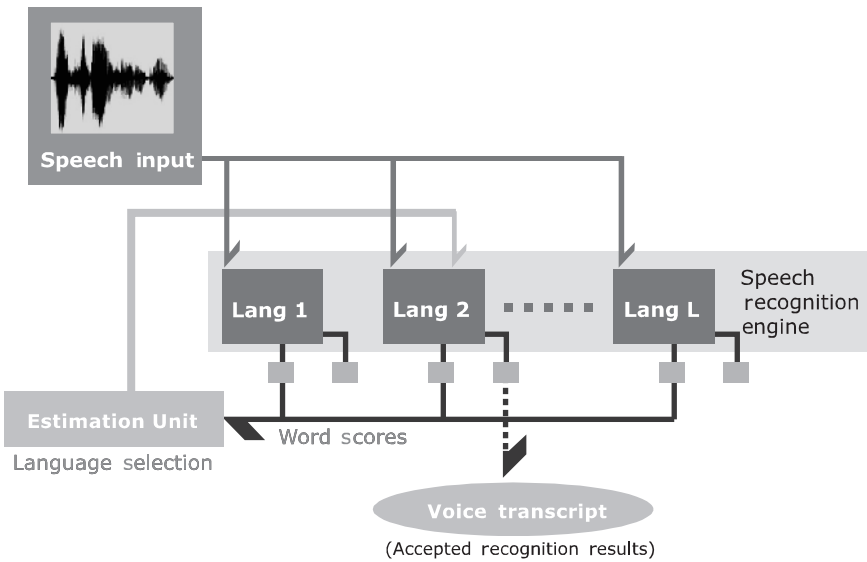


Figure 4.13 Configuration of spoken language identification unit.

Multiple speech recognition engines work simultaneously on the input speech. It is assumed that each speech recognition engine has the speaker independent model, and each recognition output word has a score within a constant range dependent on each engine.

When a speech comes, each recognition engine outputs a word sequence with scores. The estimation unit calculates a value of a discriminant function using the scores for every language. The engine with the highest average discriminant value is selected and the language is determined by the engine, whose recognition result is accepted as the transcript. If there is no distinct difference between discriminant values, that is, not higher than a certain threshold, a judgment is entrusted to the user.

Our technique is simple, it uses the existing speech recognition engines tuned in each language without a special model for language identification and acoustic features.

Combining voice transcription and video image analysis, our tool enables users to create and edit video annotation data semi-automatically. The entire process is as shown in Figure 4.14.

Our system drastically reduces the overhead on the user who analyzes and manages a large collection of video content. Furthermore, it makes conventional natural language processing techniques applicable to multimedia processing.

4.3.7.4 Scene Detection and Visual Object Tracking

As mentioned earlier in Section 4.2.3, visual scene changes are detected by searching for significant changes in the color histogram of successive frames. Then, frame sequences can be divided into scenes. The scene description consists of time codes of the start and end frames, a keyframe (image data in JPEG format) filename, a scene title, and some text representing topics. Additionally, when the user specifies a particular object in a frame by mouse-dragging a rectangular region, an automatic object tracking method is executed and time codes and motion trails in the frame (series of coordinates for interpolation of object movement) are checked out. The user can name the detected visual objects interactively. The visual object description includes the object name, the related URI, time codes, and motion trails in the frame.

Our multimedia annotation also contains descriptions on auditory objects in video. The auditory objects can be detected by acoustic analysis on

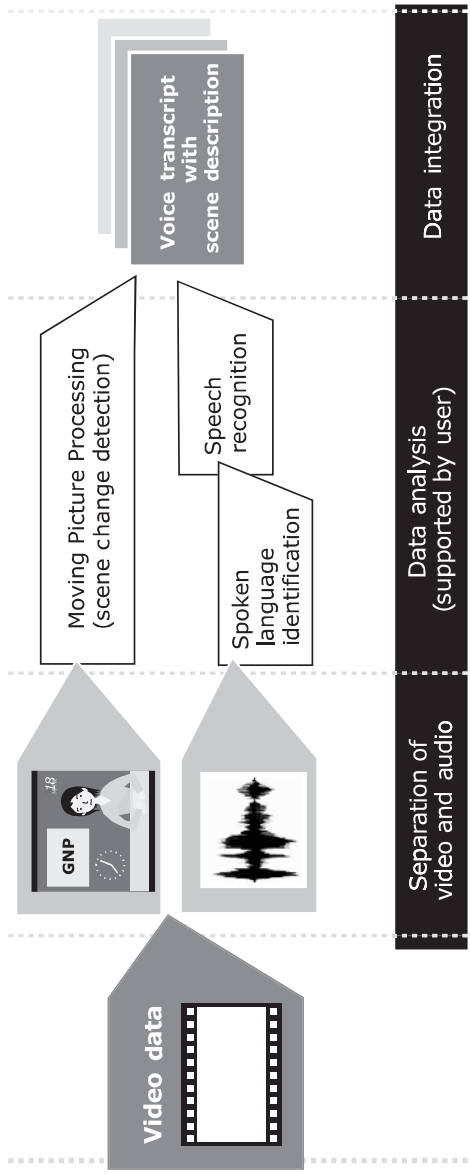


Figure 4.14 Multilingual video data analysis.

the user-specified sound sequence visualized in waveform. An example scene description in XML format is shown here:

```
<scene>
   <seg in="0.066733" out="11.945279" keyframe="0.187643"/>
   <seg in="11.945279" out="14.447781" keyframe="12.004385"/>
   <seg in="14.447781" out="18.685352" keyframe="14.447781"/>
   ...
</scene>
```

An example object description is as follows:

```
<object>
   <vobj begin="1.668335" end="4.671338" name="David"
    description="anchor" img="o0000.jpg" link="http://...">
     <area time="1.668335" top="82" left="34" width="156" height="145"/>
     <area ... />
   </vobj>
   ...
</object>
```

4.4 Semantic Transcoding

Semantic transcoding is a transcoding technique based on annotation, used for content adaptation according to user preferences. The transcoders here are implemented as an extension to an HTTP proxy server. Such an HTTP proxy is called a transcoding proxy.

Figure 4.15 shows the environment of semantic transcoding.

The information flow in transcoding is as follows:

1. The transcoding proxy receives a request URI with a client ID.
2. The proxy sends the request of the URI to the Web server.
3. The proxy receives the document and calculates its hash code.
4. The proxy also asks the annotation server for annotation data related to the URI.
5. If the server finds the annotation data of the URI in its database, it returns the data to the proxy.
6. The proxy accepts the data and compares the document hash code with that of the already retrieved document.
7. The proxy also searches for the user preference with the client ID. If there is no preference data, the proxy uses a default setting until the user gives a preference.

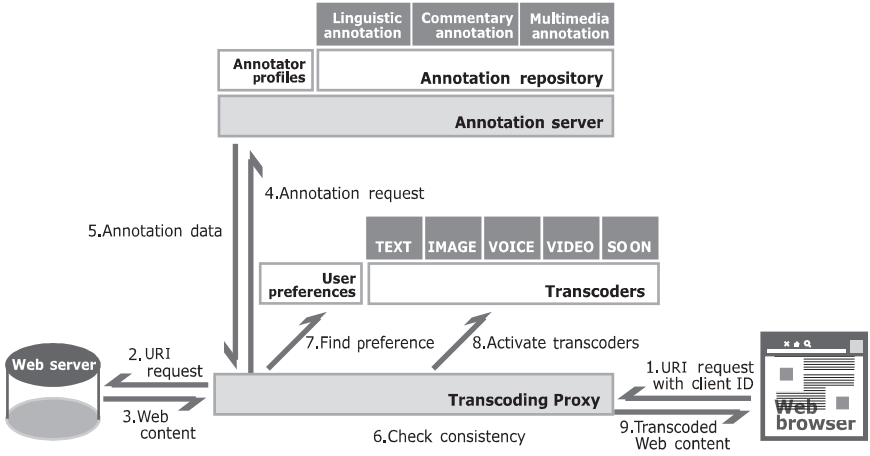


Figure 4.15 Transcoding environment.

8. If the hash codes match, the proxy attempts to transcode the document based on the annotation data by activating the appropriate transcoders.
9. The proxy returns the transcoded document to the client Web browser.

The transcoding proxy and various kinds of transcoding are explained in more detail in the following sections.

4.4.1 Transcoding Proxy

We employed IBM's WBI mentioned in Section 2.5.3 as a development platform to implement the semantic transcoding system [44].

WBI is a customizable and extendable HTTP proxy server. WBI provides APIs for user level access control and easy manipulation of input/output data of the proxy.

The transcoding proxy based on WBI has the following functionality:

1. Management of personal preferences;
2. Gathering and management of annotation data;
3. Activation and integration of transcoders.

4.4.1.1 User Preference Management

For the management of personal preferences, we use a Web browser's cookie to identify the user. The cookie holds a user ID assigned by the

transcoding proxy on the first access, and the ID is used to identify the user and to select user preferences defined previously. The ID stored as a cookie value allows the user, for example, to change an access point using *Dynamic Host Configuration Protocol* (DHCP) with the same preference setting. There is one technical problem. Generally, cookies can be accessed only by the HTTP servers that have set their values, and ordinary proxies do not use cookies for user identification. Instead, conventional proxies identify the client by the host name and IP address. Thus, when the user accesses our proxy and sets/updates the preferences, the proxy server acts as an HTTP server to access the browser's cookie data and associates the user ID (cookie value) and the host name/IP address. When the transcoding proxy works as a coventional proxy, it receives the client's host name and IP address, retrieves the user ID, and then obtains the preference data. If the user changes access point and host name/IP address, our proxy performs as a server again and reassociates the user ID and such client IDs.

4.4.1.2 Collecting and Indexing of Annotation Data

The transcoding proxy communicates with annotation servers that hold the annotation database. The second step of semantic transcoding is to collect annotations distributed among several servers.

The transcoding proxy creates a multiserver annotation catalog by crawling distributed annotation servers and gathering their annotation indices. The annotation catalog consists of server name (e.g., host name and IP address) and its annotation index (set of annotator names and identifiers of the original document and its annotation data). The proxy uses the catalog to decide which annotation server should be accessed to get annotation data when it receives a user's request.

4.4.1.3 Integrating the Results of Multiple Transcoders

The final stage of semantic transcoding is to transcode requested contents depending on user preferences and then return them to the user's browser. This stage involves activation of appropriate transcoders and integration of their results.

As mentioned previously, there are several types of transcoding. In the following sections we describe four types: text, image, voice, and video transcodings.

4.5 Text Transcoding

Text transcoding is the transformation of text contents based on linguistic annotations. We implemented several types of text transcoding such as text summarization, language translation, and dictionary-based text paraphrasing.

4.5.1 Text Summarization

As an example of a basic application of linguistic annotation, I have developed an automatic text summarization system as a part of the text transcoder. Summarization generally requires deep semantic processing and a lot of background knowledge. However, most previous works use several superficial clues and heuristics on specific styles or configurations of documents to summarize.

For example, clues for determining the importance of a sentence include (1) sentence length, (2) keyword count, (3) tense, (4) sentence type (such as fact, conjecture, and assertion), (5) rhetorical relation (such as reason and example), and (6) position of sentence in the whole text. Most of these are extracted by a shallow processing of the text. Such a computation is rather robust.

Present summarization systems [45, 46] use such clues to calculate an importance score for each sentence, choose sentences according to the score, and simply put the selected sentences together in order of their occurrences in the original document. In a sense, these systems are successful enough to be practical, and are based on reliable technologies. However, the quality of summarization cannot be improved beyond this basic level without any deep content-based processing.

Our text summarization method employs a spreading activation technique to calculate the importance values of elements in the text [47, 48]. Since the method does not employ any heuristics dependent on the domain and style of documents, it is applicable to any linguistically annotated document. The method can also trim sentences in the summary because importance scores are assigned to elements smaller than sentences.

A linguistically annotated document naturally defines an intra-document network in which nodes correspond to elements and links represent the semantic relations. This network consists of sentence trees (syntactic head-daughter hierarchies of subsentential elements such as words or phrases), coreference/anaphora links, document/subdivision/paragraph

nodes, and rhetorical relation links. Figure 4.16 shows a graphical representation of the intradocument network.

A spreading activation method is used to evaluate a degree of importance of each node in the intradocument network. The method is an iterative process to calculate an activation value of each node from the activation values of its adjacent nodes connected with normal and/or reference links. The calculation is denoted by equation of matrices.

Suppose the number of all nodes in the network is n, then the connection matrix A is represented as follows:

$$A = \begin{pmatrix} a(1,1) & a(1,2) & \cdots & a(1,n) \\ a(2,1) & a(2,2) & \cdots & a(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ a(n,1) & a(n,2) & \cdots & a(n,n) \end{pmatrix}$$

where the value $a(i,j)$ is 1 if there is a link from node i to node j; otherwise, the value is 0.

Let X be a set of activation values of nodes, and C be a set of external inputs of nodes. They are defined as follows:

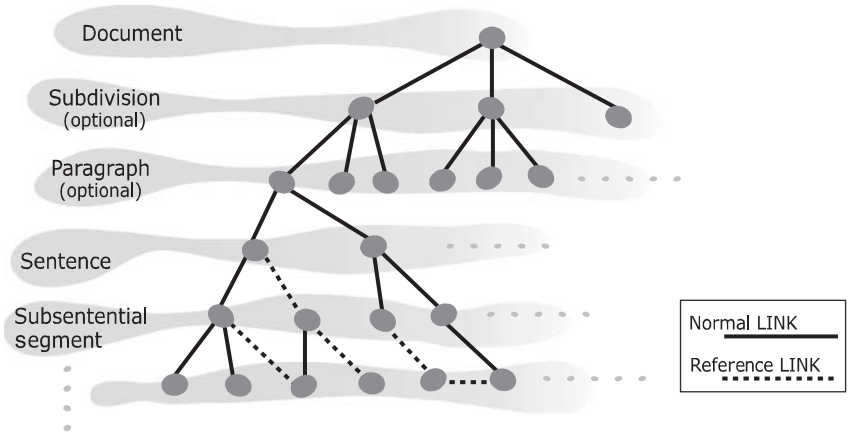


Figure 4.16 Intradocument network.

$$X(t) = \begin{pmatrix} x(t,1) \\ \vdots \\ x(t,n) \end{pmatrix}$$

$$C = \begin{pmatrix} c(1) \\ \vdots \\ c(n) \end{pmatrix}$$

where $x(t, i)$ is an activation value of node i in time t (number of iterations).

The following asymptotic equation is obtained from the above definitions:

$$X(0) = C, \quad X(t+1) = A \cdot X(t) + C \quad (t = 0, 1, 2, ...)$$

X can be calculated by the following equation when $t \rightarrow \infty$ (E is a unit matrix):

$$X = A \cdot X + E \cdot C, \quad X = (E - A)^{-1} + E \cdot C$$

We can get the activation values of all nodes and then an ordering of importance is decided such that the node with a higher activation value is more important.

The summarization algorithm works as follows:

1. Spreading activation is performed in such a way that two elements (nodes) are identical (they have the same activation value) if they are coreferent or one of them is the syntactic head of the other.
2. The unmarked element with the highest activation value is marked for inclusion in the summary.
3. When an element is marked, the following elements are recursively marked as well, until no more elements are found:
   - The marker's head;
   - The marker's antecedent;
   - The marker's compulsory or *a priori* important daughters, the values of whose relational attributes are `agt` (agent), `pat` (patient), `rec` (recipient), `sbj` (syntactic subject), `obj` (syntactic object), `psr` (possessor), `cnt` (content), `cau` (cause), `cnd` (condition), `sbm` (subject matter), and so forth;

- The antecedent of a zero anaphor in the marker with some of the above values for the relational attribute.

4. All marked elements in the intradocument network are generated preserving the order of their positions in the original document.
5. If a size of the summary reaches the user-specified value, then terminate; otherwise go back to step 2.

The size of the summary can be changed by simple user interaction. Thus, the user can see the summary in a preferred size by using an ordinary Web browser without any additional software. The user can also input any words of interest. The corresponding words in the document are assigned numeric values that reflect degrees of interest. These values are used during spreading activation for calculating importance scores.

Figure 4.17 shows the summarization result on the normal Web browser. This is the summarized version of the document shown in Figure 4.10.

4.5.2 Language Translation

The second type of text transcoding is language translation. We can predict that translation based on linguistic annotations will produce a much better result than many existing systems. This is because the major difficulties of present machine translation come from syntactic and word sense ambiguities in natural languages, which can be easily clarified in annotation [49]. An example of the result of English-to-Japanese translation of the previous document is shown in Figure 4.18.

4.5.3 Dictionary-Based Text Paraphrasing

My group also realized a kind of text paraphrasing that makes use of word sense annotation and paraphrases words by embedding their word sense definitions into the original document to generate a new document [50].

Embedding occurs at the user's initiative, which means that the user decides when and where to embed the definition. The generated document can also be the target for another embedding operation, which can be iterated until the document is understandable enough for the user.

One of the examples of embedding a document into another document is quotation. Transcopyright [51] proposes a way for quoting hypertext documents not just embedding the quoted document in the original document but copyright and access control of the quoted document are also

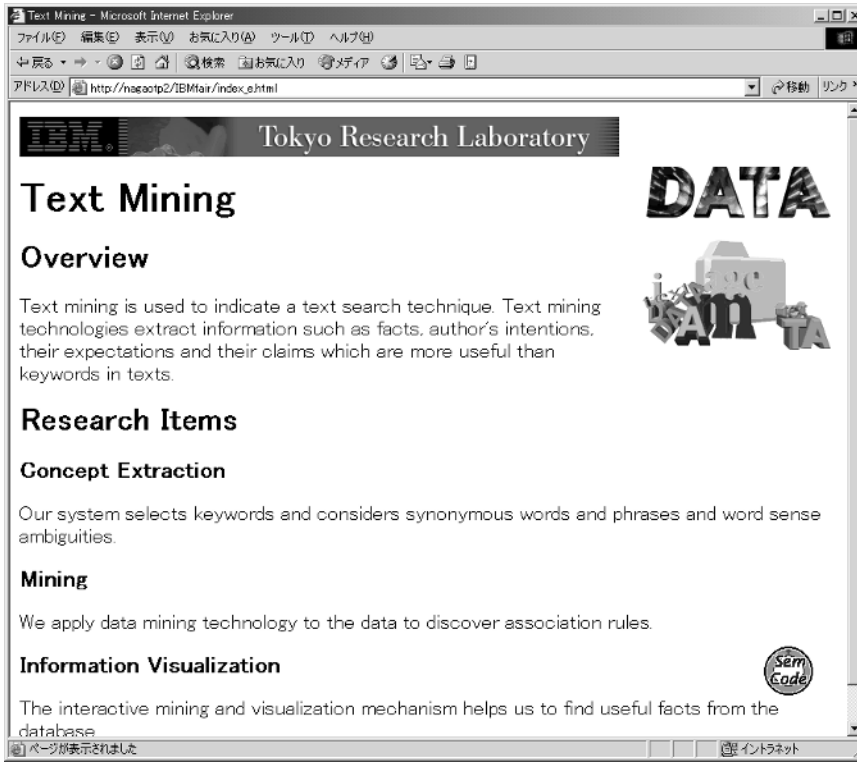


Figure 4.17 Summarized documents.

managed systematically. Quoting, however, means importing other documents as they are. Our approach is to convert other documents so that they fit into the original context, preserving the semantics and improving the readability at the same time.

As the result of embedding, there are no windows hiding any part of the original text, which makes the context easy to follow, and the new document is ready to be used for further natural language processing.

Figure 4.19 shows an example of a Web document after the automatic dictionary lookup. Words marked with a different background color have been successfully looked up.

The conventional method such as showing the definition of a word in a pop-up window hides the neighboring text (Figure 4.20).

Figure 4.21 shows the result of paraphrasing the word "agent." It was successfully paraphrased using its definition "personal software assistants

Figure 4.18 Translated document.

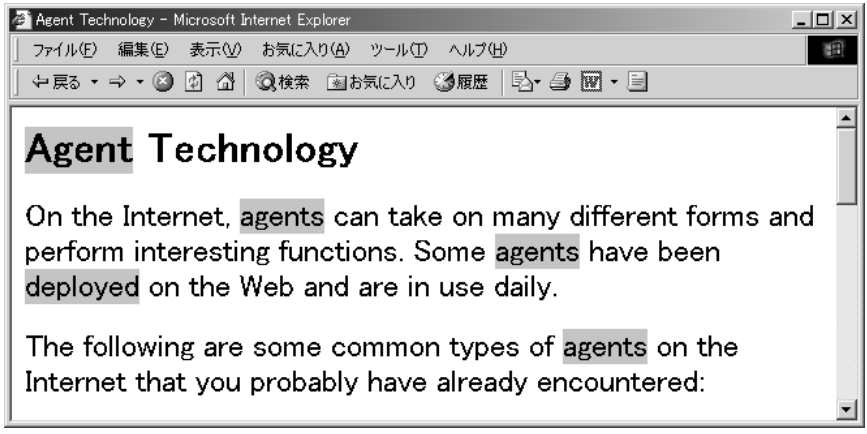


Figure 4.19 Example of a Web document showing dictionary lookup results.

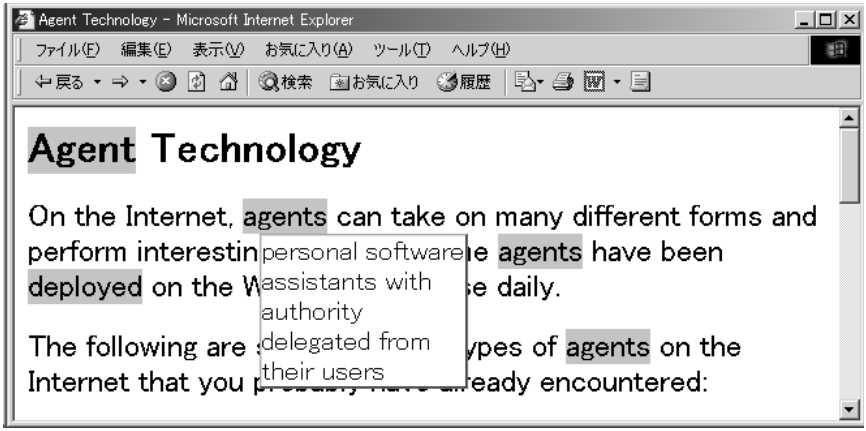


Figure 4.20 Example of the conventional method pop-up window for showing the definition.

with authority delegated from their users." The word "deployed" was also paraphrased by the definition "to distribute systematically." The paraphrased area is marked by a different background color.

Figure 4.22 shows the result of paraphrasing the word in the area already paraphrased. The word "authority" was paraphrased by its definition "power to make decisions."

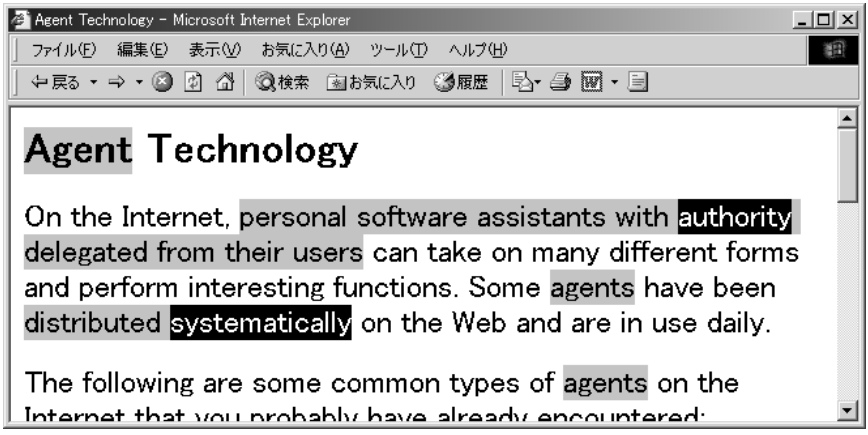


Figure 4.21 Example of the results after paraphrasing "agents" and "deployed."

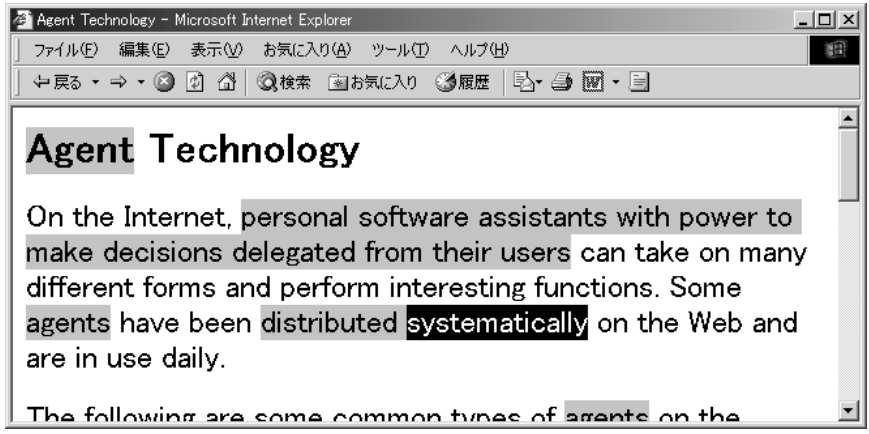


Figure 4.22 Example of incremental paraphrasing.

Semantically embedding word sense definitions into the original document without changing the original context is much more difficult than showing the definition in pop-up windows.

For example, replacing some word in a sentence only with its word sense definition may cause the original sentence to be grammatically wrong or less cohesive. This is because the word sense definitions are usually incapable of simply replacing original words because of their fixed forms. For appropriately integrating the word sense definition into the original context, we employ linguistic annotation to both original documents and the word sense definitions to let the machine know their contexts.

Thus, we need two types of linguistic annotations for our text paraphrasing. One is the word sense annotation to retrieve the correct word sense definition for a particular word, and the other is the syntactic annotation for managing smooth integration of word sense definitions into the original document.

Using the linguistic annotations, our paraphrasing module offers a way to paraphrase words in the document on user demand. One of the objectives of this task is to make on line documents more understandable by paraphrasing unknown words using their word sense definitions.

Users can interactively select words to paraphrase by casual movements like mouse clicks. The paraphrase history is stored for later use such as profile-based paraphrasing (yet to be developed), which automatically selects words

to paraphrase based on user's knowledge. There remains some research issues in profile-based paraphrasing, such as estimation of appropriate knowledge level of each user.

The resulting sentence can also be a target for the next paraphrase. By allowing incremental operation, users can interact with the document until there are no paraphrasable words in the document or the document has become understandable enough. If the paraphrased word/phrase also contains word-sense-annotated words, it can also be the target for the next paraphrasing. This process can be iterated until there are no paraphrasable words in the document.

As described, the original document and the word sense definitions are annotated with linguistic annotation, which means they have graph structures. A word corresponds to a node, a phrase, or sentence to a subgraph. Our paraphrasing is an operation that replaces a node with a subgraph to create a new graph.

Linguistic operations are necessary for creating a graph that correctly fits the original context.

We have made some simple rules (principles) for replacing a node in the original document with a node representing the word sense definition. There are two types of rules for paraphrasing. One is a global rule which can be applied to any pair of nodes: the other is a local rule, which takes syntactic features into account. Below is the description of paraphrasing rules (principles) that we used this time.

Org stands for the node in the original document to be paraphrased by *Def*, which represents the word sense definition node. Global rules are applied first, followed by local rules. Pairs to which rules cannot be applied are left as they are.

4.5.3.1 Global Rules

1. If the word *Org* is included in *Def*, paraphrasing is not performed to avoid the loop of *Org*.
2. Ignore the area enclosed in parentheses in *Def*. That area is usually used for making *Def* an independent statement.
3. Avoid Double negation to avoid the increases the complexity of the original sentence.
4. To avoid redundancy, remove from *Def* the same case-marked structure found both in *Org* and *Def*.
5. Other phrases expressing contexts in *Def* are ignored, since similar contexts are likely to be in the original sentence already.

4.5.3.2 Local Rules

The left column of this table shows the pair of linguistic features[4] corresponding to *Org* and *Def*. (For example $N-N$ signifies the rule to be applied between nodes having the noun features.)

N–N	Replace *Org* with *Def* agreeing in number
N–V	Norminalize *Def* and replace *Org* (e.g., explain → the explanation of)
V–N	If there is a verbal phrase modifying *Def*, conjugate *Org* using *Def's* conjugation and replace *Org*
V–V	Apply *Org*'s conjugation to *Def* and replace *Org*
AD–N	Replace *Org* with any adverbial phrase modifying *Def*
AJ–N	Replace *Org* with any adjective phrase modifying *Def*

The paraphrasing process is as follows:

1. On a user's request, the proxy server retrieves a document through which it searches for words with word sense annotations. If found, the proxy server changes their background color to notify the user of the paraphrasable words.
2. The user specifies a word in the document on the browser.
3. Receiving the word to be paraphrased, the proxy server looks it up in on-line dictionaries using the concept ID assigned to the word.
4. Using the retrieved word sense definition, the proxy server attempts to integrate it into the original document using linguistic annotation attached to both the definition and the original document.

As for paraphrasing rules concerning structured data, a system called *Kura* [52] employed a transfer-based lexico-structural paraphrasing engine.

By paraphrasing, no extra layer (window) is necessary for showing the word sense definition as in conventional methods, and other natural language processing techniques such as summarization, translation, and voice synthesis can be easily applied to the results.

4. *N* stands for the noun feature, *V*, *AJ*, and *AD* for the verb, adjective, and adverb features, respectively.

4.6 Image Transcoding

Image transcoding converts images into those of different size, color (full color or grayscale), and resolution (e.g., compression ratio) depending on the user's device and communication capability. Links to these converted images are made from the original images. Therefore, users will notice that the images they are looking at are not original if there are links to similar images.

Figure 4.23 shows the document that is summarized in one-third size of the original and whose images are reduced to half. In this figure, the preference setting subwindow is shown on the right-hand side. The window appears when the user double-clicks the icon on the lower right corner (the transcoding proxy automatically inserts the icon). Using this window, the user can easily modify the parameters for transcoding.

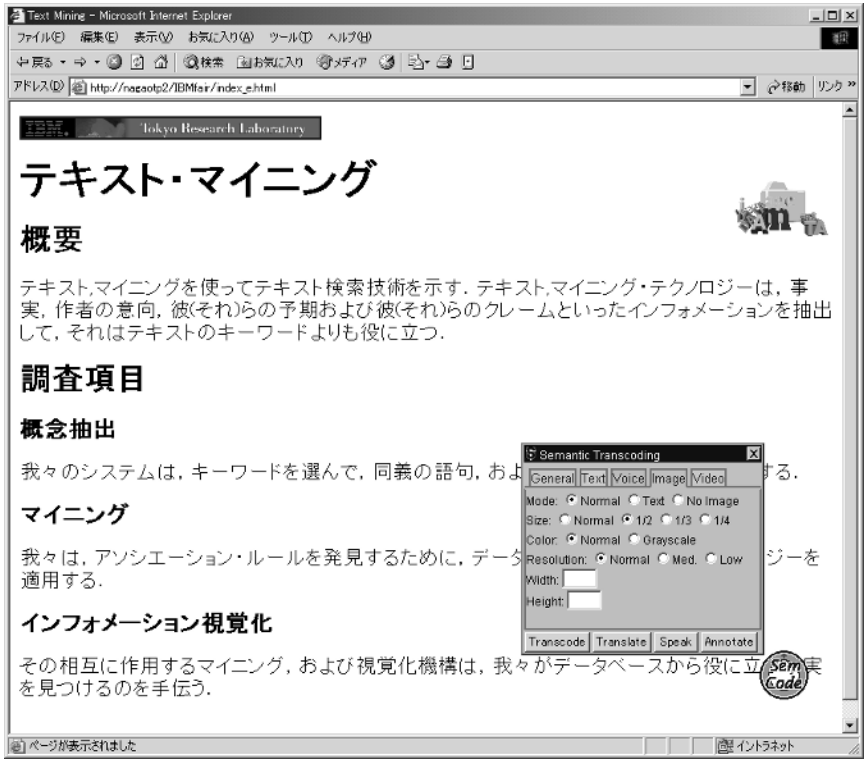


Figure 4.23 Image transcoding and preference setting window.

By combining image and text transcodings, the system can, for example, convert contents to just fit the client screen size.

4.7 Voice Transcoding

Voice synthesis also works better if the content has linguistic annotation. For example, *Speech Synthesis Markup Language* (SSML) has been discussed in [53]. A typical example is processing proper nouns and technical terms. Word level annotations on proper nouns allow the transcoders to recognize not only their meanings but also their readings.

Voice transcoding generates a spoken language version of documents. There are two types of voice transcoding. One is when the transcoder synthesizes sound data in audio formats such as *MPEG-1 Audio Layer 3* (MP3). This case is useful for devices without voice synthesis capability such as cellular phones and PDAs. The other is when the transcoder converts documents into more appropriate style for voice synthesis. This case requires that a voice synthesis program is installed on the client side. Of cource, the client-side synthesizer uses the result of voice transcoding. Therefore, the mechanism of document conversion is a common part of both types of voice transcoding.

Documents annotated for voice include some text in commentary annotation for nontextual elements and some word information in linguistic annotation for the reading of proper nouns and unknown words in the dictionary. The document also contains phrase and sentence boundary information so that pauses appear in appropriate positions.

Figure 4.24 shows an example of the voice-transcoded document in which an audio player is embedded in the top area of the document. When the user clicks the play button, the embedded MP3 player software is invoked and starts playing the synthesized voice data.

4.8 Multimedia Transcoding

Based on multimedia annotation, we developed a module for multimedia transcoding, especially, video summarization and translation. One of the main functions of the system is to generate an interactive HTML document from multimedia content with annotation data for interactive multimedia presentation, which consists of an embedded video player, hyperlinked keyframe images, and linguistically annotated transcripts. Our summariza-

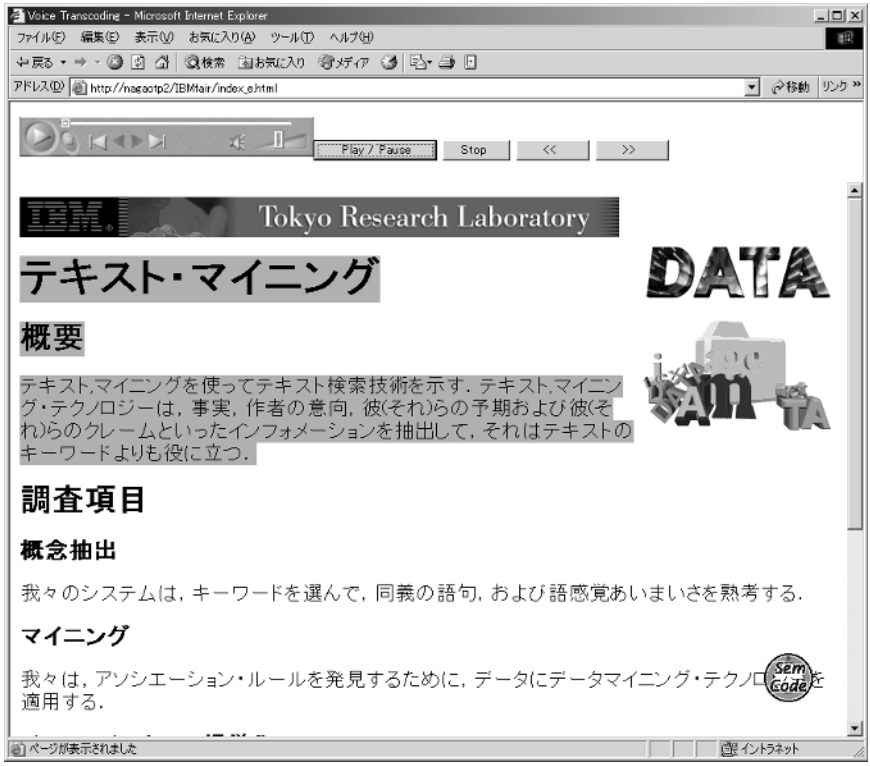


Figure 4.24 Voice transcoding.

tion and translation techniques are applied to the generated document called a multimodal document (see Section 4.8.1).

There are some previous work on multimedia summarization such as Informedia [28] and CueVideo [54]. They create a video summary based on automatically extracted features in video such as scene changes, speech, text and human faces in frames, and closed captions. They can process video data without annotations. Currently, however, the accuracy of their summarization is not good enough for practical use because of the failure of automatic video analysis. Our approach to multimedia summarization attains sufficient quality for use if the data has enough semantic information. As mentioned earlier, we have developed a tool to help annotators create multimedia annotation data. Since our annotation data is declarative, hence task-independent and versatile, the annotations are worth creating if the multimedia content will be frequently used in different applications such as automatic editing and information extraction.

4.8.1 Multimodal Document

Video transformation is an initial process of multimedia summarization and translation. The transformation module retrieves the annotation data accumulated in an annotation repository (XML database) and extracts necessary information to generate a multimodal document. The multimodal document consists of an embedded video window, keyframes of scenes, and transcipts aligned with the scenes as shown in Figure 4.25. The resulting document can be summarized and translated by the modules explained later.

This operation is also beneficial for people having devices without video playing capability. In this case, the system creates a simplified version of the multimodal document containing only keyframe images of important scenes and summarized transcripts related to the selected scenes.

4.8.2 Video Summarization

The proposed video summarization is performed as a byproduct of text summarization. The text summarization is an application of linguistic

Figure 4.25 Multimodal document.

annotation. The method is cohesion-based and employs spreading activation to calculate the importance values of words and phrases in the document.

Thus, the video summarization works in terms of summarization of a transcript from multimedia annotation data and extraction of the video scene related to the summary. Since a summarized transcript contains important words and phrases, corresponding video sequences will produce a collection of significant scenes in the video. The summarization results in a revised version of the multimodal document that contains keyframe images and summarized transcripts of selected important scenes. Keyframes of less important scenes are shown in a smaller size. An example screen of a summarized multimodal document is shown in Figure 4.26.

The vertical time bar in the middle of the screen of the multimodal document represents scene segments whose color indicates if the segment is included in the summary or not. The keyframe images are linked with their corresponding scenes so that the user can see the scene by just clicking its related image. The user can also access information about objects such as people in the keyframe by dragging a rectangular region enclosing them. This

Figure 4.26 Summarized multimodal document.

information appears in external windows. In the case of auditory objects, the user can select them by clicking any point in the time bar.

4.8.3 Video Translation

One type of our video translation is achieved through the following procedure. First, transcripts in the annotation data are translated into different languages for the user choice using the text transcoder, and then, the results are shown as subtitles synchronized with the video. The other type of translation is performed in terms of synchronization of video playing and speech synthesis of the translation results. This translation makes another-language version of the original video clip. If comments, notes, or keywords are included in the annotation data on visual/auditory objects, then they are also translated and shown on a pop-up window.

In the case of bilingual broadcasting, since our annotation system generates transcripts in every audio channel, multimodal documents can be coming from both channels. The user can easily select a favorite multimodal document created from one of the channels. We have also developed a mechanism to change the language to play depending on the user profile that describes the user's native language.

4.8.4 Multimedia Content Adaptation for Mobile Devices

Some mobile phones have a capability of playing multimedia content. In most cases, however, multimedia content requires higher computational costs to retrieve and view than text (with images) content.

The multimedia transcoding technique can also be applied to content adaptation for mobile phones. The adaptation has two steps: creation of a multimodal document from multimedia annotation and splitting of the multimodal document into scene-wise fragments. Each fragment is represented like a card of WML [55]. Figure 4.27 shows the results of video content adaptation for mobile phones. A list of hyperlinks to video clips is coverted into a menu page (the left figure). Selecting an item in the list triggers a transcoding of a requested video and a first card including a small-sized scene keyframe image and a part of a transcript text is displayed (the center figure). A full transcript of each scene is shown by selecting a text bounding box (the right figure).

Another functionality is an interoperability between client devices. For example, a quick browse of video clips are performed by using mobile phones, and some particular scenes in them are carefully watched by using

Figure 4.27 Multimedia content adaptation for mobile phones.

PCs. In this example, a transcoding proxy has a role of device-sharable bookmarks that allow users to leave notes on content URIs with scene IDs to and to retrieve these information from them. Since all connections between clients and content servers are mediated by the transcoding proxy and the users of multiple devices can assign the same user IDs with their clients on the different devices, it is easy to use the transcoding proxy as a common place to exchange data among clients.

4.9 More Advanced Applications

Semantic annotations are usable on many kinds of applications. Some of them are introduced here, such as open scalable platform of annotation sharing and reuse, knowledge discovery from large on-line resources, and multimedia restoration using digital watermarking.

4.9.1 Scalable Platform of Semantic Annotations

A first step towards building intelligent content is to have the infrastructure needed to handle and associate semantic annotations with content. A system

for sharing commentary annotations on the Web such as Annotea [40] can be a simple form of an open annotation infrastructure that is a remark about a document identified by a URI, made by the author of the document or by a third party, with or without author knowledge. In a shared Web annotation system, annotations are stored in specialized servers. The annotations are shared in that everyone having access to an annotation server is able to consult the annotations associated with a given document and add their own annotations.

Semantic annotations that help machines recognize meanings of content are still very difficult for ordinary people to create. However, if they are willing to associate their comments with Web content and are reluctant to embed linguistic annotation markups in their comments by using the annotation editor, then the semantic transcoding proxies can infer the meanings of their comments and associated content.

Therefore, being able to associate such annotations with Web resources is an important milestone for building an intelligent Web. Based on W3C standards including XML, RDF [3] (see Section 3.1.3), RDF Schemas [56] (see also Section 3.1.3), and XPointer/XLink [57] (see Section 3.4.3), we can make our semantic annotation system a more scalable infrastructure for associating annotations with Web content without having to modify the present Web servers.

Some benefits of using W3C standards are as follows:

1. A generic RDF infrastructure allows a variety of applications to reuse the information that is stored in an annotation system with other RDF tools that are not necessarily specific to annotations.
2. W3C specifications contribute to building an open annotation platform such as an RDF Schema for describing the properties of annotations, XPointer for associating annotations to documents, and HTTP for the client/server interactions.
3. By storing annotations in RDF databases, it is possible to make customized queries and responses for further annotation reuse.
4. An annotation RDF Schema defines general properties about annotations. Users can extend it by defining their own annotation types or by adding other annotation properties.
5. A general infrastructure for associating annotations with documents and for storing and retrieving this annotation can be constructed on any operating system environment. In principle, it is possible to build an annotation client on top of any browser that handles XML, XPointer, XLink, and RDF.

Other application systems of annotations can handle the following:

- *Shared bookmarks and keywords:* Shared bookmarks and keywords are quite similar to commentary annotations. The annotation schema provides a set of fixed annotation types. The user is expected to classify his or her annotation by selecting one of these types. With shared bookmarks, the user should be able to define his/her own types on-the-fly, for example, by highlighting keywords on the annotated document. These types can then be used to automatically classify the bookmarks.
- *Annotation display:* Currently, a user can see commentary annotations either by clicking icons next to the fragment of the content that was annotated or by pointing images or hyperlinks that were annotated. A new client system can have other ways for displaying annotations. For example, by embedding the full text, summaries, or paraphrases of the comments in the annotated document.
- *Author information and its retrieval:* Usually, the author profiles are provided as string data. It is possible to expand the schema about the author so that the author is defined by another RDF Schema and use this property in the annotation. It will then be easier to search for the information of the author.
- *Robust associations:* The annotation-content associations based on URI and XPointer are very simple. If a user edits an annotated document, in some cases, the XPointer of an annotation may point to the wrong place and thus become a misleading annotation. A better association will be one that is more tolerant of document changes, but robust enough to prevent misleading annotations.
- *Discussion threads:* When a user wants to reply to an annotation, he or she can either modify the body of the annotation or make a new annotation. This can become cumbersome as we would need to browse each annotation in order to follow the discussion. This situation can be improved by adding new RDF properties for distinguishing such discussions and by showing all the replies to a given annotation in a specialized view.

4.9.2 Knowledge Discovery

I am also planning to apply our technology to knowledge discovery from huge on-line resources.

The annotations will be very useful to extract and share some essential points in the content. For example, an annotator adds comments to several documents, and he or she seems to be a specialist of some particular field. Then, the machine automatically collects documents annotated by this annotator and generates a single document including summaries of the annotated documents.

The annotations are also used in knowledge discovery, where huge amounts of Web content are automatically mined for some essential points. Unlike conventional search engines that retrieve Web pages using user specified keywords, knowledge miners create a single document that satisfies a user's request. For example, the knowledge miner may generate a summary document on a certain company's product strategy for the year from many kinds of information resources of its products on the Web.

Currently, my group is developing an information collector that gathers documents related to a topic and generates a document containing a summary of each document.

A related technology such as text mining is applied to discover a trend or an essence from large on-line text resources. Text mining has made a remarkable progress in business intelligence applications [58]. In particular, the text mining researchers have recognized that incoming customer e-mail (inquiry) can be accurately categorized and *frequently asked questions* (FAQs) are retrieved to semi-automatically compose a reply for the inquiry, and at the backoffice, the accumulated inquiries are analyzed to discover emerging problems and trends of popular products/topics.

Discussion records, such as Web *bulletin board system* (BBS) records and mailing lists, contain a great deal of human knowledge. In such discussion records, several subjects may be discussed in one record and each record may not have an appropriate ''subject'' or ''title.'' Therefore, when people want to know the recent trends from discussion records, they cannot easily find them because each subject may be spread out over several records. Such a possibly dispersed subject is called a ''topic.'' Another difficulty in finding knowledge from discussion records is that several aspects of a single topic may be found in various records. Therefore, it is not easy to retrieve the particular topic by retrieving certain discussion records in isolation. People will find a similar difficulty when fragments of several topics are mentioned in one record. In this case, it is necessary to classify the discussion records according to the topics. A communication knowledge discovery called discussion mining was proposed by IBM researchers [59]. It uses the quotation information to recognize the topic in the records and reconstruct a new document called a *thread summary* that describes a certain branch of the

topic by concatenating the fragments of discussion records. By using the proposed method, people can find individual topics and trends using search and mining techniques over the thread summaries. Since thread summaries also captures human (sender-receiver) relationships, it may be possible to identify ''authority'' and ''hub'' persons from a collection of specific type of thread summaries.

There are many unresolved issues before we can realize true knowledge discovery, but we can say that annotations have greatly facilitated this activity.

Many of the benefits our system offers could be obtained by the full conversion of the underlying content into suitably annotated documents. While this conversion may eventually take place on the Web, our system allows easy adaptation of existing Web documents for transcoding, even when the annotator is not the owner of the original document.

Also, content-based retrieval of on line content including multimedia data is being pursued. Such retrieval enables users to ask questions in natural language (either spoken or written).

4.9.3 Multimedia Restoration

Elements in multimedia annotation are correleated along the time stamp of a recording, with the video and audio content. These elements are employed to provide descriptions of any segment of content. The annotation elements are also used to describe the locations of the repository including correlated content, and to provide information concerning data structures and date conversion and content characteristics.

The annotation elements for video can be used to describe individual scenes such as scene title, start and end time stamps, and locations or pointers of external information linked to each scene. A special multimedia player that is capable of acquisition and interpretation of annotation can present an interactive multimedia content with index information.

The index information allows a user to generate a summary interactively by removing less important scenes. The method to integrate annotations with content has three types:

1. *Look up a table including correspondences between annotations and their associated content.* The semantic transcoding system employs this method. This works if the location or identifier of the content and the content itself are unchanged.

2. *Retrieve an annotation first and get the location or identifier of the associated content from the annotation.* This method works if annotations are searchable by users and the location or identifier of the content and the content itself are unchanged.
3. *Get the content first and automatically retrieve its related annotations.* This method works if pointers or locators to the annotations are embedded in the content and these embedded information can survive after the content is edited, modified, or converted into another format (e.g., from MPEG-2 to MPEG-4, QuickTime, and so on).

Here I concentrate on the third method since it is the only one that is effective even if the multimedia content is redelivered after it is modified, edited, or renamed.

In the case of the time-code–based method for relating annotations with multimedia content, the start time and end time stamps of each scene are not correct when the content is edited. Since timing information included in the annotations is created for the original content, such annotations usually do not work for the edited content. When the content is distributed after editing, users cannot generate appropriate index information using annotations for the original content. Of course, if the annotations are modified according to editing of the associated original content, then users can use the annotations for the edited content without any trouble. However, large numbers of people will edit and reuse multimedia content individually and will give its copies to their friends or acquaintances, then making consistency of time codes between the content and its annotations to be unmanageable.

I proposed a solution to such problem, in which pointers to the annotations is embedded in the content using an information hiding technique. The information hiding technique enables content authors to permanently embed a digital watermark into some frames of the motion picture or audio [60]. It is not affected by format conversions, signal processing, lossy compression, or analog transmission. The watermark is embedded directly in the content of the picture or audio and is practically impossible to remove. Furthermore, the watermark does not affect the quality of the content. The information hiding technique is generally employed for digital watermarking or digital signature, and the information embedded in the content using this technique is sufficiently robust to resist the editing of the content.

The pointer information for annotations corresponding to each scene or phrase in the content is embedded repeatedly in the portion of the content

(i.e., series of frames or duration of speech) relevant to the scene or phrase. Therefore, even when the content is edited, the correlations between the annotations and the content for the scenes or phrases are not destroyed.

The pointer information consists of the address of the annotation corresponding to the content, scene IDs used to identify scenes in the content, and visual/auditory object IDs used to identify visual/auditory objects in the content. The address of the annotation (i.e., URI) identifies the location of the annotation on the network, and is embedded throughout the video signal, while the scene or object IDs identify scene or object elements included in the annotation, and a relevant ID is embedded in a particular time period in which each scene or object appears in the content. Since the addresses of the metadata and the scene/object IDs are embedded using different keys in different content layers, they do not interfere with each other.

A scene ID is embedded in each of the frames of a scene using the data hiding technique. As a specific example, 12 bits of data can be embedded in each frame beginning at a frame corresponding to one arbitrary time code and ending at a frame corresponding to another arbitrary time code that serves as a delimiter for the scene (the creator of the metadata can arbitrarily determine the delimiter that is to be used between scenes). This means that 4,096 scene IDs can be embedded in one frame, and this constitutes a satisfactory amount of data to be used to identify a scene. Since the scene ID is embedded in all the frames of the pertinent scene, even when only part of the scene is extracted, the scene ID can be accurately read from the data for the extracted content.

The address of the annotation is dispersed among and repetitively embedded in the content data using the information hiding technique. As an example, 72 bits of data can be embedded in each interval of about 1 second. Since 72 bits of data are equivalent to 9 bytes, nine information bearing characters, 1 byte for each character, can be embedded. However, with this amount of information, it is difficult to directly embed data having a variable length, such as the URIs. Therefore, a method is employed whereby a string of nine characters that corresponds to each URI is embedded as the address information for the annotation, and a client system that reproduces the content recovers the URI from the embedded address information. Since the data sizes of the addresses (the address information) provided for the annotations are relatively large, addresses cannot be uniformly embedded in all the frames. However, since address information is dispersed and repetitively embedded, so long as the remaining content has, at the least, a specific length, even after the content has been divided by editing, the

probability that URIs can be recovered is high. In the example wherein 72 bits of data are embedded in each interval of 1 second, a URI can be recovered so long as an interval equal to or longer than 1 second remains.

Based on the embedded information, the annotation for the original content is automatically adapted for its modified versions. For example, scene information in a video clip is updated in the following steps:

1. Scan some short interval in the video and detect the address to its annotation.
2. Obtain the annotation from the detected address information.
3. Scan each frame in the video and detect scene IDs.
4. Find a frame wherein scene IDs are changed and get a time stamp of the frame.
5. Collect start and end time codes for each portion including a scene ID.
6. Generate an XML element for each portion such as:

   ```
   <scene id=" detected scene ID" begin=" start time for a portion where in a scene ID is embedded" end=" end time for a portion where in a scene ID is embedded"
   [other attributes such as the scene title that are included in the the annotation concering the same scene ID]>
   ```

7. Create a new index data including the set of the generated XML elements

Based on the above techniques, I also propose a method to restoring scenes that have been missed due to the editing of the original content.

The association is established between the content and the annotation for the original content that is stored in an annotation server, and the annotation is employed, regardless of whether the content has been edited. Thus, when the obtained content has been edited, the user can employ the annotation to restore some particular portions that have been deleted by editing the content. In this process, it is premised, based on information described in the annotation indicating the location of the original content, that an access method is prepared to get the original content (i.e., that the client system is connected to the content server).

Figure 4.28 is a schematic diagram for explaining the process of restoring missing scenes in the content. First, the user obtains the content, and then obtains the annotation from the annotation server using the pointer

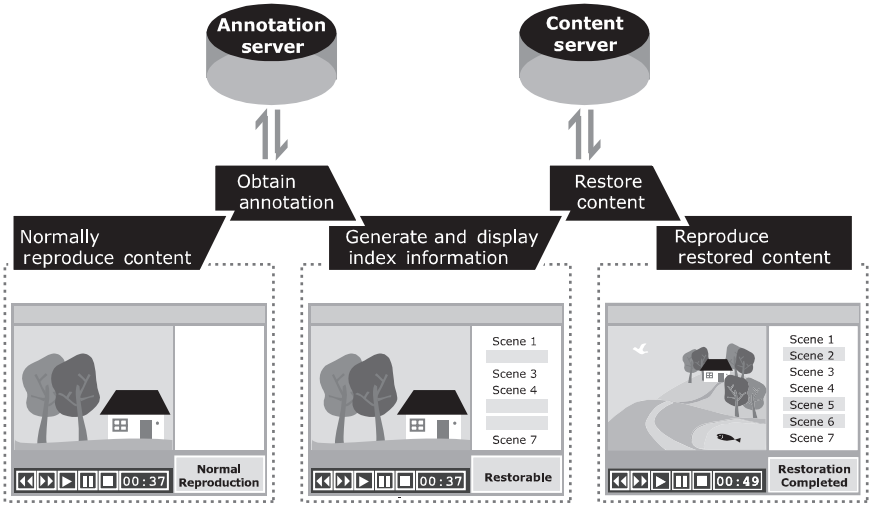


Figure 4.28 Restoration process of multimedia content.

information that is embedded in the content. Thereafter, index information corresponding to the content is generated.

As shown in Figure 4.28, the content is displayed in the main window of the client system, and the index information is shown in the adjacent subwindow. Checking the differences between the generated index and that in the annotation works to find positions and IDs of missing scenes. The user can notice that the content is edited by looking at the index subwindow. The positions of missing scenes are also indicated there.

The user thereafter refers to the information for the missing scenes, and issues a request for the restoration of a desired scene. The restoration request can be issued for all the missing scenes, or for only one or for several selected missing scenes.

The client system employs the location information of the original content in the annotation and obtains it from the content server, and then restores the edited content. Then, the content, together with the restored scene, are reproduced.

When all the scenes missed in the content that was obtained first are restored, the reproduced content becomes equal to the original content. However, instead of downloading the original content, restoring only scenes (differences in the content) requested by the user can reduce communication traffic and the time required for the transfer of data can be shortened. In addition, the user can select only a desired scene for restoration, and the time required for downloading unnecessary scenes can be further reduced.

Since the information concerning the annotation is embedded in the content, a content author can efficiently manage multiple versions of the content that are generated while the final content is being prepared.

I also propose a method for managing a version using information that is embedded in the contents. First, a content and annotation author prepares the primary content and the corresponding annotation, and embeds the pointer information for the annotation in the primary content. When a second content is prepared by editing the primary content, additional pointer information for the annotation that corresponds to the second content is embedded in the second content. Therefore, the pointer information for the annotation that corresponds to the primary content and that for the annotation that corresponds to the second content are embedded in the second content. Similarly, when the second content is further edited, additional pointer information for the annotation that corresponds to a third content is embedded in the third content.

The annotation that corresponds to the edited content (the second or the third) is generated by unifying and processing the annotation that corresponds to the content that has not yet been edited (the primary or the second). Therefore, data concerning the editing history is automatically added to the annotation. Thus, the annotation for all the content used for the editing process can be referred to, and an understanding of the editing condition of the data, from the original (primary) content to the final content, can be obtained by comparing the annotations.

This method for embedding in the content the pointer information for the annotation can also be applied to audio and image contents. When the method is applied to audio content, for example, the address of the annotation can be embedded throughout the audio signals, and a phrase ID can be embedded in each phrase of the audio content. When the method is applied to static image content, merely the address of the annotation is embedded throughout the image signal. Since the data size of still images is smaller than that of moving pictures or of audio data, there will be more requests to view the original content than to acquire missing regions.

4.10 Concluding Remarks

I have discussed a full architecture for creating and utilizing annotation including an authoring tool to create linguistic, commentary, and multimedia annotations and a mechanism to apply such data to semantic

transcoding that automatically customizes Web content depending on user preferences such as text and multimedia summarization and translation.

Linguistic processing is an essential task in those applications, so natural language technologies are very important in intelligent content. The linguistic annotation tool compensates for automatic natural language processing by disambiguation of syntactic, semantic, and pragmatic structures of text.

The main component of the multimedia annotation tool is a multilingual voice transcriptor to generate transcripts from multilingual speech in video clips. The tool also extracts scene and object information semiautomatically, describes the data in XML format, and associates the data with content. I also presented some advanced applications on multimedia content based on annotation. My group has implemented video-to-document transformation that generates interactive multimodal documents, video summarization using a text summarization technique, and video translation.

This technology also contributes to commentary information sharing like Annotea and device-dependent transformation for any device. One of our future goals is to make Web content intelligent enough to answer our questions asked using natural language. I imagine that in the near future we will not use search engines but will instead use knowledge discovery engines that give us a personalized summary of multiple documents instead of hyperlinks. The work in this book is one step toward a better solution of dealing with the coming information deluge.

While our current prototype system is running locally, I am also planning to evaluate the semantic transcoding system with open experiments jointly with Nagoya University and Cyber Assist Research Center in Japan. In addition, I will distribute our annotation editor, with natural language and multimedia processing capabilities.

References

[1] W3C, "The Semantic Web Community Portal," http://www.semanticweb.org/, 2002.

[2] Nagao, K., Y. Shirai, and K. Squire, "Semantic Annotation and Transcoding: Making Web Content More Accessible," *IEEE MultiMedia, Special Issue on Web Engineering*, Vol. 8, No. 2, 2001, pp. 69–81.

[3] W3C, "Resource Description Framework (RDF) Model and Syntax Specification," http://www.w3.org/TR/REC-rdf-syntax/, 2002.

[4] W3C, "Naming and Addressing: URIs, URLs, ...," http://www.w3.org/Addressing/, 2002.

[5] Harnad, S., "The Symbol Grounding Problem," *Physica D*, Vol. 42, 1990, pp. 335–346.

[6] Nakashima, H., and K. Hasida, "Location-Based Communication Infrastructure for Situated Human Support," *Proc. 5th World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2001)*, Vol. IV, 2001, pp. 47–51.

[7] MPEG, "MPEG-7 Context and Objectives," http://drogo.cselt.stet.it/mpeg/standards/mpeg-7/mpeg-7.htm, 2002.

[8] Hasida, K., "Global Document Annotation," http://i-content.org/GDA/, 2002.

[9] Chomsky, N., *Syntactic Structures*, The Hague, Netherlands: Mouton, 1957.

[10] Mann, W. C., and S. A. Thompson, "Rhetorical Structure Theory: Toward a Functional Theory of Text Organization," *Text*, Vol. 8, 1988, pp. 243–281.

[11] Gazdar, G., and C. Mellish, (Eds.), *Natural Language Processing in LISP: An Introduction to Computational Linguistics*, Reading, MA: Addison-Wesley, 1989.

[12] Nagao, K., "A Preferential Constraint Satisfaction Technique for Natural Language Analysis," *Proc. 10th European Conference on Artificial Intelligence (ECAI-92)*, New York: John Wiley & Sons, 1992, pp. 523–527.

[13] Seo, J., and R. F. Simmons, "Syntactic Graphs: A Representation for the Union of All Ambiguous Parse Trees," *Computational Linguistics*, Vol. 15, 1989, pp. 19–32.

[14] Takeda, K., "Designing Natural Language Objects," *Proc. International Symposium on Database Systems for Advanced Applications*, 1991, pp. 444–448.

[15] Procter, P., (Ed.), *Longman Dictionary of Contemporary English*, Essex, England: Longman Group Limited, 1978.

[16] Chodorow, M. S., R. J. Byrd, and G. E. Heidorn, "Extracting Semantic Hierarchies from a Large On-Line Dictionary," *Proc. 23rd Annual Meeting of the Association for Computational Linguistics (ACL-85)*, 1985, pp. 299–304.

[17] Nakamura, J., and M. Nagao, "Extraction of Semantic Information from an Ordinary English Dictionary and Its Evaluation," *Proc. 12th International Conference on Computational Linguistics (COLING-88)*, 1988, pp. 459–464.

[18] Guthrie, L., et al., "Is There Content in Empty Heads?" *Proc. 13th International Conference on Computational Linguistics (COLING-90)*, 1990, pp. 138–143.

[19] Maruyama, H., "Structural Disambiguation with Constraint Propagation," *Proc. 28th Annual Meeting of the Association for Computational Linguistics (ACL-90)*, 1990, pp. 31–38.

[20] Montanari, U., "Networks of Constraints: Fundamental Properties and Applications to Picture Processing," *Information Sciences*, Vol. 7, 1974, pp. 95–132.

[21] Mackworth, A. K., "Consistency in Networks of Relations," *Artificial Intelligence*, Vol. 8, 1977, pp. 99–118.

[22] Xuang, X., Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh, Scotland: Edinburgh University Press, 1990.

[23] Erman, L., et al., "The Hearsay II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," *ACM Computing Surveys*, Vol. 12, No. 2, 1980, pp. 213–254.

[24] Hayes, P. J., et al., "Parsing Spoken Language: A Semantic Caseframe Approach," *Proc. 11th International Conference on Computational Linguistics (COLING-86)*, 1986, pp. 787–792.

[25] Young, S. R., et al., "High Level Knowledge Sources in Usable Speech Recognition Systems," *Communications of the ACM*, Vol. 32, No. 2, 1989, pp. 183–194.

[26] Chung, S., D. Moldovan, and R. F. DeMara, *A Parallel Computational Model for Integrated Speech and Natural Language Understanding*, Technical Report IMPACT-92-008-USC, International Consortium for Massively Parallel Advanced Computing Technologies, 1992.

[27] Nagao, K., K. Hasida, and T. Miyata, "Understanding Spoken Natural Language with Omni-Directional Information Flow," *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, San Francisco, CA: Morgan Kaufmann Publishers, 1993, pp. 1268–1274.

[28] Smith, M. A., and T. Kanade, *Video Skimming for Quick Browsing Based on Audio and Image Characterization*, Technical Report CMU-CS-95-186, School of Computer Science, Carnegie Mellon University, 1995.

[29] Zhang, H., C. Low, and S. Smoliar, "Video Parsing and Indexing of Compressed Data," *Multimedia Tools and Applications*, Vol. 1, 1995, pp. 89–111.

[30] Lucas, B. D., and T. Kanade, "An Iterative Technique of Image Registration and Its Application to Stereo," *Proc. 7th International Joint Conference on Artificial Intelligence (IJCAI-81)*, 1981, pp. 674–679.

[31] Rowley, H., S. Baluja, and T. Kanade, *Human Face Detection in Visual Scenes,* Technical Report CMU-CS-95-158, Computer Science Department, Carnegie Mellon University, 1995.

[32] Sato, T., et al., "Video OCR: Indexing Digital News Libraries by Recognition of Superimposed Caption," *ACM Multimedia Systems, Special Issue on Video Libraries*, Vol. 7, No. 5, 1999, pp. 385–395.

[33] W3C, "XML Pointer Language (XPointer) Version 1.0," http://www.w3.org/TR/xptr/ 2001.

[34] Rivest, R., "RFC1321 The MD5 Message-Digest Algorithm," http://www.faqs.org/ rfcs/rfc1321.html, 1992.

[35] Maruyama, H., K. Tamura, and N. Uramoto, *XML and Java: Developing Web Applications*, Reading, MA: Addison-Wesley, MA, 1999.

[36] TEI, "Text Encoding Initiative," http://www.uic.edu/orgs/tei/, 2002.

[37] CES, "Corpus Encoding Standard," http://www.cs.vassar.edu/CES/, 2002.

[38] EAGLES, "Expert Advisory Group on Language Engineering Standards," http://www.ilc.pi.cnr.it/EAGLES/home.html, 2002.

[39] Miller, G., "WordNet: A Lexical Database for English," *Communications of the ACM*, Vol. 38, No. 11, 1995, pp. 39–41.

[40] Kahan, J., et al., "Annotea: An Open RDF Infrastructure for Shared Web Annotaions," *Proc. 10th International World Wide Web Conference (WWW-10)*, 2001.

[41] Roscheisen, M., C. Mogensen, and T. Winograd, *Shared Web Annotations as a Platform for Third-Party Value-Added Information Providers: Architecture, Protocols, and Usage Examples*, Technical Report CSDTR/DLTR, Computer Science Department, Stanford University, 1995.

[42] Schickler, M. A., M. S. Mazer, and C. Brooks, "Pan-Browser Support for Annotations and Other Meta-Information on the World Wide Web," *Computer Networks and ISDN Systems*, Vol. 28, 1996.

[43] Nagao, K., S. Ohira, and M. Yoneoka, "Annotation-Based Multimedia Summarization and Translation," *Proc. 19th International Conference on Computational Linguistics (COLING-02)*, 2002.

[44] Ihde, S. C., et al., "Intermediary-Based Transcoding Framework," *IBM Systems Journal*, Vol. 40, No. 1, 2001, pp. 179–192.

[45] Watanabe, H., "A Method for Abstracting Newspaper Articles by Using Surface Clues," *Proc. 16th International Conference on Computational Linguistics (COLING-96)*, 1996, pp. 974–979.

[46] Hovy, E., and C. Y. Lin, "Automated Text Summarization in SUMMARIST," *Proc. ACL Workshop on Intelligent Scalable Text Summarization*, 1997.

[47] Hasida, K., S. Ishizaki, and H. Isahara, "A Connectionist Approach to the Generation of Abstracts," in G. Kempen, (Ed.), *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics*, Boston, MA: Martinus Nijhoff, 1987, pp. 149–156.

[48] Nagao, K., and K. Hasida, "Automatic Text Summarization Based on the Global Document Annotation," *Proc. 17th International Conference on Computational Linguistics (COLING-98)*, 1998, pp. 917–921.

[49] Watanabe, H., et al., "An Annotation System for Enhancing Quality of Natural Language Processing," *Proc. 19th International Conference on Computational Linguistics (COLING-02)*, 2002.

[50] Higashinaka, R. and K. Nagao, "Interactive Paraphrasing Based on Linguistic Annotation," *Proc. 19th International Conference on Computational Linguistics (COLING-02)*, 2002.

[51] Nelson, T. H., "Transcopyright: Dealing with the Dilemma of Digital Copyright," *Educom Review*, Vol. 32, No. 1, 1997, pp. 32–35.

[52] Inui, K., et al., "KURA: A Transfer-Based Lexico-Structural Paraphrasing Engine," *Proc. 6th Natural Language Processing Pacific Rim Symposium, Workshop on Automatic Paraphrasing: Theories and Applications*, 2001.

[53] SABLE, "A Speech Synthesis Markup Language," http://www.cstr.ed.ac.uk/projects/ssml.html, 2002.

[54] Amir, A., et al., "CueVideo: Automated Indexing of Video for Searching and Browsing," *Proc. SIGIR'99*, 1999.

[55] Wireless Application Protocol Forum, "Wireless Markup Language Version 2.0," http://www1.wapforum.org/tech/documents/WAP-238-WML-20010626-p.pdf, 2001.

[56] W3C, "Resource Description Framework (RDF) Schema Specification 1.0," http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, 2000.

[57] W3C, "XML Linking Language (XLink) Version 1.0," http://www.w3.org/TR/xlink/, 2001.

[58] Nasukawa, T., and T. Nagano, "Text Analysis and Knowledge Mining System," *IBM Systems Journal*, Vol. 40, No. 4, 2001.

[59] Murakami, A., K. Nagao, and K. Takeda, "Discussion Mining: Knowledge Discovery from Online Discussion Records," *Proc. 1st NLP and XML Workshop*, 2001, pp. 31–38.

[60] Katzenbeisser, S., and F. A. P. Petitcolas, (Eds.), *Information Hiding Techniques for Steganography and Digital Watermarking*, Norwood, MA: Artech House, 1999.

5

Future Directions of Digital Content Technology

This chapter will conclude the book with some future directions of digital content technologies and their applications such as new business models and social information infrastructures.

The business models include next-generation portal services on proxy servers and personalized information editing services using transcoding. Social information infrastructures are not just for business but also for education and politics. Some projects on a digital government are also presented. The digital government essentially requires a social information infrastructure for information disclosure and direct communication with citizens.

In this last chapter, I also want to answer the question that was presented in the first chapter: How can we survive the information deluge caused by the recent evolution of the World Wide Web? I am sure that readers of this book may already have an idea of this answer after reading the previous chapters. In order to clearly answer the question, I introduce two technical topics that are very important to associate the enhanced digital content with our daily life. They are agent technology and ubiquitous computing.

Agent technology has been mainly developed by artificial intelligence researchers. It consists of learning mechanisms for adaptation of changing environments and generation of autonomous behaviors, communication mechanisms for information sharing and negotiation with other autonomous entities, and problem solving mechanisms for reasoning about presuppositions and consequences, constructing plans of actions, and achieving goals. For information seeking tasks, an intelligent module called an agent should

understand its user's intention and meanings of target content. The agent technology makes digital content more intelligent. Agents can easily process semantically annotated content, and efficiency of content retrieval that meets user's requirements will become very high.

Ubiquitous computing is a new concept of human-computer interaction. Users' physical behaviors will be related to some sort of information processing. Such information processing will be context-aware and be adapted to the real-world environment in ubiquitous computing. The ubiquitous computing makes digital content more situated. Such situated content can quickly capture the users' focus of attention like outstanding commercial films or eye-catching signboards because it can omit a background context and decrease the users' cognitive load to understand it. Semantically annotated content is always transcoded according to a user's interests and methods to view it. So it is possible to deeply adapt content to the user's situation and context in the real-world by employing ubiquitous computing technology. Situated content with well-suited style and presentation never makes humans lost in information space.

The future of digital content will be a sharable knowledge of the world. Globally shared digital content with semantic annotation can be a source of the world knowledge that is very useful in our everyday lives. Humans interacting with the knowledge system will become well educated. The knowledge system can tell us about world affairs and histories in a form that is easily understabable for individual persons. Humans will learn what the truth is and on what they should set their sights.

5.1 Content Management and Distribution for Media Businesses

As described in previous chapters, annotation is a key technique to create and manage information-rich content. Transcoding is also an important method to distribute personalized content for any media. Annotation and transcoding also work for security of content. The combination of these methods enables digital content/media business companies to manage and distribute rich content securely for any media, to anyone, to any place or any device, at any time.

My idea of a business model construction is briefly explained in the following list:

- Digital Content + Annotation → Enhanced Media
- Enhanced Media + Rights → Media Asset

- Media Asset + Distribution Channels → Delivery Service
- Delivery Service + User Profiles + Transcoding → Personalized Service
- Personalized Service + Security Models + Transcoding → Secured Personalized Service (= New Business Model/Opportunity)

I describe two important topics in this business model: rights management and transcoding-enabled services.

5.1.1 Rights Management

Rights management deals with copyright, ownership, and other properties related to content and annotation. The Content ID Forum has been discussing standard frameworks of rights management [1]. Content ID is issued by one of the content ID management centers, which can be operated by content holders, content creators, or even third parties, under the supervision of the world *Registration Authority* (RA). A unique content ID is embedded into each content using a header and/or digital watermark. Parts of attribute information can also be embedded into each content as *distribution content descriptors* (DCD). The full set of content attribute information is stored in databases (IPR-DB) managed by content ID management centers using content ID as a key.

Possible functions that are provided by a content ID mechanism are:

- *Copyright function:* Enables anyone to get copyright-related information on a particular content and attribute information such as date of creation;
- *Content-attribute search function:* Provides searching and retrieving of contents under the uniform standards;
- *Commercial function:* Enables efficient collection of content distribution history, content sales history, and other content-related information across all agents and throughout the commercial field;
- *Fraudulent-use detection function:* Enables detection of fraudulent use of content on the network using Content ID as a key and also enables users to examine usage rules of the content, which will provide mechanisms to ensure observance of those usage rules;

- *Certification function:* Enables content to be certified as free from alteration and tampering after being issued by the official author;
- *Editing-history reference function:* Enables viewing of edit history of content stored on an IPR database;
- *Database common-key function:* Simplifies searching and mutual referencing through common identification codes assigned when constructing digital archives.

Using content ID as a key, charging and royalty allocation for a specific content item can be performed automatically without an intermediary thereby reducing handling fees. Again, using content ID as a key, accurate metering can be performed even for partial sales of contents based on mechanical processing and for various user-usage formats.

Standardization of methods for detecting fraudulent use of content means that entities that monitor such fraudulent use on the network (Net Police) can be divided into global and functional levels (for example, there can be people that insert watermarks and people that detect them).

Compared to the present situation in which services are provided separately, productivity increases for Net Police entities on the whole.

From the viewpoint of distributors, statistical processing on the basis of a common content ID code can now be performed. This will enable the sales breakdown of content (such as who purchased what type of content when and in what units) to be determined based on the distribution and sales history of that content. From the viewpoint of copyright holders, agents and distributors can be searched through statistical information based on content ID embedded in one's own content.

There will be no need to modify content because of ambiguity in copyright information, promoting the frequent reuse of existing content. Productivity in content creation rises and the total cost of content creation drops. From the viewpoint of copyright holders, originality can be emphasized regardless of the distribution path and reuse format.

For paintings, antiques, and the like, value has been traditionally associated with scarcity (as in "this is the only one of its kind"). In a similar manner, the number of officially released instances of certain content can be limited by assigning individual IDs to each instance of digital content created by a copyright holder and by then announcing the owners of such content instances. Scarcity can also be attributed to digital content and value allocated accordingly.

5.1.2 Portal Services on Transcoding Proxies

Transcoding enables business opportunity to extend from conventional portal Web sites to proxy-based programmable channels with personalized content.

Digital content will ultimately redefine how we work, communicate, interact, learn, and play. Since transcoding dynamically adapts single source content to multiple formats and multiple end-user devices, Web design paradigm will soon change from today's static content to dynamic digital channels.

The effective integration of profiles and permission marketing will allow the delivery of personalized content and new types of products created from a media repository.

A new media business model will require that the delivery of content be optimized for different forms of interconnected digital channels, billing or commerce systems, customer service, promotion/marketing, and ease of use. Proxy servers or intermediaries will play a very important role in such new business models because they are intersections of informarion flow and better places where value-added operations are performed.

The first-generation publishing for the Web was publishing news from printed editions, information access, and e-commerce for physical goods. News portals emerged as some of the first Web brands and supplied content to other portals such as *America Online* (AOL) and *Cable News Network* (CNN).

Publishers of digital content on the Web are now building the second-generation portals or "hubs" for their business customers and consumers supporting multiple media types, personalization, and multiple end-user devices. Therefore, there is much of potential for transcoding proxies to be a next portal for a personalized publishing service.

Transcoding proxies also have a potential for understanding user demands and connecting them with appropriate advertising and merchandising services. Since publishers want to exploit all their content to better advise their users or customers of their services, they will seek ways to better target content from their repository to the users' profiles by using transcoding proxies. The desire is to fully exploit and monetize content that is today less accessible for potential users and is dormant such as less reusable archives.

The ultimate goals of digital publishers are to create a better product, to bring other relevant content to market, to build brand loyalty, and then to increase revenues. I think that transcoding of enhanced media with annotation opens up the true beginning of new businesses with enabled digital content.

5.1.3 Secure Distribution Infrastructure

The next challenge is to build the secure network infrastructure for digital content. This has up until now proven to be complex and challenging from an integration perspective. A transcoding-based security architecture for content access control such as XACL [2] (see Section 3.3.2) will be a key technology in this processes.

Just as the Web made every company a publisher, digital media will make every company a broadcaster. Personalized documents, video, and audio that are securely distributed via transcoding will soon transform the Internet into one of several key digital channels or networks.

We will get a vast opportunity to redefine industries, create new business models, and particpate in the emerging value chain for digital content creation, management, and distribution.

5.2 Intelligent Content with Agent Technology

Digital content technology and agent technology are complementary to each other for a more intelligent content that can be considered as a personalized intelligent knowledge source.

Agent technology has been implemented as so-called software agents such as *Maxims* [3] and *Softbots* [4]. Maxims is a software agent that classifies e-mail according to their importance and urgency. Softbot is a communication-capable information-seeking agent.

The Maxims agent has mechanisms for learning and adapting. In this case, to learn is to acquire the user's habits, preferences, and interests and to be able to adjust the parameters of probabilities, utilities, and thresholds such as *tell-me* and *do-it* thresholds. If an agent's certainty factor on its decision is higher than the do-it threshold, then the agent will do it autonomously. If the factor is less than do-it but higher than tell-me threshold, then the agent asks the user for his or her command or control.

Agent learning plays a role of user personalization that can determine the system's behavior by considering the user's preferences. This function is useful when determining what tasks are to be performed and at what time to begin them, as well as choosing the content to be presented and the timing of the presentation.

Semantically annotated content can facilitate agent learning, because the semantic features contribute to precise classification of conditions that trigger the agent's individual behavior.

A software agent collaborates with other agents as in a multiagent system. The agent communicates with such Internet agents as Softbots when searching for information on the Internet. It also communicates with other people's personalized agents to gain knowledge about the humans with whom the user is talking or planning to meet.

Multiagent collaboration also works for information filtering. I call this agent social filtering. The agents are personalized for their specific users and have knowledge of the users' interests. So, the agent can automatically search for information that will be required by the user in the near future, then filter it based on other users' recommendations that are sharable and accessible to the agent.

The *Recommender System* is a prototype system of collaborative filtering based on recommendation [5]. The system asks the user's preferences using some prepared questions and then matches a list of recommendations to the preferences. The major drawback to this kind of system is that it is only effective when the each user's preferences are broadly distributed over various interest areas.

I have been designing and implementing an agent social filtering technique based on the Recommender System with a credit assignment capability via multiagent communication. In this system, agents produce recommendations through interaction with the user. The information-seeking agents select the most appropriate recommendation that is produced by the agent of the highest credit.

Agent technology makes digital content more intelligent and usable so that content better fits the user intention and context.

5.2.1 Agent Augmented Reality

I have proposed a new concept called *agent augmented reality* that integrates technologies on augmented reality and software agents [6]. Augmented reality is a big trend in human-computer interaction that bridges between computer-synthesized worlds and the real-world. This research has as its main theme the overlay of computer-synthesized images onto the user's real-world view. Examples are the work of Bajura et al. [7] and Feiner et al. [8].

I have introduced a special agent called a *real-world agent*. A real-world agent is a software agent that recognizes the user's real-world situation, is customized to her preferences and interests, and performs tasks in the information world (e.g., cyberspace). Real world agents can communicate

with humans using natural language and with other software agents using an agent communication language such as *Knowledge Query and Manipulation Language* (KQML) [9]. Situation awareness will help the agents to recognize human intentions by using context-related information from their surroundings.

Collaboration among the real-world agents also contributes to group gatherings. In this case, the agents would inform each other of their users' current locations, collaborate in determining an appropriate meeting point, and then navigate the users to the rendezvous. Since the agents are always aware of the users' real-world situations, they can work on not only meeting scheduling, but also coordinating physical meetings.

The real-world agents also work for the augmentation of human-human communication. Personalized real-world agents can add a supplemental context about personal information and current status of users to their messages. This can help receivers of messages to understand their context better, causing the efficiency of communication to increase.

An example application of this system is a social matchmaker that searches for a person who has the same interests as the user when they are participating in a meeting or at a party. Agents communicate with other participants' agents and inquire as to their users' interests. The user will be able to interface with the person selected by the agent very smoothly because they may have similar background and/or experiences.

Another example is an intelligent telephone operator that checks the current status of the potential callee before the user makes a call to that person. If the potential callee is very busy or talking to another person, the agent will recommend that the user not call at that time. Again, agents will always take care to protect the privacy of humans and will cooperate to satisfy their users' demands.

The *Remembrance Agent* developed at MIT Media Lab [10] is a memory supplement tool that continuously watches over the shoulder of the user of a wearable computer and displays one-line summaries of previous e-mails, on-line papers, and other text information that might be relevant to the user's current context. These summaries are listed on the bottom few lines of a heads-up display, so the user can read the information with a quick glance. To retrieve the whole text described in a summary line, the user hits a quick chord on a chording keyboard.

The real-world agents can also help users make use of previous conversations within the context of the current conversation, by showing a summary of texts like the Remembrance Agent. This will facilitate the conversation by reducing duplication.

5.2.2 Community Support System

My group is developing a community support system that integrates augmented communication via real-world agents and asynchronous communication based on e-mails and bulletin boards. Users can attach their agents to text messages for conveying some contextual information about their physical environments. Also, they can hide their messages inside agents by translating the messages into an agent language whose semantics are well defined by the agent's ontology. In this case, more advanced communication will be possible since agents can understand the content of messages and will be able to convey nuances as well as physical contexts. Futhermore, interactions between the messenger agent and the receiver agent will result in intelligent filtering and summarization of messages.

Community activities will be supported by the agent-based intelligent message handling system. The communication among community members will overcome, not only the gap of space but also that of time. The communication will be customized for each member by her agent using her personal information. Received messages will be filtered according to user preferences. Messages being sent will be personalized with the user's situational information (locations, physical environments, mental states, and so forth).

Cooperation among agents will work as social filtering that is a type of information filtering based on recommendations given by other users. The agents exchange the users' recommendations with their interests about some specific topics. A recommendation includes a user or agent ID, keywords, an identifier of the information resource (e.g., URL), an annotation of contents, and an evaluation value (normalized positive value). The evaluation value is decided considering a frequency of access to the information and counts of usage, reference, and recommendation of the information. The recommendations are generated semi-automatically and stored in a shared database. First, an agent searches for recommendations about a particular topic. Then, the agent tries to contact agents that are responsible for the selected recommendations. Next, the agent tries to assign a credit to each responsible agent contacted. The credit is determined through communication between the requesting agent and the responsible agent, because the credit depends on a task of the requesting agent and a career of the responsible agent. The agent repeats the credit assignment process for all selected recommendations. Then, it gives the user the recommendations that have higher credits.

Another important function of the community support system is processing semantically annotated documents. Agents read documents with

annotations and translate them into expressions in their own language that are unambiguous and used for content-based retrieval and summarization. Although converting natural language texts to agent language is not so easy in general, it is not difficult to do that on annotated content that has disambiguated sentence meanings and identified objects linked with referring expressions. The semantic annotation is tightly coupled with the agent ontology and the agent language construction.

Text and multimedia content can be shared with community members. Each member can annotate information on the shared video or audio. By preparing multiple layers of annotation for each content, the community support system can combine or separate several people's annotation. Also, using annotator's information (e.g., interests and preferences), the agent can select more appropriate annotated information for the user. I think that these jointly annotated contents will also work as a common memory of the community. It will be elaborated by the community and will become common knowledge. The knowledge will support our daily activities.

The real-world agents can utilize real-world information as well as online digital content. Considering physical environments, the agents can guide their users to meet in some place in the real-world. The agents negotiate and decide the most appropriate time and place to meet. Since interagent communication includes information on physical environments and personal status/plans, mutual understanding for each other will be established more efficiently than by current e-mails or telephone calls.

When people have agents communicating with each other, there is a problem of personal privacy. In addition, before one can depend on his agent to perform properly for him, he must fully understand the rules used by the agent and become convinced that the agent will do this consistently and reliably. These are the same problems you have when using people to act as agents for you. We urgently have to propose solutions for these issues.

This idea is closely related to the concept of ubiquitous computing that is presented in the following section.

5.3 Situated Content with Ubiquitous Computing

In the framework of ubiquitous computing, recognizing the human environment will become easier, because it proposes that very small computational devices (i.e., ubiquitous computers) be embedded and integrated into the physical environment in such a way that they operate smoothly and almost transparently. These devices are aware of their physical

surroundings, and when a human uses a physical device that contains ubiquitous computers or enters some area where physically embedded computers are invoked to work, these computers are aware of the human's activities. From the viewpoint of reliability and cost-performance, ubiquitous computing does not compare well with mobile computing, since ubiquitous computers require very long battery lives and are significantly difficult to maintain. In addition, when ubiquitous computers are personalized to users, as in an active badge system [11], for example, all user personal data is processed in the shared environment, while in mobile computing, each user's personal information can be encapsulated within their own machine. So, ubiquitous computing also experiences privacy problems.

I use the term "situated content" to mean the content that is well matched with the situation in the real-world context. This can be very efficient for people to understand, because the background context has already been recognized by them. An example of situated content could happen in the KABUKI guidance system. KABUKI is a type of traditional Japanese drama. The guidance system uses a small radio-like device and a headphone for each member of the audience. The system gives the user a better understanding using a real-time narrative that describes the action, without disturbing the overall appreciation of the KABUKI drama. The pace of the action dynamically changes with each performance and if the narration were to get ahead of, or drop behind the action, it would become meaningless, so synchronizing the narration with the action is very important and also very difficult. Currently, narrations are controlled manually, but it is possible for the system to be automated by using a real-world situation awareness technique.

5.3.1 Situation Awareness

A real-world situation includes the place where the human is, the time when an event occurs, living and nonliving things that exist in the vicinity, and any physical action that is being performed (e.g., looking at something).

Using situation awareness, people can naturally interact with computers without being especially conscious of the computers' domains or regulations. Situations and normal human behavior in these situations can provide important clues to allow the computing systems to recognize human intentions and to predict what they will do next. Also, the systems may be able to clarify human desires by accepting the input of information both vocally and/or by physical interaction.

People move from one situation to another through physical action such as walking. When moving towards a situation, the user can retrieve information related to the situation that is being confronted. This can be an intuitive information-seeking process. Walking through real-world situations is a more natural way to retrieval information than searching within complex information spaces. Situated interaction can be considered as matching retrieval cues to real-world situations. For example, if a person wants to read a book, he or she naturally has the idea of going to a place where a bookshelf exists. This means that a situation that includes a bookshelf can be a retrieval cue related to searching for books.

5.3.2 Augmented Memory

One of the interesting functions of ubiquitous computing is the augmentation of human memory. The system stores summarized descriptions of the user's behavior in association with situations (including time) where the behavior occurred. A behavior description includes time, location (or ID), and related things (object IDs and/or human IDs). It may also contain visual/spoken information, if available.

Thus, human memory can be indirectly augmented by accumulating memory retrieval cues related to real-world situations. Human memories consist of mixtures of real-world situations and information that was accessed from those situations. Therefore, recognizing a real-world situation can be a trigger for extracting a memory partially matched with the situation and associating information related to the memory.

The context of an episodic memory provides lots of cues for remembrance. These cues include the physical environment of an event, who was there, what was happening at the same time, and what happened immediately before and afterwards [12]. This information helps us both recall events given a partial context and associate our current environment with past experiences that might be related.

5.3.3 Situated Content in Action

Ubiquitous computing gives the opportunity to bring new sensors and context-dependent information processing into everyday life, such that these pieces of information on the physical context can be used by the information providers to generate situated content.

Digital content becomes situated via transcoding in the ubiquitous computing environment. The following is a typical scenario where situated content works in our daily life.

A girl walking in the park with her boyfriend notices a monument and says, "What's that?" with an accompanying pointing action (Figure 5.1). This question is transfered into the cyberspace as a structured query (in XML format) with contextual information such as time, location, the girl's profile, and so forth.

A lot of related on-line resources are retrieved and connected with the query and then transcoded into a situated content that is concise and efficient for the user to understand considering the context. The result is transferred to the user's pervasive device in the most appropriate format (Figure 5.2).

The girl uses the device to see and hear the information of the monument. She notices that this information is an answer to the question that she presented just before now (Figure 5.3). The answer is not necessarily delivered to her own device. The information may be displayed on screen around there when it is available.

Figure 5.1 Example of situated content (Scene 1).

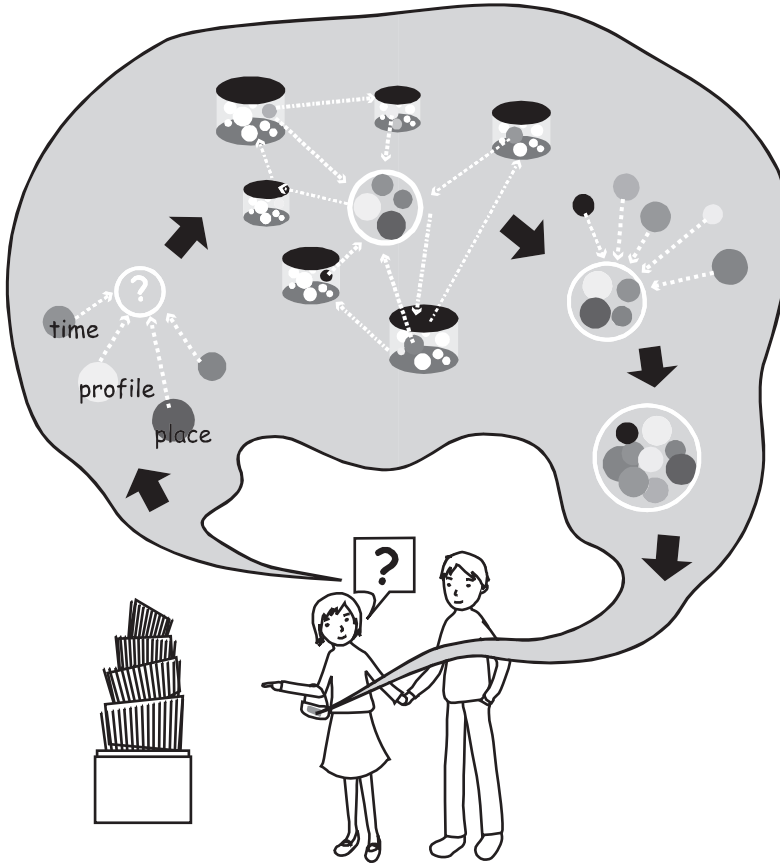


Figure 5.2 Example of situated content (Scene 2).

This typical scenario is also applied to advertisement of commercial products. However, it is effective if a user's intention is easily inferred by the user's behavior and physical circumstance; otherwise this type of information presentation can be just an annoying interruption.

Situated digital content is very efficient and informative for users who do not have a particular goal of information search in their daily life. So, it will be one of the good solutions of information overload.

5.4 Towards Social Information Infrastructures

The term *information infrastructure* refers to the communications networks and associated software that support interaction among people and

Figure 5.3 Example of situated content (Scene 3).

organisations. The Internet is the phenomenon that has driven the debate to date. The term ''information infrastructure'' is useful as a collective term for present networks (i.e., the Internet) and likely future facilities.

One of the specific goals of the social information infrastructure especially in government (so-called digital government) is to speed innovation, development, deployment, and application of advanced technologies into usable systems. Two existing examples help illustrate how this goal might be achieved.

1. Communication tools support and nurture linkages and relationships that were not possible through more formal means of communication. More and more often, we expect people, including government staff, to have and use e-mail. Discussion listservs and shared Web sites routinely connect distributed organizations

and virtual communities, fostering increased discussion and co-operation among those who share a common interest. Easy public availability of such information as government contracts or grant programs fosters greater equity and efficiency of government purchasing and the distribution of public resources.

These relatively ubiquitous capabilities are being augmented by more advanced collaborative tools in such areas as distance learning, just-in-time training, and anytime-anyplace meeting-ware. Use of the Internet and video conferencing techniques to deliver entire curricula from remote sites extends higher education and lifelong learning to many who would otherwise not be able to attend classes. Distance courses in specialized topics enable elementary and high school students to pursue studies unavailable to them in their home districts. Thanks to networked collaboration, these students can even conduct joint science experiments with their counterparts around the world. Although the pedagogic effectiveness of alternate modes of study and instruction are still being evaluated, it is clear that network-supported learning will play an increasingly important role in the future of education.

2. Information visualization, knowledge extraction, data integration, and digital library technologies have put the power of distributed information to useful social, scientific, and individual purposes. Data mining tools aid in identifying fraud and abuse in government programs. Data warehouses gather and integrate data from disparate sources, and data management and knowledge discovery tools are used to conduct planning and program evaluation in areas ranging from capital construction, to economic forecasting, to the performance of schools. Technologies such as data intensive computing environments facilitate the use of information from disparate heterogeneous databases. Digital library technologies are emerging to help users find and use information and records regardless of physical format or location.

   Today, the use of these advanced analysis tools varies considerably across agencies and levels of government, and it is too early to tell which applications will be most useful and adaptable. Applications of these technologies are limited today by at least three important considerations: poor or variable data quality, the willingness and ability of organizations to share information

across their boundaries, and, when applications involve information about people, threats to personal privacy.

Digital content technology largely contributes to these examples. In the following sections I discuss major issues for constructing a true social information infrastructure.

5.4.1 Content Management and Preservation

An increasing number of important government records are now stored exclusively in digital media. Many of these records contain multimedia formats, and they are often associated with automated workflow and electronic document repositories. Depending on the circumstances, informal information such as e-mail messages may be part of an official government record. Few guidelines now exist for effectively managing digital public records, yet their numbers grow dramatically every month. Preservation of electronic records is a particular challenge, as the media, software, and hardware used to create records and maintain them for active use are replaced with new generations every few years. Ironically, while the records of the seventeenth and eighteenth centuries remain readable today, our own generation of records is rapidly disappearing due to technological advances. At the same time, government archives are increasingly trying to accommodate the digitization of historical records in order to make these holdings more widely accessible to more users.

Finally, with the increasing availability of information in electronic form, it is becoming easier to use information for purposes beyond the original reason for its collection. Yet most government records systems are created without regard to the needs or preferences of secondary users, whether they are in different units of the same agency, in other organizations, or are future users whose interests come into play long after the records have served their primary purpose. More extensive research into archives and records management theory and practice are needed to resolve these issues.

Technologies based on XML, Java, and TCP/IP may present a better solution, but this is not yet proven.

5.4.2 Security Mechanisms

The exchange of information through a network is not a new phenomenon in government. In the past, telecommunications was accomplished using

dedicated point-to-point connections between pairs of agencies, or through secure value-added networks. TCP/IP networks are now replacing these facilities. The use of these Internet protocols facilitates communication between partners because only a single connection needs to be maintained to communicate with all partners on the network. However, the communication channel must retain properties that duplicate those found in earlier modes of communication: secure and private communication, authentication of messengers, integrity of messages, and stability of the network. One way to achieve this goal is to create a separate network, closed to all but trusted communicators. This model works for certain types of transactions, but since government agencies often work closely with many other organizations, a more affordable and open solution is needed.

At present, there are no commonly implemented models of security architecture that provide a trusted basis for electronic interactions. The array of issues, and the limited choices of technologies and strategies has led to very slow progress in deploying these architectures. In such an environment, it is not surprising that issues of security dominate much of the discussion in government about networking.

Security annotation and transcoding may present a new solution to this issue.

5.4.3 Human Interfaces

The standard user interface and the World Wide Web browser have done much to extend useful computing to every area of our society. The standard interface, commonly based on Microsoft Windows, flattens the learning curve needed for each new application. The Web browser's ease of use and widespread public acceptance have led many agencies to use this technology in direct public contact. One attractive feature of the Web is its ability to integrate information and services from separate organizations into a single user presentation. This technique has been used to develop Web sites that serve as a portal to all that a government unit has to offer. Today, most of these sites are limited to a single level of government, and do not represent true integration of services. Instead, they typically provide an extensive table of contents of many agency programs and services. However, many government agencies have begun reorienting their Web services around the needs of users rather than around their organizational structures.

Further advances in human interfaces are likely to focus both on simplicity and increased power. Digital library technologies, for example, will

put the power of multiple databases to more effective uses. Information visualization technologies allow users to manipulate large data sets to get a better understanding of the information they contain. Research into the interaction between people and machines, including speech recognition/synthesis and computer vision/graphics, will likely lead to innovations in the way people perceive and use the information environment.

Customizable human interfaces with digital content are realizable with technologies of semantic annotation and transcoding.

5.4.4 Scalable Systems

The models and processes for designing and developing large, complex systems have advanced much less than the specific technologies they might employ. While all organizations face this issue, the development of large government information systems face special challenges that lead to an especially risk-prone environment. Typically, a significant number of participants and organizations have a stake in the system. This may be due to the innate complexity of the underlying program or existing systems, to legislative mandates, or because a large number of organizations play a role in the system development process. In such an environment, it is very difficult to maintain consistent approaches to architecture, data definitions, data collection, information quality, data integration, and overall system functionality.

These complications add time, cost, and complexity to the development life cycle. As a consequence, design and implementation may take years, conflicting directly with the rapidity of technological change. By the time they are completed, the best technologies for the job may well have changed. For example, the recent redesign of the air traffic control system in the United States by the *Federal Aviation Administration* (FAA) was begun before the widespread commercialization of the GPS. Such major technology shifts can cause wholesale changes to system design in the middle of the development process.

Existing software development models such as the waterfall and spiral models do not deal explicitly with these kinds of changes. Prototyping, while very useful in some projects, seems to have less utility in dealing with the complexities of these large systems with their enormous interoperability issues, and long development times.

An essence of digital content technology is interoperability and versatility provided by description standards and reuse/integration of classic techniques such as language, speech, and video analyses. When one

technology shifts to another, a new method for annotation and transcoding may need to be invented. The total system, however, will not change but be enhanced incrementally, since digital content infrastructure has a layered structure as presented in Chapter 1.

5.5 Final Remarks

Digital content technology presented in this book will contribute to future information businesses, personal communication systems, ubiquitous computing systems, and social information infrastructures. It will make people and machines more knowledgeable and thoughtful so that people can make good decisions promptly in both personal and social situations.

It will also change our lifestyle to be more creative and active, since we will not have to do tedious office tasks that can be done by machines, and we will do more knowledge-intensive work utilizing intelligent and situated content. The work remaining for humans will be that which is impossible to be fully automated.

References

[1] The Content ID Forum, "Effective and Smooth Distribution of Digital Content," http://www.cidf.org/english/, 2002.

[2] Kudo, M., and S. Hada, "XML Document Security Based on Provisional Authorization," *Proc. 7th ACM Conference on Computer and Communication Security (CCS'00)*, 2000, pp. 87–96.

[3] Maes, P., "Agents That Reduce Work and Information Overload," *Communications of the ACM*, Vol. 37, No. 7, 1994, pp. 30–40.

[4] Etzioni, O., and D. Weld, "A Softbot-Based Interface to the Internet," *Communications of the ACM*, Vol. 37, No. 7, 1994, pp. 72–76.

[5] Resnick, P., and H. R. Varian, "Recommender Systems," *Communications of the ACM*, Vol. 40, No. 3, 1997, pp. 56–58.

[6] Nagao, K., "Agent Augmented Reality: Agents Integrate the Real-World with Cyberspace," in T. Ishida, (Ed.), *Community Computing: Collaboration over Global Information Networks*, New York: John Wiley & Sons, 1998.

[7] Bajura, M., H. Fuchs, and R. Ohbuchi, "Merging Virtual Objects with the Real-World: Seeing Ultrasound Imagery Within the Patient," *Computer Graphics*, Vol. 26, No. 2, 1992, pp. 203–210.

[8] Feiner, S., B. MacIntyre, and D. Seligmann, "Knowledge-Based Augmented Reality," *Communications of the ACM*, Vol. 36, No. 7, 1993, pp. 52–62.

[9] Finin, T., et al., *Specification of the KQML Agent-Communication Language,* Technical Report EIT TR 92-04, Enterprise Integration Technologies, 1992.

[10] Rhodes, B. J., and T. Starner, "Remembrance Agent: A Continuously Running Automated Information Retrieval System," *Proc. 1st International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology (PAAM'96)*, 1996, pp. 487–495.

[11] Want, R., et al., "The Active Badge Location System," *ACM Transactions on Information Systems*, Vol. 10, No. 1, 1992, pp. 91–102.

[12] Tulving, E., (Ed.), *Elements of Episodic Memory*, Oxford, England: Clareandon Press, 1983.

About the Author

Katashi Nagao received a B.E., an M.E., and a Ph.D. in computer science from the Tokyo Institute of Technology in 1985, 1987, and 1994, respectively. Since 1987, he has been researching natural language processing and machine translation systems at IBM Research, Tokyo Research Laboratory. In 1991, he started conducting the research projects on natural language dialogue, multiagent systems, and human-computer interaction at Sony Computer Science Laboratories, Inc. From 1996 to 1997, he was a visiting scientist at Beckman Institute for Advanced Science and Technology at the University of Illinois at Urbana-Champaign. He rejoined IBM Tokyo Research Laboratory and started the semantic transcoding project in 1999. He is currently a professor at Nagoya University, Nagoya, Japan. He is continuing research on digital content technology.

For further information, about updating, correcting, or related software, please visit the Web site at http://www.nagao.nuie.nagoya-u.ac.jp/books/DigitalContent.html.

Index