An Authoring Tool for Customizable Documents

by

Kimberley J. Parsons

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1997

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Kimberley J. Parsons

I further authorize the University of Waterloo to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Kimberley J. Parsons

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

I describe the design and implementation of an authoring tool for customizable documents. The authoring tool was written as part of the HealthDoc project at the University of Waterloo, a project that aims to generate health-education materials that are tailored to the medical condition and personal situation of an individual patient. The authoring tool described in this thesis would be used by a medical writer to create documents that can be customized for individual readers.

The HealthDoc authoring tool will be used to specify variations of a single document and the discourse structure of the document. The author enters ordinary English text and indicates the conditions under which each piece of text is relevant for a particular type of patient. The author also includes in the document supplemental linguistic information that is used by the rest of the HealthDoc system to perform linguistic "repair" of the document, to ensure that the final document is coherent and rhetorically effective. The linguistic information currently used by HealthDoc is rhetorical relations and coreference. The authoring tool provides facilities to indicate rhetorical relations between pieces of text and to indicate groups of words and phrases that are coreferential.

While the authoring tool was designed for the HealthDoc project, it could be used to author customizable documents on any topic. I explain the steps performed by an author to create a customizable document and I document the process a computer programmer would use to design an authoring tool for a given application. I also provide a list of recommended enhancements to the authoring tool that could be used in alternative designs or in subsequent implementations.

Acknowledgments

I acknowledge the financial support of IBM, the Information Technology Research Center, and the University of Waterloo and thank them for their generosity. I would also like to thank my family for always supporting me and believing in me.

Contents

List of Figures

Chapter 1

Introduction

In this thesis I describe the design and implementation of an authoring tool for customizable documents. The authoring tool was developed as part of the HealthDoc project at the University of Waterloo. It is used to specify variations of a single document and the discourse structure of the document.

Using the authoring tool, the author enters ordinary English text and indicates the conditions under which each piece of text is relevant for a particular patient. The author also includes in the document supplemental linguistic information that is used by the rest of the HealthDoc system to perform linguistic "repair" of the document, to ensure that the final document is coherent and rhetorically effective.

This introductory chapter provides background knowledge about the HealthDoc project, which defines the scope of the authoring tool I developed. Section 1.1

explains the goals of the HealthDoc project, and then section 1.2 describes the process by which HealthDoc generates customized health-education brochures.

Section 1.3 describes the HealthDoc *master document* that the author creates using the authoring tool. This section includes a description of the content of the master document and explanations of the types of supplementary linguistic information included in the master document.

Section 1.4 then provides an outline of the rest of the thesis.

1.1 The HealthDoc Project

The HealthDoc Project (DiMarco, Hirst, Wanner, and Wilkinson 1995, Hirst and DiMarco 1996) is a research project at the University of Waterloo that aims to generate health-information brochures tailored to the medical condition and personal characteristics of the individual patient. The project uses natural language generation techniques to automatically generate multiple versions of a similar document. By tailoring information to the individual patient, the HealthDoc system will provide doctors with an effective way to affect patient compliance with suggested medical regimes, lifestyle, and so on.

Studies have shown that health information that is specifically tailored to a patient's medical condition and personal situation is much more effective than generic information. For example, experiments in a family practice clinic in North Carolina using customized smoking-cessation brochures produced a cessation rate of 20.8%, which was 2.8 times the cessation rate of the control group (7.4%) (Strecher et al. 1994). Similar studies have shown that if patients receive a tailored educational brochure, they are more likely to remember receiving the brochure (Campbell et al. 1994), more likely to read most of the content of the brochure (Skinner, Strecher, and Hospers 1994), and more likely to heed its recommendations (Strecher et al. 1994).

1.2 Generation by Selection and Repair

One of the main research contributions of the HealthDoc project is a new model of natural language generation, based on 'generation by selection and repair'. Lending support to this model is the hypothesis that it is easier to rewrite than to write. Instead of attempting to generate a document from scratch, HealthDoc will start with an existing (but unpolished) draft document, and revise it.

Figure 1.1 shows the structure of the HealthDoc system, which I will explain in the rest of this section.

In order to create tailored health-information brochures, the HealthDoc project selects pieces of text from a pre-written *master document* that contains all of the

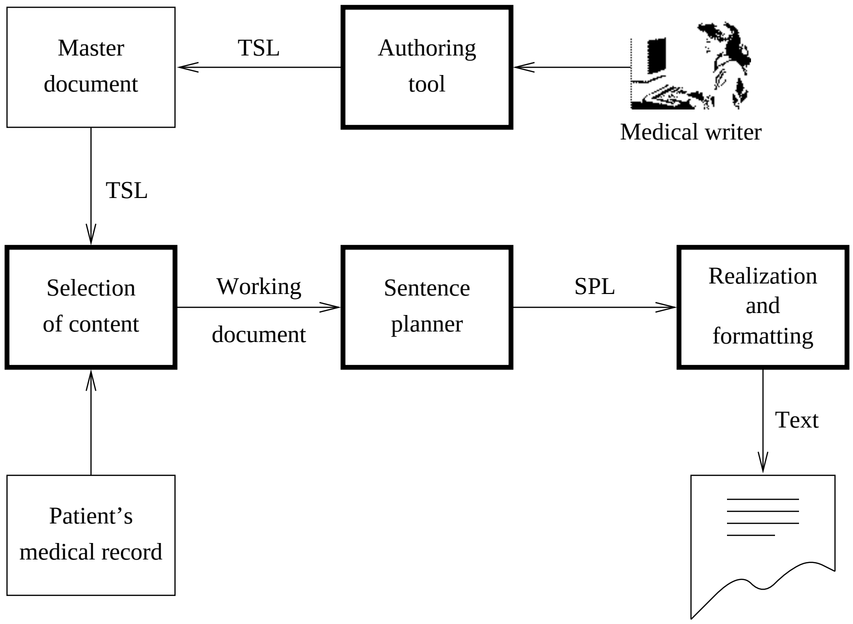


Figure 1.1: Generation by selection and repair in the HealthDoc system. Boxes with heavy lines represent processes, and boxes with light lines represent sources of information; the arrows represent flow of information.

information one might wish to say to any patient. Each piece of information in the master document is marked to indicate the types of patients for whom it is relevant. The master document is authored by a medical professional, and the document is represented in Text Specification Language (TSL) (see section 1.3.3).

Pieces of text are selected from the master document on the basis of information contained in the patient's medical record and assembled into a *working document* for the patient. This working document contains only the information relevant to the individual patient.

However, simply pasting pieces of text together is unlikely to result in a coherent, smooth document, unless the author of the master document painstakingly ensures that every possible combination of texts is coherent and smooth. When Strecher et al. (1994) prepared their smoking-cessation brochures, they discovered that as they added more snippets of text to the system, the task of ensuring that every possible combination of texts was coherent became more and more difficult (Victor Strecher, p.c.). In the system they were using, there were over 55 000 possible versions of the brochure (Strecher et al. 1994). The HealthDoc system relieves the author of the burden of ensuring that every possible combination is coherent by automatically repairing the text of the working document.

This repair of the working document is performed by the *sentence planner*. The sentence planner takes as input the text of the working document and performs sophisticated linguistic repair of the text, such as reducing redundancies, performing aggregation (e.g., by combining sentences), and repairing references (e.g., by replacing a repetitive phrase by a pronoun).

1.2.1 Examples of Text Repairs

One type of repair that the sentence planner performs is aggregation, which is combining two sentences into one. For example, consider sentences (1) and (2) below.

(1) The condition that you have is non–insulin-dependent diabetes.

(2) Non–insulin-dependent diabetes is the most common type of diabetes.

The sentence planner will reduce repetition by combining these two sentences into one, resulting in sentence (3).

(3) The condition that you have is non–insulin-dependent diabetes, which is the most common type of diabetes.

Another type of sentence repair is reference repair. The sentence planner can replace repetitive or redundant phrases with pronouns or can re-word the phrase. For example, consider sentences (4) and (5) below.

(4) Insulin-dependent diabetes is the less common type of diabetes.

(5) Insulin-dependent diabetes most often develops in children and young adults.

The sentence planner could replace *insulin-dependent diabetes* in sentence (5) with *this condition*, which changes sentences (4) and (5) to sentences (6) and (7) below.

(6) Insulin-dependent diabetes is the less common type of diabetes.

(7) This condition most often develops in children and young adults.

The sentence planner outputs the repaired document in Sentence Plan Language (SPL) (see section 1.3.3), which is a semantic representation of the text. The SPL is used as input to the Penman text generator (Penman Natural Language Generation Group 1989), which converts the SPL into English sentences ("realization" in figure 1.1). The repaired document is then formatted and printed, and given to the patient.

By this process, then, the HealthDoc system tailors an existing document by selecting the relevant pieces and then repairing the resulting document to restore coherence and cohesion.

1.3 The Master Document

The process of creating a master document is much more complex than creating an ordinary document, so master documents must be carefully authored by an appropriate expert. A HealthDoc master document would be written by an author with medical expertise, such as a medical writer, a nurse, or a doctor. The author is not assumed to have any knowledge of the inner workings of the HealthDoc system.

1.3.1 The Content of a Master Document

In the HealthDoc master document, the author must include all of the possible variations of each piece of text, and must specify the conditions under which each variation is relevant. For example, in a master document about smoking cessation, information about smoking and pregnancy is relevant only to female patients, while information about the effect of second-hand smoke on young children is relevant to both male and female patients, but only those with young children. The author must mark the information about smoking and pregnancy to indicate that it is relevant only to female patients (or only to female patients who are pregnant), and must mark the information about the effect of second-hand smoke on children to indicate that it is relevant to both male and female patients, but only those with young children.

In order to perform linguistic repair of the working document, the sentence planner requires supplemental linguistic information in addition to the English text and the conditions for when the text is relevant. Thus, the author must also include this linguistic information in the master document. The sentence planner currently uses rhetorical relations and coreference information (see section 1.3.2), but other types of linguistic information are likely to be needed in future versions of HealthDoc.

1.3.2 Supplementary Linguistic Information

The HealthDoc sentence planner currently uses rhetorical relations and coreference information to repair the text.

Rhetorical Relations

Rhetorical relations are semantic relations between clauses, sentences, or groups of sentences. Such relations include elaboration, evidence, and sequence.[1] As an example, consider the following two sentences (Mann and Thompson 1986):

(8) The program as published for calendar year 1980 really works.

[1]Figure 5.5 shows more of the possible rhetorical relations.

(9) In only a few minutes, I entered all the figures from my 1980 tax return and got a result which agreed with my hand calculations to the penny.

Sentence (9) is *evidence* for sentence (8).

The sentence planner uses these relations to arrange and order the text according to a model of rhetoric known as Rhetorical Structure Theory (Mann and Thompson 1986). The authoring tool allows the author to specify the rhetorical relations that hold between specific sentences or groups of sentences in the master document.

Coreference

Words or phrases that refer to the same object or concept are called *coreferential.* Pronouns are one example of coreference, as are phrases that re-word a concept. For example, in the sentences

(10) You have type II diabetes.

(11) This condition is most common in adults over age 40.

the phrases *type II diabetes* and *this condition* are coreferential, since *this condition* is a reference back to the medical condition *type II diabetes.*

There are different types of coreference (e.g., superordinate, specific, generic), and the authoring tool allows the author to specify which type of coreference is

```
'(asc / ascription
  :focus diab6
  :tense present
  :domain (cond / abstraction
           :specific cond0
           :lex condition
           :determiner the
           :process (have / ownership
                     :lex have-possession
                     :tense present
                     :domain (hearer / person)
                     :range cond))
  :range (diab6 / abstraction
           :specific diab4
           :lex diabetes
           :determiner zero
           :property-ascription (ins / quality
                                 :lex insulin-dependent)))
```

Figure 1.2: Representation of the sentence *The condition that you have is insulin-dependent diabetes* in Sentence Plan Language.

involved. The sentence planner uses knowledge of coreferential words and phrases to reduce repetition and to perform aggregation in the working document.

1.3.3 Internal Semantic Representations and File Formats

Once the master document is complete, the English sentences of the document are semi-automatically converted to an internal representation called Sentence Plan Language (SPL) (Banks 1997, DiMarco and Banks 1997). SPL is a semantic representation of the English text, and is used as input to the KPML system (Bateman

1995), a derivation of the Penman surface text generator (Penman Natural Language Generation Group 1989). An example sentence represented in SPL is given in figure 1.2. This SPL example generates the sentence *The condition that you have is insulin-dependent diabetes.*

The SPL for each sentence, the rhetorical relations, coreference relations, and sentence ordering are stored in an internal HealthDoc representation written in Text Specification Language (TSL). TSL is a specification language for master documents that indicates textual and non-textual variations at all levels of document granularity, the conditions for selecting variations, and additional information (e.g., rhetorical relations and coreference links). Figure 1.3 contains part of the TSL for a master document about diabetes. This TSL example shows a set of two variations, each of which contains two sentences. One variation is for patients with insulin-dependent diabetes and the other is for patients with non–insulin-dependent diabetes.

Master document files created by the authoring tool are saved in TSL format.

1.3.4 The HealthDoc Authoring Tool

The authoring tool described in this thesis was designed to support the creation of customizable documents for the HealthDoc project. It is the user interface used by

```
(variation-set
          :name 'variation-set1
          :set-of-variations '(variation1 variation2))
(variation
          :name 'variation1
          :condition "(= diagnosis insulin-dependent-diabetes)"
          :sentence-list '(sent1 sent2))
(variation
          :name 'variation2
          :condition "(= diagnosis non-insulin-dependent-diabetes)"
          :sentence-list '(sent3 sent4))
(sentence
          :name 'sent1
          :english "The condition that you have is
                    insulin-dependent diabetes."
          :spl ( ... the SPL goes here ... ))
(sentence
          :name 'sent2
          :english "Insulin-dependent diabetes is the less common
                    form of diabetes."
          :spl ( ... the SPL goes here ... ))
(sentence
          :name 'sent3
          :english "The condition that you have is
                    non-insulin-dependent diabetes."
          :spl ( ... the SPL goes here ... ))
(sentence
          :name 'sent4
          :english "Non-insulin-dependent diabetes is the most
                    common type of diabetes."
          :spl ( ... the SPL goes here ... ))
```

Figure 1.3: Variation set containing two variations in Text Specification Language.

the medical writer to input the master document; it provides facilities for entering the text of the master document, specifying the conditions under which each piece of text is relevant, and specifying relations (rhetorical, coreferential, and ordering) between pieces of text. The author enters ordinary text, but the authoring tool reads and writes files in TSL format.

While this authoring tool was originally designed for the HealthDoc project, it is not limited to health-information brochures, and can be used to author customizable documents on any topic.

1.4 Outline

The second chapter of this thesis discusses the nature of documents in general, and the advantages of customizable documents. This chapter also describes the results of previous research on the effects of tailored documents in the health-care field.

The third chapter defines the concept of an authoring tool, and discusses different models of authoring. It presents a continuum of software tools for authoring text, and a number of paradigms used for authoring in various domains. This chapter also includes a survey of commercial, freeware, and academic authoring tools for text, hypertext, HTML, and multimedia applications.

The fourth chapter analyses the requirements for an authoring tool for customizable documents, with an emphasis on authoring customizable health-information brochures.

The fifth chapter presents the design of the HealthDoc authoring tool, including the design decisions made and descriptions of the functionality of the software.

The final chapter gives an evaluation of the authoring tool and summarizes its research contributions. It also presents suggestions for future work and possible extensions of the authoring tool.

Chapter 2

The Case for Customization

In this chapter I discuss the advantages of documents that are customized for an individual reader. The chapter begins with an explanation of the weaknesses of non-tailored (generic) documents. Next, I discuss some different definitions of a 'document' and the implications of each definition for customization. Then, having explained why we might want to customize documents, I review some studies in health-education which used customized brochures to positively affect patients' behaviour and attitudes. I then give some examples of fields other than health-education that could benefit from customization, and I list some possible ways that documents could be tailored to improve their effectiveness.

2.1 The Nature of Documents

2.1.1 The Weaknesses of Generic Documents

Until recently, documents have been linear representations of information[1]. They have been designed and written to be read in a particular order. Countless English teachers have reinforced this idea: when writing, one starts with an introduction, elaborates on an idea, and then ends with a conclusion and summary.

However, in many cases, a reader cannot simply start at the beginning of a document and read through to the end. For example, if the document is a software manual, the reader will not have time to read the entire manual. He will skip to the part that is most likely to contain the information he needs.

Unfortunately, a reader who skips into the middle of a document may be confused, because he is missing information that was presented earlier in the document. Because writers are aware of this potential problem, they are forced to repeat information in different sections of the document, so that it *is* possible to jump around

[1]The definition of 'document' is changing as non-linear documents such as hypertext documents gain popularity. Hypertext is *not* designed to be read linearly; a defining characteristic of hypertext is that the document can be read in whatever order the reader desires.

Most hypertext documents are still generic documents. However, research projects in adaptive and flexible hypertext are creating hypertext documents that are customized for individual readers.

For example, the HealthDoc project is also investigating customized hypertext, in the form of customized Web pages (DiMarco and Foster 1997). The authoring tool described in this thesis could be used to author both regular documents and hypertext documents.

in the document. But repeating the same information in several places forces those readers who *do* read the document linearly to read repetitive information.

The author of the document faces a trade-off between including enough repetitive information to allow the reader to "enter" the document at *any* point and providing a document that is still readable when read in order, cover-to-cover.

A second dilemma faced by authors is deciding the amount of information to include. If the author tries to provide a comprehensive information source, he will end up including information that is irrelevant to many readers, so that readers will be tempted to jump around in the document, searching for that information that is relevant to them. Even in a shorter document that can be easily read in its entirety, too much irrelevant information may cause the reader to conclude that the document is not applicable to him, so he will ignore its contents, even though parts of the document are applicable.

On the other hand, if the author includes only the information that is relevant to all readers, he may be leaving out too much detail. The reader might get the impression that the document is only a surface treatment of the subject and then dismiss it as superficial and inadequate.

So, by including either too much or too little information, the author risks the document being ineffective for some or all readers.

If an author could include in a document exactly those topics that are relevant to a particular reader, no more, no less, then he would be able to communicate more effectively with each reader. An ability to produce customized documents would give an author a means to increase the relevance of the document to a particular reader and thereby communicate more effectively, with a greater likelihood of influencing the reader.

2.1.2 So What Exactly is a "Document"?

In a discussion of "documents", customized or otherwise, it is useful to understand the nature of documents. What are the defining characteristics of a document?

The Two Views of a Document

Levy (1988) defines a document as a "communicative artifact", a definition that encompasses two different definitions of a document. First of all, a document is an "artifact": it is a piece of paper or a book; it is tangible; it exists in physical reality[2]. Secondly, a document is "communicative": it is constructed to serve a communicative function; it is composed of elements or modules that are designed and ordered to contribute to some communicative goal; it is a collection of information organized according to some plan.

[2]This definition can be stretched somewhat to include the electronic storage of documents, such as word-processor files or hypertext documents.

The Document as a Physical Artifact

If we consider a document to be a physical artifact, there is a large amount of software to support the creation of documents. All manner of text processors and word processors assist an author in constructing a document. They provide the ability to save a document in electronic form or to print the document on paper. They allow the author to manipulate the physical characteristics of the document, such as margins, fonts, or colours.

Customizing a document can be performed by changing the physical characteristics of the document. Different readers could receive documents in different fonts (e.g., larger fonts for seniors), documents with different illustrations, or documents in different paper formats (e.g., DIN A4 paper in Europe or 8.5" $\times$ 11" in North America). Current authoring software (e.g., a word processor) provides support for managing and manipulating documents as physical artifacts.

The Document as a Collection of Information

In contrast, if we consider a document to be a collection of pieces of information, organized according to a communicative plan, then there is a lack of software to support the creation of documents. While some software (e.g., more-advanced word processors) assist the author in organizing the structure of the document (e.g., by using templates for different document types or applying formatting styles for different levels of topic headings), there are very few software tools that support the collection and organization of pieces of information into a coherent document.

The commercially available authoring tools do not assist an author in managing pieces of information or in fulfilling a communicative plan. This planning is performed informally by the author, using a type of folk knowledge acquired through experience. The authoring tool described in this thesis provides facilities for formally organizing the rhetorical structure of a document, and focuses on the pieces of information in the document rather than on the physical characteristics of the document.

Customizing a document can be performed by customizing the pieces of information that an individual reader receives. This type of document tailoring requires software that allows the author to specify the types of readers for whom a piece of text is relevant. The authoring tool described in this thesis provides this functionality.

The Document as a Database

If a document is considered to be a collection of modules of information, the line between document and database becomes blurred. A document can be considered to be a repository of information, organized in a particular fashion, according to a communicative goal. Pieces of the information could be re-used, however, to achieve other communicative goals by re-arranging the order of the modules and filling in the gaps as necessary.

In order to re-use the modules of information, the author must somehow index the modules so that they can be retrieved. The author must, in essence, create a database of information modules, indexed to allow effective re-use.

One way that the modules could be indexed is by their function in the rhetorical structure of the document. Another method of indexing could be the type of reader for whom they are relevant. By combining these two methods of indexing, the author can create customized documents that are both relevant and rhetorically effective: the document for an individual reader would be composed of those information modules that are most relevant to that reader *and* the modules would be organized according to a communicative plan.

The authoring tool described in this thesis does not allow the author to create an actual database of pieces of text, but it does combine customization and rhetorical structure, so that an author can create a tailored document that is organized according to some communicative plan.

2.2 The Advantages of Tailored Documents: Some Success Stories

It can be a great benefit to the reader if a document provides only the information desired or needed, instead of also including irrelevant information. A tailored document is more pertinent for the reader, so he is much more likely to recall receiving the document, and much more likely to read the entire document (Skinner et al. 1994, Campbell et al. 1994).

In addition to reading more of the document, recipients of tailored documents are also more likely to change their behaviour as a result of reading the document. Previous research has shown that health information that is specifically tailored to a patient's medical condition and personal situation is much more effective than generic information in producing changes in health behaviour. For example, experiments at a family practice office in North Carolina using customized smoking-cessation brochures produced a cessation rate of 20.8%, which was 2.8 times the cessation rate of the control group (7.4%) (Strecher et al. 1994).

Customized documents can present in clear detail only the information that is necessary, and can maximize persuasiveness for the intended reader. The author can concentrate on a few key topics, instead of touching on many irrelevant topics. This reduces the sheer length of generic documents, thereby improving the likelihood that a reader reads the entire document.

Tailored documents also allow the author to take advantage of some of the strengths of face-to-face communication. For example, by tailoring an educational message for an individual patient, a doctor can address the patient's health beliefs, current health practices, and factors influencing health behaviour.

In the next sections, I will discuss some studies in health communication that have used customized health information to positively affect patients' behaviour or attitudes.

2.2.1 Smoking Cessation

In two studies in North Carolina, Victor Strecher and his colleagues evaluated the effects of tailored pamphlets on smoking cessation (Strecher et al. 1994). They sent customized letters to cigarette smokers who were identified at family practice medical clinics. The letters were tailored according to the patient's cigarette consumption, interest in quitting, and perceived benefits and barriers to quitting.

In the first study, one group received tailored letters, and the control group received no letters. The light-to-moderate smokers in the group that received the tailored letters had a cessation rate of 30.7%, while the control group had a cessation rate of only 7.1%. Heavy smokers in the tailored group had a cessation rate of 9%, compared to 7.7% in the control group (not a statistically significant difference). When both the light-to-moderate smokers and heavy smokers were grouped together, the cessation rate was 20.8% in the tailored group and 7.4% in the control group.

In the second study, one group received tailored letters, and the control group received generic health letters. In this case, the light-to-moderate smokers who received tailored letters had a cessation rate of 19.1% and the light-to-moderate smokers in the control group had a cessation rate of 7.3%. The heavy smokers who received tailored letters had a cessation rate of 3.9%, compared to 9.8% of those who received generic letters.

Strecher et al. explain that the heavy smokers probably did not respond as well as the light-to-moderate smokers because heavy smokers need more than just a persuasive argument to quit smoking: they need medical assistance to overcome a physical addiction.

Nevertheless, the results of these studies are very encouraging: in moderate and light smokers, tailored letters produced a cessation rate over four times that of no letter (Study 1) and a cessation rate approximately three times that of generic letters (Study 2). It is also interesting to note that there was no significant difference in the cessation rates of those who received generic letters and those who received no letter, which highlights the ineffectiveness of non-tailored materials.

2.2.2 Mammography Screening

Skinner, Strecher, and Hospers (1994) describe another study at the University of North Carolina in which women received tailored letters addressing individual beliefs and barriers associated with mammography, as well as the risk of breast cancer and the individual's screening status.

The letters contained information specific to the individual, but were worded as if they were written for a general audience, so that the results could be compared with a generic letter. The pictures in the letter were tailored by race, and the captions were tailored according to the individual's current health behaviour.

The women who received tailored letters were more likely to remember receiving the letters than recipients of generic letters. Recipients of tailored letters were also more likely to have read more of the content. Fifty-three percent of those who received tailored letters read "most or all" of the content, while only forty percent of those who received generic letters read "most or all" of the letter.

2.2.3 Improving Dietary Behaviour

In a third study at the University of North Carolina, tailored letters were sent to patients to encourage a decrease in fat intake and an increase in fruit and vegetable intake. Thirty-two percent of those who received a tailored letter reduced their fat intake, compared to nine percent of those who received a generic letter and three percent of those who received no letter (Campbell et al. 1994). There was no significant increase in fruit and vegetable intake in any group.

Of those who received a tailored letter, 73% remembered receiving the letter, while only 33% of those who received a generic letter did.

2.2.4 Asthma Treatment

In a study on health education in asthma treatment, Osman et al. (1994) present the impressive results of customized brochures sent to patients with severe asthma. In a year-long study of 801 adult asthma patients, the series of four tailored booklets reduced hospital admissions by 54%.

The patients received the first booklet at the clinic, and the following three were sent by post. The booklets were customized by the name of the prescribed drug, a coarse-grained user model (e.g., smoker or non-smoker), and questions from a patient-reply questionnaire. The booklets were produced using a spreadsheet and mail-merge software to piece together blocks of text for each patient.

2.2.5 Explaining Migraine

Buchanan et al. (1992) describe an interactive program to educate patients about the treatment of migraine headaches. The system generates explanations and gives information about the disease, possible therapies, and medications. The output of the program is generated dynamically, on the basis of the patient's questions, a user model of the patient, and the current discourse context.

The system's responses are tailored according to the patient's medical record (e.g., names of prescribed drugs), the patient's characteristics as collected in a pre-visit survey (e.g., symptoms, past treatments), the dialogue history (e.g., user's utterances, system's previous responses), the user's goals (inferred from utterances and questions), and the dialogue's focus history (the objects that were in focus at each turn in the dialogue) (Carenini and Moore 1993). The explanations are generated using schemas for each type of explanation that the patient can request. For example, if the patient chooses "Describe" and the name of a drug, a schema for describing drugs is filled in. If the patient chooses "Compare" and two drugs, a schema for comparing two drugs is filled in.

In a preliminary evaluation of the patients' assessments of the program (Carenini, Mittal, and Moore 1994), 100% of the patients liked using the program, 94% said that the program told them something they didn't already know, and 75% learned something that they would not have asked their doctor.

Results of the study show that patients definitely prefer to hear advice from their own doctor: when the voice was familiar, 70% rated the presentation as "excellent"; when the voice was unfamiliar, only 45% rated the same presentation as "excellent". Thus, simply tailoring the presentation by changing the voice on the tape increased patient satisfaction by 35%.

2.2.6 Explaining the Medical Record

The PIGLET (Personalised Intelligent Generator of Little Explanatory Texts) project (Binsted, Cawsey, and Jones 1995) at the University of Glasgow provides patients with tailored explanations of their medical record. The explanations are presented as interactive electronic hypertext.

PIGLET reads the patient's medical record, and assumes that the patient will be interested in all of the issues listed in the record. The list of issues is presented to the patient as an initial list of issues that can be explored. The patient selects which issues he would like to investigate, and PIGLET generates an explanation.

The system's answers are tailored on the basis of broad assumptions about the patient's likely knowledge and concerns. The user categories are based on the patient's major chronic problem and his knowledge about the disease. For example, a patient who has had diabetes for several years likely already understands basic terms like “insulin”, whereas a patient who has just been diagnosed will need to have such terms explained. When describing an object or concept, PIGLET tries to find a description given in terms of related objects or concepts that the patient already knows.

PIGLET’s text generation is performed using schemas to select and organize the content and templates to do sentence-level generation.

Preliminary evaluation of the system by seven diabetes professionals and ten diabetes patients was very positive. The health professionals thought the system would be useful, and the patients felt that the explanations were helpful and relevant (Cawsey, Binsted, and Jones 1995).

2.2.7 The Disadvantages of Tailored Documents

One difficulty that Strecher et al. encountered when writing the customized brochures was keeping track of all of the possible brochures that could be produced and ensuring that all of the possible brochures were coherent. The brochures were created by selecting snippets of text from a library of text and graphics and then using a word processor to compose the results. Over 55 000 different versions of the letter were possible (Strecher et al. 1994).

As the number of text snippets in the library increased, it became more and more difficult to ensure that all possible combinations of texts were coherent and cohesive. For example, the brochures produced tend to be very repetitive (e.g., the paragraph openings are all the same) (Victor Strecher, p.c.). The HealthDoc system would solve this problem by performing repair after the snippets had been selected, so that the author doesn't need to ensure in advance that all possible combinations of texts are coherent.

2.2.8 The Next Step...

The tailoring systems discussed above all tailor the *content* of the document. They choose information to include or exclude on the basis of a user model of the patient (e.g., smoker or non-smoker). Some of the systems (e.g., the Migraine Project) also attempt to tailor the document on the basis of the current discourse context. The document is tailored on the basis of the dialogue's focus history and on the user's goals.

However, none of the systems discussed attempt to tailor the style of the document, to produce different rhetorical effects. Since the long-term research goal of the HealthDoc project is to investigate theories of text composition (e.g., computational models of rhetorical structure, lexical semantics), the HealthDoc system will attempt to adjust the syntactic structures of a document to convey different stylistic effects (DiMarco and Hirst 1993a) and will investigate the role of near-synonymy and lexical choice in document composition (DiMarco and Hirst 1993b, DiMarco, Hirst, and Stede 1993, Hirst 1995).

One reason that the other tailoring systems do not attempt to customize the style of the documents is that the task is simply too complicated, given the types of text planning being used. In systems that compose snippets of text to create the final document, it is extremely difficult and time-consuming to ensure that every possible combination of texts is coherent. It would be intractable to manually create snippets that not only fit together coherently but that also fit together appropriately in different rhetorical styles. Because the HealthDoc system will perform linguistic repair *after* the snippets of text are selected, it will also be able to tailor the style of the document through a process of stylistic revision.

Even systems that use explicit text-planning techniques (the Migraine Project and PIGLET) would still have difficulty tailoring the style of the document, although it would be easier than in the text-snippet approach. If the text planning is performed by using schemas or template-filling, then the system could include schemas and templates that are based on different models of discourse. However, style is much more than just the overall discourse structure, so schemas and templates cannot really provide customization of style.

2.3 The Potential of Tailored Documents

Natural language generation systems that produce customized documents have additional applications besides those in health information. For example, nurses filling out Worker's Compensation forms must write many versions of a similar document, and would benefit from automation of some or all of this process. In non-medical industries, companies that produce only one type of documentation for their product would benefit from producing different versions of the documentation, tailored for different audiences.

For example, computer companies produce manuals that are used by technicians, sales people, repair people, and end-users alike. The company can produce either one large manual that attempts to cover all topics at the same level (giving too much detail for some readers and not enough detail for others), or it can produce a series of manuals, one for each possible type of reader. The cost and effort of producing and maintaining many different manuals would be reduced by automating the process of selecting what information to include in each manual, and by producing a customized manual for each individual user.

In addition to producing tailored documentation for technical manuals and product information, companies could use customized brochures as part of their marketing efforts: a brochure customized for a prospective customer could help establish a rapport and an atmosphere of trust between the two parties.

Now that this chapter has demonstrated the advantages of customization, the chapters that follow will explain my approach to authoring customizable documents.

Chapter 3

Authoring Tools: A Critical Survey and Evaluation

3.1 Definition

Authoring Tool: An authoring tool is a software tool to support an expert author in creating and maintaining a document.[1] The authoring tool provides methods to specify more than just the content of the document: it assists the author in structuring the document and organizing the content.

[1]The document being authored may be a text document, multimedia presentation, or any other organized medium (e.g., sheet music).

An authoring tool is not a magic wand; it is a sculptor's chisel. An authoring tool does not "de-skill" the writing task to allow people without good writing skills or knowledge to produce well-written documents. It is designed for people who are not computer professionals, but who are willing to use the computer to enhance their work (Goffinet 1996). A NATO workshop on advanced authoring tools (Ibrahim 1991) emphasized that the end-users of authoring tools are professionals (or teams of professionals), so the tools should be tailored to expert authors, not to beginners.

Of course, authoring tools must be usable by beginners, but not necessarily to their full extent. A beginner must be able to create a document, but need not use the advanced features of the tool. For example, a beginning user writing customizable documents for HealthDoc might use the authoring tool to specify pieces of text that are appropriate for particular classes of patients, but need not specify additional linguistic information such as discourse structure. An advanced user would have the expertise to include extra linguistic information in the document.

An authoring tool should guide, but not force, the author into good writing practice. Ultimate responsibility for the quality of the document rests with the author.

3.2 Authoring Tools: The Software Continuum

Give the definition above, there are several different genres of software products that could be considered to be authoring tools. A survey of these products provides a continuum of functionality in authoring tools.

At the lowest level, an ASCII text processor, such as *vi* or *emacs*, is the simplest possible authoring program. However, since text processors manipulate only simple text, and do not assist the author in constructing the document, they cannot be considered *authoring tools*.

The next level of text processor is a word processor, such as *Microsoft Word* or TeX. Using the more-basic features of the word processor, the author specifies formatting details such as a larger font size for titles. At this level, the tool is used to store and manipulate the raw text as well as some text formatting. This text formatting may imply the text structure, but the structure remains only implicit.

At the next stage of authoring tools, the software is used to explicitly specify the structure of the text. For example, when using LaTeX, the author specifies the function of a piece of text (e.g., this is a "subsection" title), instead of specifying formatting characteristics (e.g., this is in 14-point bold). Some more-advanced features of some word processors, such as template forms for different types of documents, are also used to explicitly specify text structure.

The next set of tools, for authoring HTML, Web pages, or hypertext (see section 3.4), moves beyond specifying only text structure to include the relationship of the document to other existing documents, or a part of the document to other parts. At this stage, the author specifies not only the structure of an individual document, but the structure of an interconnected web of documents.

At the next level of complexity, multimedia authoring tools give the author the ability to specify not only the relationship between multiple texts, but the relationship of text to other media, such as video or audio clips. Multimedia authoring tools allow the author to assemble pieces of text, audio, and video into a single presentation. In addition, the tools provide the ability to specify "time-lines," so that the author has precise control over the timing of a multimedia presentation. She can control exactly when a certain video clip appears or when a sound recording begins.

The authoring tools discussed thus far are all used to specify the actual text of a document. In contrast, the TechDoc authoring tool (Rösner and Stede 1994), called the Document Workbench, is used to specify the underlying semantics of a document. Instead of entering text, the author inputs a traditional AI plan, which is used to manipulate objects in a knowledge base. Using the AI plan, TechDoc produces a text plan, which is a representation of the semantic content of a text. The text plan is then converted to English or German text. The Document

Workbench is used to specify the overall document structure and the semantic content of the document.

At the next step in the continuum, the HealthDoc authoring tool is used to specify not only the document's physical structure but also its rhetorical structure. The author explicitly specifies the rhetorical relations between parts of the document. The HealthDoc authoring tool is also used to specify variations of a single document so that the author can create tailored documents.

3.3 Authoring Paradigms

The more-complex authoring tools described above, such as those for hypertext, HTML, or multimedia presentations, are based on various authoring paradigms and metaphors. The strengths and weaknesses of each authoring tool depend partially on the paradigm chosen (Siglar 1996). This section describes the various authoring paradigms, and their advantages and disadvantages.

3.3.1 Scripting-Language Paradigm

The first authoring-tool paradigm is *scripting*, which is the closest paradigm to traditional computer programming, and is usually based on an object-oriented scripting language. The author specifies the elements (text, graphics, video clips, etc.) by filename, the sequencing, and the synchronization of these elements. A scripting authoring tool might provide facilities to create and edit the elements themselves (e.g., graphics software to create illustrations), but the main purpose of the authoring tool is to specify the sequencing and synchronization of the elements. Authoring a document using a scripting language takes longer than using other authoring paradigms, but scripting gives the author more fine-grained control over the document. The scripting paradigm allows powerful interactivity with the audience in multimedia applications by controlling the exact timing of the presentation, and allows the author more direct control of the document.

Scripting would be a poor choice of paradigm for an authoring tool for customizable documents, since the author is an expert in the subject domain, not a computer programmer, and should not need to be able to program a detailed script for the document. Scripting is better for multimedia presentations, which require precise control over the timing of different media elements and varied types of input/output.

3.3.2 Iconic/Flow Paradigm

The second paradigm is the *iconic/flow* paradigm (figure 3.1), which has the fastest development time, and is particularly well-suited to rapid prototyping and applica-

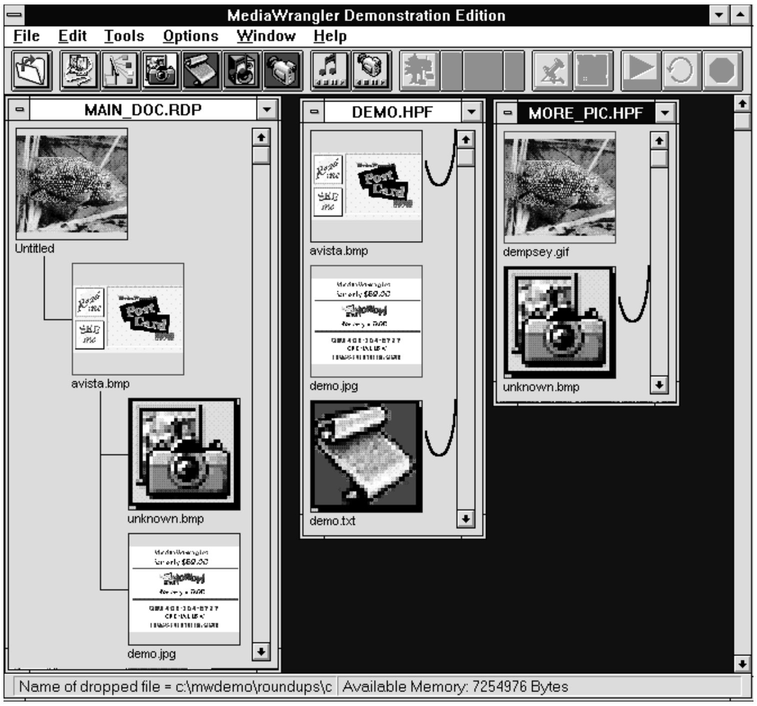


Figure 3.1: The MediaWrangler authoring tool, an example of the iconic/flow authoring paradigm (see section 3.4.5 for a description of MediaWrangler).

tions that interact heavily with the end-user. The authoring tool provides a set of icons representing the possible functions or interactions of the elements in a document, and flow-lines show the links between the icons. For example, there could be an icon for "display text" and one for "play music", and a flow-line between them indicating that they are to occur simultaneously. The icons and flow-lines together constitute a flow-chart of the document.

This paradigm could be used to author customizable documents, particularly documents that include multimedia elements such as audio or pictures. The icons could represent different parts of a document (e.g., text, audio, or pictures), and the flow-lines could be used to tie certain elements of the document to each other. For example, a flow-line could indicate that a certain picture must always appear on the same page as a certain paragraph in a text document. A weakness of the iconic/flow paradigm for authoring customizable documents is that the author should not need to specify all the paths through the document. The conditions attached to each piece of text implicitly indicate the paths through the documents.

3.3.3 Frame Paradigm

A third paradigm, the *frame* paradigm, is similar to the iconic/flow paradigm, except that the links between icons are conceptual, and do not necessarily represent

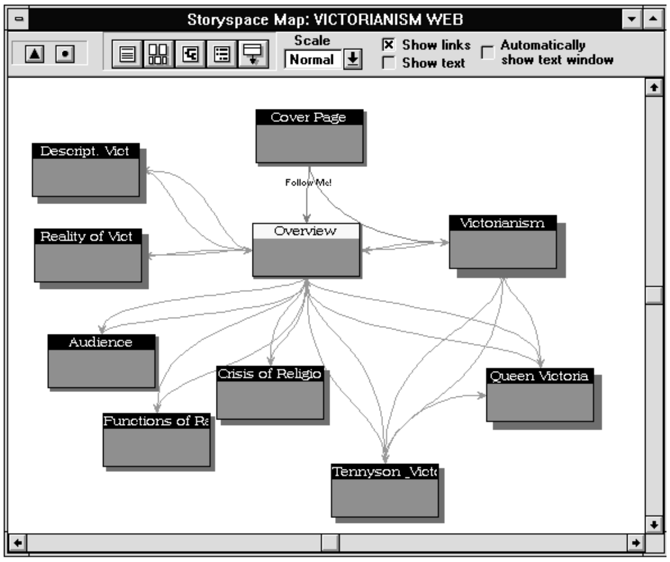
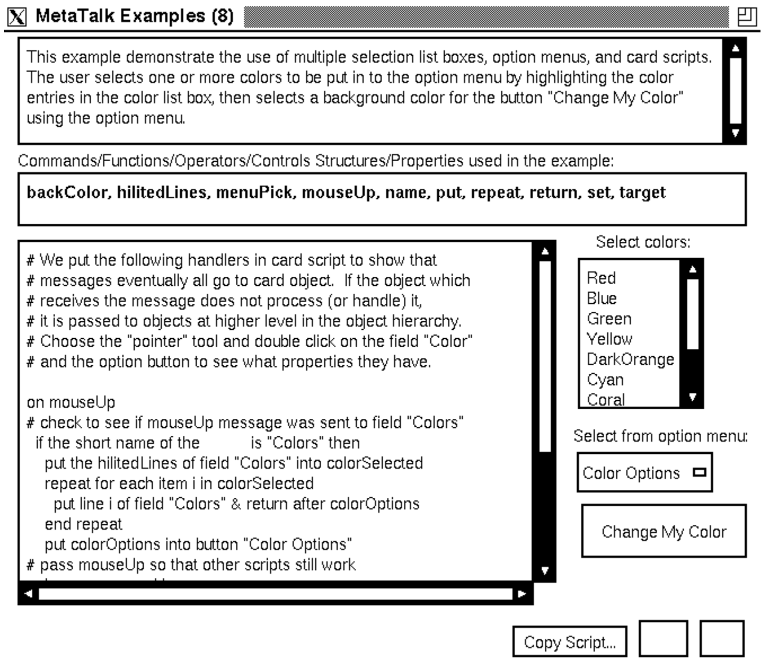


Figure 3.2: The StorySpace authoring tool, an example of the frame authoring paradigm (see section 3.4.2 for a description of StorySpace).

the actual flow through the document (figure 3.2). This paradigm shows more promise for an authoring tool for customizable documents: the icons could represent pieces of the document (e.g., sections, subsections, or pictures) and the conceptual links between icons could be used to represent the discourse structure or other relations between pieces of the document.

3.3.4 Index-Card Paradigm

The next paradigm is the *index-card paradigm* (figure 3.3), which is best for hypertext and other navigation-intensive applications. This paradigm is based on a metaphor of index-cards: the author organizes the content of the document into




```
on showFields
  put the points of graphic "path" into the pathpoints

  # move the field below bottom of card
  set loc of field 4 to last line of pathpoints
  show field 4   # needed later when field is hidden
  repeat with i = the number of lines in the pathpoints \
          down to 2
     # move field from point A to point B
     move field 4 from line i of pathpoints \
          to line i - 1 of pathpoints
  end repeat

end showFields
```

Figure 3.3: The MetaCard authoring tool, an example of the index-card authoring paradigm (see section 3.4.3 for a description of MetaCard). The illustration shows a card containing tutorial information about using selection lists in a multimedia presentation. The script below the illustration specifies that a piece of text (in field 4) is to move around the screen by moving through a list of points on a path (pathpoints).

individual index-cards and specifies navigation links between the cards. If the content on one card has links to the content on other card, then a reader can jump from one card to another. Figure 3.3 shows an index-card authoring tool.

The concept of index-cards and links between cards is potentially useful for an authoring tool for customizable documents. Each card could contain a chunk of information, and the navigation links could be modified to indicate different paths though the document, depending on the characteristics of an individual reader. For example, the links in a document about health-information could be used to specify the conditions under which an individual patient would receive a particular piece of text.

3.3.5 Cast/Score Paradigm

The fifth paradigm, *cast/score*, uses a musical score as a metaphor. Like a piece of sheet music, this paradigm has a horizontal track, and multimedia elements are placed on the track. To indicate that a set of elements are to appear simultaneously, the elements are placed in vertical columns, much like a musical chord. Each cast member appears on one horizontal track, in the same way that an individual instrument is represented by one staff of music. This paradigm is beneficial for applications that use a lot of animation, or that have a lot of synchronized

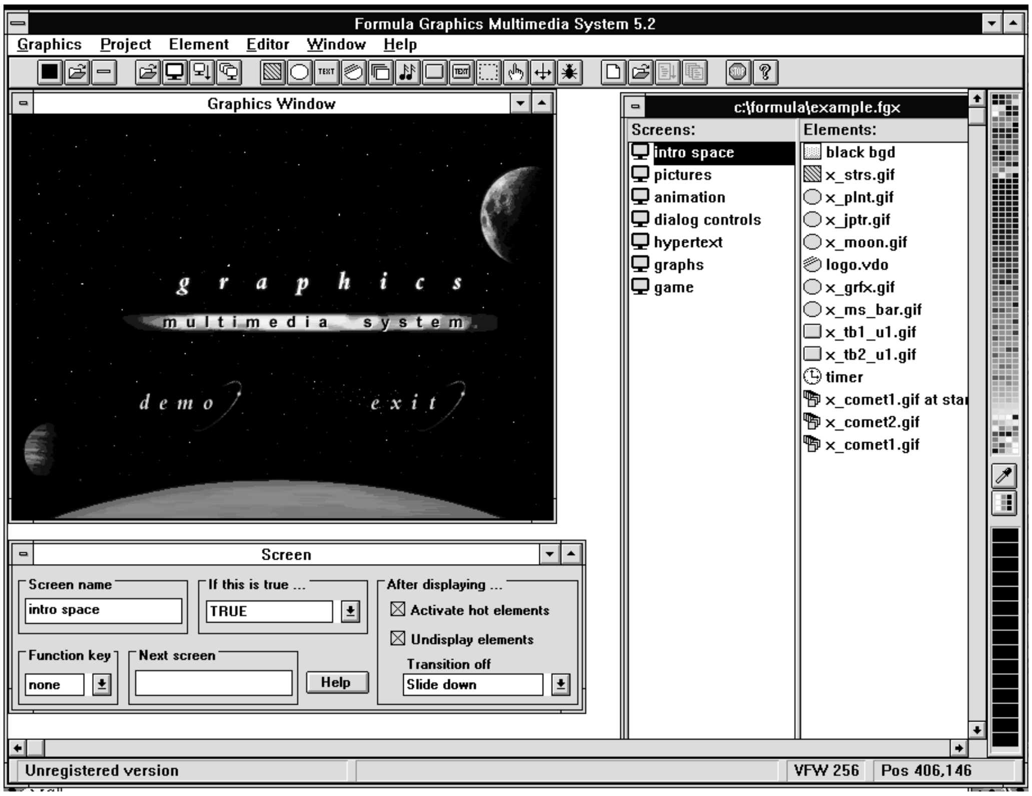


Figure 3.4: The Formula Graphics authoring tool, an example of the hierarchical objects authoring paradigm (see section 3.4.4 for a description of Formula Graphics). A *project* contains a list of *screens* which in turn contain lists of elements. An object-oriented scripting language is used to specify the synchronization of elements.

media. However, it is not appropriate for an authoring tool for customizable text documents.

3.3.6 Hierarchical Objects Paradigm

A final paradigm, *hierarchical objects* (see figure 3.4), is a combination of object-oriented scripting and icon-based paradigms. Elements of the document are represented visually as embedded objects and icons. This paradigm is useful for complex multimedia applications, since very complicated constructions are possible. The idea of embedding objects could be used for authoring customizable documents (e.g., embedding subsections inside sections, sentences inside paragraphs, or illustrations inside paragraphs). This paradigm could be used to construct complicated customized multimedia documents (e.g., customized web pages). It is more powerful than necessary and thus more complicated than necessary for authoring documents that contain only text and illustrations.

3.3.7 Which Paradigms are Most Suitable for Customizable Documents?

Of the authoring tool paradigms listed above, those most suited to hypertext documents (i.e., the frame and card/scripting paradigms) are also likely to be suited to customizable text-based documents. Both types of documents are static, and thus do not require timing controls to specify when multimedia elements will appear. Both types of documents also require the ability to specify links between pieces of the document.

3.4 Authoring Tools: Real-Life Examples

As a first step in designing an authoring tool for constructing customizable documents, I reviewed a variety of commercial and freeware authoring tools. I looked for elements of the authoring tools that could be re-used or adapted for use in an authoring tool for customizable documents.

In general, the sample tools I reviewed were disappointing. The term "authoring tool" is applied to so many different types of tools,[2] that a comprehensive survey or comparison of authoring tools is impossible. Nevertheless, I will present some of the typical authoring tools I investigated.

[2]Some software systems that are called "authoring tools" are not really authoring tools according to the definition given in section 3.1.

For example, the I-Doc (Intelligent Documentation) system (Information Sciences Institute, University of Southern California 1996, Johnson 1994), is used to automatically generate explanations about code in a software system. The explanations are tailored to the user on the basis of his software expertise, the desired amount of detail, the task involved, and his role (e.g., designer, application programmer, maintainer, or end user).

The documentation system composes explanations by combining textual descriptions from a database and automatically generating natural language output to fill in the rest. To generate natural language output, the I-Doc system adds grammatical annotations to the knowledge base. I-Doc attempts to provide these grammatical annotations automatically, but a human checks them and corrects them as necessary.

The human "checker" is not really authoring the document, however. He is providing the software system with encoded knowledge that the system will use to author the explanations itself. So, although the I-Doc system is an explanation generator, and is, as such, a tool for authoring explanations and user help, it is not an *authoring tool* according to the definition given: Because the explanations are generated automatically, I-Doc is not used to assist a human author in constructing a document.

3.4.1 Document Workbench

The Document Workbench is the authoring tool for the TechDoc project at the Forschungsinstitute für Anwendungsorientierte Wissensverarbeitung (FAW) at the University of Ulm, Germany (Rösner and Stede 1994). TechDoc takes as input traditional AI plans, and produces text from those plans. The text from the plans is combined with canned text and templates to produce technical manuals.

The Document Workbench has facilities for entering and editing AI plans, but these are not of use to HealthDoc. Some of the utility functions in TechDoc, such as including illustrations or viewing a knowledge base, could be included in HealthDoc at a later stage of the project.

The Document Workbench is not a multimedia authoring tool, and thus does not fit neatly into one of the authoring tool paradigms listed above. It could be considered to be a type of *scripting-language* tool, if the AI plans are considered to be a programming language that specifies the ordering of the document.

3.4.2 StorySpace

StorySpace (Eastgate Systems, Inc. 1996) (see figure 3.2) is the most promising of the authoring tools I surveyed. This program is used to create hypertext documents. StorySpace is a good example of the *frame* authoring tool paradigm: it uses icons (in the shape of rectangles) to represent pieces of the document and conceptual links (in different colours) to represent relationships between the icons.

StorySpace uses a graphical interface to specify the layout of the pages of the hypertext system. Each page is represented by a box containing the title of the page and the text of the page. The author draws links between the pages to create an interconnected web of pages.

There are different types of links between pages: some are between entire pages to indicate that one page can be reached from another, and some are between specific parts of a page (e.g., a specific word) and another page.

The approach used by StorySpace is very relevant to HealthDoc. Instead of each box representing a page title and text, it could represent the condition indicating when a piece of text is relevant, and the actual piece of English text. The boxes could contain pieces of text of arbitrary size: a section, paragraph, sentence, phrase, or individual word.

The links between boxes could represent the discourse structure of the document. For example, the HealthDoc system currently uses rhetorical relations between sentences and coreference links between phrases. Links between boxes could represent rhetorical relations, and links from specific parts of the box (e.g., individual words) could represent coreference links between individual words of two sentences in the master document.

The design of the authoring tool described in this thesis uses many elements of the StorySpace authoring tool. In particular, I used the ideas of little boxes on the screen, with a label for each box (in StorySpace the label is a title; in the HealthDoc authoring tool, the label is the condition under which the piece of text is relevant). The boxes can be moved around and grouped together to form sections and subsections. I also re-used the idea of drawing links between the boxes (lines on the screen) to indicate the discourse structure of the document.

3.4.3 MetaCard

MetaCard (MetaCard Corporation 1996) (see figure 3.3) is used to develop graphical user interface (GUI) applications and GUI front-ends to shell scripts and other character-based applications. It is used for UNIX systems, and is similar to PC and Macintosh products such as HyperCard. MetaCard is a classic example of the *index-card* authoring paradigm.

The main uses of MetaCard are for developing application prototypes, adding GUI front-ends to UNIX applications, authoring on-line presentations (e.g., multi-media presentations), and authoring on-line documentation and computer-assisted education.

Creating MetaCard applications involves learning a scripting language, similar to a simple programming language, to specify the links between individual cards. Because the author needs to learn this scripting language, MetaCard has a significant learning curve.

I personally found MetaCard annoying to use. There are many windows open simultaneously, the windows are so large that they need to overlap (so you cannot see everything at once), and the windows are difficult to close when they are no longer needed.

MetaCard's approach to authoring is more complicated than necessary for an authoring tool for customizable documents.

3.4.4 Formula Graphics

Formula Graphics (Harrow Software 1996) (see figure 3.4) is used to develop interactive multimedia presentations (e.g., animating a children's book) or business applications with databases. It is based on the *hierarchical objects* authoring tool paradigm.

It uses an object-oriented programming language of instructions, operators, and functions. The author constructs a presentation by specifying a series of storyboards, and then chooses elements (e.g., artwork, animation, sound clips) to include in the storyboards.

Formula Graphics is well-designed and would be very helpful in authoring multimedia applications. However, it did not contain any elements that would be useful in the HealthDoc authoring tool.

3.4.5 MediaWrangler

MediaWrangler (Altavista Technology, Inc. 1996) (see figure 3.1) is used to link pictures, sound, and text together into an interactive multimedia application. It is based on the *iconic/flow* authoring tool paradigm, and uses the metaphor of a lasso and a hitching post.

A 'hitching post' is used to organize individual media into groups that are based on a common theme (e.g., pictures of flowers, animal sounds). The 'lasso' is used to 'round up' individual media and link them together using hyperlinks.

MediaWrangler relies very heavily on pictorial icons (e.g., the menu consists exclusively of icons), but the icons are difficult to identify. It is also difficult to know which icon to use, because the 'hitching-post' metaphor is confusing.

I did not find any elements in MediaWrangler that would be useful in the HealthDoc authoring tool.

3.4.6 VivaTEXTE

VivaTEXTE (VivaTEXTE 1996) is another multimedia authoring tool, designed for beginning users. It relies heavily on Microsoft Windows, and has the same look-and-feel as Windows. The interface is similar to a word processor, which is already familiar to new users.

VivaTEXTE exploits the built-in Windows help functions to produce hypertext documents for Windows applications. The VivaTEXTE authoring tool produces documents in "WinHelp" format: a formatting standard for interactive help files in Windows. WinHelp files are normally used to provide hypertext help documents; VivaTEXTE allows the author to create other types of hypertext documents, and to add graphics and sound to create multimedia documents. Because all Windows systems provide the functionality to read WinHelp files, no additional software is needed at the reader's end, and because most users are already familiar with Windows help, no additional training is required for a reader to use a VivaTEXTE document.

The VivaTEXTE authoring tool for creating files in WinHelp format is based on standard Windows word processors (e.g., WordPerfect) with additional operations to create elements of WinHelp files, such as hypertext jumps or pop-up windows.

Except for its similarity to a word processor, there were no elements of Viva-

TEXTE that could be used in the HealthDoc authoring tool. It is very platform-specific, and it is tied too closely to WinHelp functionality.

The VivaTEXTE authoring tool is similar to the *index-card* authoring paradigm, although the text of the document is not explicitly organized into index-cards. VivaTEXTE is more like a word-processor with advanced features for creating links than a graphical authoring tool.

3.4.7 MicroMass and IntelliWeb

The IntelliWeb authoring tool (MicroMass Communications, Inc. 1996) was used by the University of North Carolina in their studies of the effect of tailored letters on smoking cessation (Strecher et al. 1994), dietary improvement (Campbell et al. 1994), and mammography screenings (Skinner, Strecher, and Hospers 1994).[3] IntelliWeb contains a database of pieces of text, and conditions for when each piece of text is relevant. The documents are in HTML, and include diagrams and formatting information.

To specify that a particular piece of text is relevant to a particular reader, the author inserts specially defined HTML tags around the text, and includes in the tag the condition for selecting the text. The tag to indicate that a piece of text is selectable is called "CHUNK".

[3] See section 2.2 for details of these studies.

As an example, consider a health-education document explaining diabetes. The document could be tailored according to the type of diabetes the patient has: either insulin-dependent or non–insulin-dependent. The text for a patient with insulin-dependent diabetes would be specified using the CHUNK tag as follows:

```
<CHUNK name = "introduction"
       condition = "(diagnosis==insulin-dependent)"
       action = NIL>

       This is a piece of text for patients who have
       insulin-dependent diabetes, including the HTML
       tags necessary to format this piece of text.
</CHUNK>
```

Currently, IntelliWeb does not include an authoring tool to assist in tagging text. The author uses the word processor or HTML authoring tool of his choice. MicroMass Communications, Inc. is developing an authoring tool, which has not been released yet.

The IntelliWeb system is similar in spirit to the HealthDoc tailoring system, but HealthDoc also includes extra linguistic information that is used to repair the selected text. IntelliWeb does not attempt to repair the selected text, so an

IntelliWeb author must ensure that all possible combinations of text are coherent. Since the number of possible combinations increases exponentially with each new piece of text added, it becomes impossible to check every combination, even for relatively short documents. Thus, the documents produced can be very awkward and unpolished (Victor Strecher, p.c.).

3.4.8 Document Sciences Corporation

Document Sciences Corporation, a subsidiary of Xerox, offers a software system for composing, assembling, and producing documents for "mass personal communication" (Document Science Corporation 1996). By combining database management, document management, and database publishing, they can produce personalized documents ("personal communication") in high volumes ("mass"). The Document Sciences authoring tool is for text-based documents, not multimedia documents, so it does not follow any of the authoring tool paradigms listed in section 3.3.

Elements that could be included in a document are tagged with logical rules, using tags that are similar to HTML tags. The tailoring performed by Document Sciences is less sophisticated than that performed by the authoring tool described in this thesis: the Document Sciences system is used to insert variables in the text (e.g., names, prices), to select whole documents from a database of documents, or to format a document for printing in a particular format. The authoring tool in this thesis will be used to select the content of the document at *any* level (e.g., word, sentence, paragraph) and to tailor the style and content of the text according to the user's personal characteristics.

Document Sciences Corporation's strengths are in high-volume production, and document production on a variety of platforms.

3.4.9 CapSoft Development

To customize routine paperwork, particularly legal documents, CapSoft Development (CapSoft Development 1996) has introduced authoring tools to automatically create customized documents. The documents are essentially intelligent templates, which are used to query the user and then to fill in the appropriate slots in the template. The system can also select optional paragraphs, and ensures that pronouns and verbs are correctly filled in (e.g., by gender, tense).

To write a CapSoft document, the author identifies the text that will change in each document, and inserts a variable in its place. Using the WordPerfect word processor, the author highlights the text, then chooses "variable" from the menu. Each variable has a type (e.g., number, date, true/false) and a name. When the system encounters a variable in a document, it displays dialogue boxes to prompt for the necessary information, and ensures that the information entered is appropriate for the type of the variable.

The template approach used by CapSoft by itself is inadequate for the amount of customization required in the HealthDoc project, since it doesn't allow customization at every level of document granularity (e.g., word, sentence, section) desired. However, templates could be used for routine information such as names or dates and another authoring approach could be used for tailoring at other levels.

Since CapSoft's authoring tool is used to create text-based documents, not multimedia documents, it doesn't follow any of the authoring tool paradigms listed above.

3.4.10 Eon

The Eon project at the University of Massachusetts is developing an authoring tool for intelligent tutoring systems (University of Massachusetts 1996). It is used to create knowledge-based tutorials, built from modular, re-usable instructional content. An instructor can specify a sequence of instruction modules, using generic, reusable teaching strategies.

The content and style of instruction is tailored to the individual student's needs, and the teaching strategy can be chosen dynamically, on the basis of the changing needs of the student.

The Eon authoring tool is an example of the *frame* authoring paradigm. To create a tutorial, there are two content editors: the Topic Network Editor and the Tutoring Strategy Editor. In the Topic Network Editor, different shapes represent different kinds of content (e.g., lecture, quiz, demo), and different colours represent levels of difficulty. To define pedagogical relationships, the author uses lines and arrows to connect the shapes. Once the instructional content is complete, the Tutoring Strategy Editor is used to specify how the network of content is traversed.

The idea of different shapes (or perhaps different icons) to represent different content could be used by the authoring tool for customizable documents, to represent text, images, and possibly multimedia elements. Using different colours to represent different concepts could also be re-used in the authoring tool.

3.4.11 Walden's Paths

The Walden's Paths project (Texas A & M University 1996) is developing another authoring tool for creating educational materials and tutorials. In this case, the instructor collects information from the Internet and brings the sources (the Web pages) together into a single 'path'. This path directs the student's traversal along a route through educational components extracted from the set of Web documents.

The students can wander off the path as they explore topics of interest, but can always get back on the path by clicking on a "return to path" button.

Walden's Paths contains two main software components: the Path Authoring Tool, and the Path Server. The instructor uses the Path Authoring Tool to create, modify, validate, and re-use paths. The students use the Path Server to navigate the paths. The Path Authoring Tool is used more to collect and organize sets of Web addresses than to construct a document, so it doesn't follow any of the authoring paradigms listed in section 3.3.

This project provides a useful tool for creating paths through the Web, but is not well-suited to an authoring tool for customizable documents. The paths are not tailored according to the student, so there is no functionality to specify that certain content is more relevant to some students than others.

3.4.12 Expert Publisher

Expert Publisher (Database Publishing Software 1996) produces customized documents by querying an SQL database to retrieve pieces of text that match the queries and then composing the results of the queries.

The author creates a library of re-usable objects, such as paragraphs, images, tables, or templates. Each object is then tagged with both system-generated and user-generated keywords. When composing a document, the author attaches SQL queries to elements of the document, which indicate the parts of the document that are to be customized and the query that chooses which object to include. Expert Publisher does not follow any of the authoring paradigms listed in section 3.3. The author uses a word processor to create the document and to include SQL queries in the document.

To create a final document, Expert Publisher goes through the document to find elements that have SQL queries attached, executes the queries, extracts the appropriate objects from the database, pulls the objects into a word processor such as FrameMaker, composes the document, and then formats it for printing.

Expert Publisher does not attempt to repair incoherent documents; the author must ensure that all possible combinations of objects in the document fit together.

The HealthDoc authoring tool could use the idea of a database of pieces of text and queries to retrieve relevant pieces for individual readers. The author could tag the database elements with keywords that would indicate the type of reader for whom the element was relevant.

3.4.13 Author's Argumentation Assistant

The Author's Argumentation Assistant (AAA) (Schuler and Smith 1990) is an authoring tool to create argumentation-based hypertext. AAA is an example of the *frame* authoring paradigm.

The design of AAA is based on a theory that there are seven cognitive modes for writing: exploration, situational analysis, organization, writing, editing global organization, editing coherence, and finally editing expression. To support the seven cognitive modes, there are five system nodes provided in AAA: an argumentation mode, rhetorical organization mode, tree mode, editor mode, and text mode.

In the argumentation mode, concepts are represented as nodes, and the nodes are moved manually around the screen to form clusters of loosely related ideas. Links are drawn between the nodes and clusters to indicate argumentative relationships (e.g., generalize, support position, negate) between them. The final result of this stage is an interconnected network of issues that are linked with relationships to positions and arguments.

In the rhetorical mode, the structure and content of the document are organized. The author selects elements or substructures of the document, then re-organizes them according to his rhetorical objectives and principles, and to the model of rhetoric he is using. The model of rhetoric is not stored explicitly; the author just keeps it in mind as he is writing. Some or all of the argumentative links are transformed to appropriate rhetorical ones. The rhetorical nodes are grouped into a coherent text chunk, and the author defines the reading path through the document.

In the tree mode, the author builds the hierarchical structure of the document, in terms of sections and subsections. Then, in the editor mode, the author opens up the concept nodes, and writes or edits a block of text for the node. Finally, in the text mode, the author steps through the hierarchical tree and displays it for editing, in order to check coherence and expression editing.

The Author's Argumentation Assistant contains many elements that are directly relevant to an authoring tool for customizable documents. The authoring tool presented in this thesis uses rhetorical links between pieces of text, so AAA's rhetorical mode is directly relevant. Since argumentation is as important in customizable documents as in generic documents, the AAA argumentation mode would be valuable in an authoring tool for customizable documents. The functions of the tree mode, editor mode, and text mode are all performed by the authoring tool described in this thesis, although they aren't labeled as separate modes.

Using the Author's Argumentation Assistant to author customizable documents requires extending AAA to include a way to indicate which pieces of text are relevant for which types of readers.

3.4.14 MediaMosaic

A sample multimedia authoring tool from the University of North Carolina, MediaMosaic (Lin 1992), is a multimedia editor designed to import data in multiple formats from multiple sources into a single document, and to perform mark-up editing in context. The MediaMosaic authoring tool is an example of the *hierarchical objects* authoring paradigm.

MediaMosaic is based on the idea of multiple, embedded screens. The multiple screens are controlled by a pseudo–'root'-window. Each screen can run a different X client for each medium (e.g., ghostview for postscript, xdvi for dvi files), and the author can use different editors for each medium/screen. Multimedia elements from varied sources can simply be imported into one of the embedded screens to compose a multimedia presentation from a variety of source files.

To edit the presentation, there are two views available: a document view to view the entire document, and an element view to edit a single element, using the most appropriate editor.

An authoring tool for customizable documents could also use multiple views, to edit the whole document (e.g., overall structure) and to edit individual pieces of text. The other features of MediaMosaic, such as embedded screens with different editors, are necessary for multimedia documents, but not required for text-only documents.

3.5 Ideas with Potential for an Authoring Tool for Customizable Documents

After reviewing various authoring tools, the following concepts seem relevant to an authoring tool for customizable documents, and could potentially be re-used in the design of an authoring tool for HealthDoc:

- Boxes on the screen, containing text, which can be moved manually.
- Lines (links) drawn, connecting the boxes, to indicate relationships between the boxes (e.g., discourse relationships).
- Different coloured links, to represent different types of relationships.
- Different views of the document (overall views, and more focused views).
- Different shapes to represent different content in the containers (e.g., boxes for text, circles for illustrations).
- A formal model of argumentation.
- A formal model of different rhetorical models or schemas.

- Templates for routine personalization such as names and dates.

- A database of document elements (e.g., paragraphs, illustrations), and SQL queries to retrieve the elements relevant to an individual reader.

Chapter 4

Requirements for an Authoring Tool for Customizable Documents

4.1 Determining the Requirements

The requirements for an authoring tool for customizable documents come from various sources: the users of the system (the authors), the tasks the authors need to perform, and the rest of the software in the system. For example, in the HealthDoc project, there are requirements from the medical writers (e.g., the ability to read information from a patient's medical record), requirements based on the task of authoring customizable documents (e.g., the ability to specify the types of readers for whom a piece of information is relevant), and requirements from the other software in the HealthDoc system (e.g., the ability to include the linguistic information needed by sentence planner). The authoring tool must gather enough information from the author to be able to perform the linguistic repair of the document.

4.1.1 A Cognitive Walkthrough of the Authoring Process

To help determine the tasks that an author must perform to construct a customizable document, I have manually written a master document about the HealthDoc project, and partially written one about smoking cessation.

As I was writing, I performed a "cognitive walkthrough" of the process, keeping track of the steps involved. The final authoring tool must have the functionality to perform all of these steps.

The cognitive walkthrough showed that to author a master document, the writer should proceed as follows:

1. Choose the communicative goal(s) that the document will express. For example, the communicative goals may be to inform and to persuade the reader.

2. Make a list of the prospective audiences for the document.

3. Study this list of audiences to determine the characteristics that differentiate groups in the list from one another. The document should be tailored, in

part, on the basis of these characteristics.

4. Make a list of the selection criteria that will be used to tailor the document. These criteria will be variables such as "age" or "technical-reading-level," which can take on different values for different readers. For example, if the document is going to be tailored on the basis of technical reading level, there could be a variable called "Technical-Reading-Level" that has the possible values "high," "medium," "low," or "don't know." The values of the selection criteria for an individual patient are used to select the most relevant pieces of information from the master document.

   For each variable in the list of selection criteria, determine what its possible values are.

5. Look at documents similar to the current one, to get an idea of their structure and the topics they cover. These documents may describe the same topic as the current document, or share communicative goals similar to the current document.

6. Make a list of topics covered in the similar documents that describe the same topic as the current master document.

7. Make a list of possible topics to be included in the current document. From

this large list of topics, select a pool of topics that will be included in the current master document.

8. Order this list of topics, according to the document's communicative goals. Some topics will become sub-topics of other topics.

9. While writing the document, keep in mind the prospective audiences and the selection criteria being used. For each topic and sub-topic, decide what information should be included for each type of reader, and how the information should be expressed for each type of reader.

10. If selections from the master document are going to be repaired by a sentence planner to provide stylistic variations for different readers, go through the master document and specify the rhetorical relations between pieces of text. Also go through the document and specify the words and phrases that are coreferential with each other.

4.1.2 Organizing the Requirements

The requirements that I gathered are listed in the next sections. They are divided into two main groups: those requirements that are general to all authoring tools for customizable documents, and those that are specific to the HealthDoc project.

All authoring tools for customizable documents have requirements for software development and maintainability, requirements for the user interface, word-processor functionality requirements, and customization requirements. The HealthDoc authoring tool has additional requirements for entering linguistic information (rhetorical relations and coreference), for interfacing to an English text parser, and for reading and writing files in HealthDoc's master-document format (TSL) (see section 1.3.3).

For each type of requirement, there are three sub-groupings: those requirements that provide the basic operation of the authoring tool ("Basic Requirements"), those that provide more advanced functionality that is essential to the authoring tool ("Advanced Requirements"), and those that are not essential, but are still desirable ("Optional Enhancements").

4.2 Requirements for All Authoring Tools

All authoring tools for customizable documents have some requirements in common, which are listed in this section. These requirements must be taken into consideration by a programmer when he is developing an authoring tool for end-users.

4.2.1 Software Development and Maintainability Requirements

Requirements for software development and maintainability are general to all software systems. They include the choice of an appropriate programming language, a system for version control, and software design and documentation that will allow future programmers to maintain the system.

Basic Requirements

- The programming language must be in sufficiently common use that future programmers will be able to easily learn the language, and will be able to find sufficient information about it.
- The programming language must be established and have a community of users, so that it will be maintained and updated regularly.
- It should be possible to implement individual procedures and functions in a variety of languages, so that pieces of code can be optimized. The main programming language chosen must allow the inclusion of code written in other languages.

- For the initial implementation, the programming language should be easy to learn and should allow rapid prototyping.
- The code must be structured and documented to allow others to easily understand its functionality.
- The software should be stored by a revision control system, to track changes to the code, and to enable comparisons between different versions of the code.
- The system should be implemented on a hardware platform that is widely available.
- The authoring tool should be implemented on the same platform as the rest of the system that performs the actual natural language generation.

Advanced Requirements

- Functions and procedures that would be more efficient if they were written in another programming language should be re-written in the most appropriate language.
- It would be advantageous if the authoring tool could run over the Internet. While most authors would not be using the tool over the Web, it would be

beneficial to be able to give demonstrations of the authoring tool remotely.

4.2.2 User Interface Requirements

The following requirements stem from the fact that the authoring tool includes a user interface, so it must meet standard interface requirements.

Basic Requirements

- It must be reasonably easy to use. The advanced functions should be no harder for the author to use than the more-advanced functions of a word processor.
- Authoring a master document should require only minimal linguistic expertise. Some expertise will be needed to input the linguistic information used in the repair process, but the authoring tool should attempt to limit the amount of expertise needed as much as possible.
- The author must be able to either write a new document from scratch or edit an existing document. The author needs a way to include or re-use text from an existing document (in another format) in the master document.

Advanced Requirements

- The tool should provide both "novice" and "expert" modes, especially for entering the linguistic information.
- There should be keyboard shortcuts for repetitive steps.
- There should be different ways of viewing the document. For example, there could be a full-document view, "zoom" view, section-headers-only view, and a fish-eye view.

Optional Enhancements

- Instead of the author labeling linguistic information such as rhetorical relations directly, the tool should try to automatically (or, more realistically, semi-automatically) determine this information. It should query the author to ensure that the automatically labeled information is correct.
- If the tool queries/prompts the author, there should be different levels of questioning/prompting, depending on the linguistic expertise of the user.

4.2.3 Word Processor Functionality Requirements

Because the author will be entering text in the master document, the authoring tool must provide the functionality of a simple word processor.

Basic Requirements

- The author must be able to input ordinary text.
- The authoring tool must allow document structuring such as titles or subsections, and must display this structure clearly.
- The tool must provide the ability to save documents, and later access the stored document to continue editing it.

Advanced Requirements

- The author should be able to specify different fonts or highlighting (e.g., italics).
- The tool should provide a spelling checker.
- There should be a way to include other media (e.g., illustrations) in the master document and to attach conditions to these media, indicating their relevance to different types of readers. Some media will need to be tied to specific phrases of text (e.g., an illustration should end up on the same page as the paragraph that refers to it).

Optional Enhancements

- Because the document formatting might be affected when pieces of text are selected from the master document, the repair process must include methods to repair "broken" formatting.

4.2.4 Customization Requirements

Since the authoring tool will be used to create customizable documents, there must be a way to specify the pieces of text that are relevant to each type of reader.

Basic Requirements

- There must be a way to indicate a condition for each piece of text, in order to specify the relevance of the text to different types of readers. It must be clear which condition is attached to which piece of text.

- There must be a list of selection criteria that can be used in the current document to specify the conditions under which a piece of text is relevant. The selection criteria should include a variable name and a list of possible values for the variable. For example, one variable could be "technical-reading-level", and the possible values could be "high," "low," "medium," or "don't know."

- The author must be able to add variables to or delete variables from the list of possible selection criteria. The author must also be able to add and delete possible values for each variable.

- The selection criteria should always include values for "everyone" and for "don't know."

- The conditions must conform to some pre-specified format, so that an automatic "consistency checker" can be used to verify the conditions.

- There must be a way to indicate that a group of texts are all variations of a single theme, where each text is for a different audience. For example, if there were three sentences describing the same concept, but at three different levels of technical detail, these three sentences are variations of each other. The texts that form such a set of variations should be visually grouped together, so that it is obvious that they are variations of one another.

Advanced Requirements

- There should be a way to specify variables and formulas, such as $NAME or ($LIFE-EXPECTANCY − $AGE). The variables should have types (e.g., integer, string, boolean).

- The authoring tool should provide a consistency checker, which checks the conditions specified for the pieces of text. The level of checking done may depend on the user (novice or expert).

  The consistency checker will only make suggestions; it will not enforce them.

  Types of consistency checking may include:

  - Does every piece of text have an associated selection condition?
  - Are all the conditions well-formed, according to the pre-specified format?
  - In a set of sentences that are variations of one another, are the conditions attached to the individual sentences in the variation set mutually exclusive? Will every reader receive no more than one of the texts from the set of variations?
  - Has the author specified which pieces of text are default texts to be used if the conditions for a certain reader are unknown? For example, in a document explaining the medical condition of diabetes, if it is not known whether a patient has insulin-dependent or non–insulin-dependent diabetes, the author will probably want to include information about both types of diabetes. However, if part of the document is about the interaction of diabetes with some other rare medical con-

dition, the author will probably not include this information unless it is definitely known that the patient has the other rare condition.

- In the case of HealthDoc, the consistency checker must know the format of the medical record,[1] to check that any medical conditions specified as selection criteria actually exist.

- There must be a way for the author to "test" the document by viewing the text that would be selected for a given patient. The author must be able to set the values of some of the selection criteria, to see either:
  - The actual text that would be selected.
  - The path through the document that has been chosen.

Optional Enhancements

- As part of the document "test" procedures, the authoring tool will need to determine the values of selection criteria for individual readers. For example, in the HealthDoc system, the test routines will need to be able to read the patient's medical record to determine the diagnoses for the patient.

[1]To avoid a commitment to any of the emerging standards for electronic medical records, the HealthDoc project will be using its own simplified medical record format, essentially a list of allowable diagnoses and other patient data.

This is *not* a simple task: medical records contain unconstrained text, so techniques from natural language understanding are required.[2]

4.3 Requirements Specific to the HealthDoc Authoring Tool

At the current stage of the HealthDoc project, the author must include in a master document extra linguistic information that is used to guide the repair of the document. The author must indicate rhetorical relations between parts of the text, and must also specify words and phrases that are coreferential. The types of linguistic information needed may change as the development of the HealthDoc sentence planner progresses.

4.3.1 Rhetorical Relation Specification Requirements

Rhetorical relations (Mann and Thompson 1986) are relations such as elaboration, justification, and sequence, that hold between sentences or groups of sentences in the master document. These rhetorical relations are used by the HealthDoc system to arrange and order the text of the final document into a coherent structure.

[2]For example, a simple string search for "diabetes" will not tell you whether or not a patient has that condition: the string "diabetes" could occur in a patient's record as either a positive or negative diagnosis.

Basic Requirements

- The author must have a way to specify the rhetorical relations between pieces of text.
- The possible rhetorical relations must be listed, so the author knows what the allowable relations are.
- The relations should be visible in the document.

Advanced Requirements

- To assist the author in specifying rhetorical relations, the authoring tool should attempt to (semi-)automatically determine the relations. For example, the tool could use a question-and-answer cycle to determine the author's intent, in order to distinguish between similar relations. The level of questioning would depend on the author's expertise.
- The authoring tool could offer explanations of the difference between rhetorical relations (including examples) to assist the author in choosing the correct relation.

Optional Enhancements

- Ideally, the authoring tool would automatically determine the rhetorical relations, so the author would not need to specify them at all. However, this level of natural language understanding is well beyond current research.

- The authoring tool could be expanded to include formal models of communication, argumentation, and rhetoric, which would guide the author while she is constructing the document. The models would assist the author in fulfilling her communicative plan, and the extra information could be used to better repair the final document.

4.3.2 Coreference Specification Requirements

Words or phrases that refer to the same object or concept are said to be coreferential. HealthDoc uses knowledge of coreference to resolve and repair reference, to reduce repetition and to perform aggregation in the final document.

Basic Requirements

- The author must be able to indicate words or phrases that are coreferential with each other. She must also specify the type of coreference involved (e.g., identical, intensional, extensional).

- The possible types of coreference must be listed, so the author knows what the allowable types are.
- It should be made visibly obvious in the document which words and phrases are coreferential with each other.

Advanced Requirements

- The authoring tool could highlight words or phrases that are likely to be coreferential with other words or phrases. For example, pronouns are likely to be refer to a preceding noun; the same noun used in two consecutive sentences is also likely to be referring to a single object.
- Since text tends to produce chains of coreferential phrases (Morris and Hirst 1991), the tool should show whole sets of phrases that are coreferential. Clicking on one of the references should show the whole set of related references and the type of coreference involved.

Optional Enhancements

- The authoring tool could highlight pairs of words/phrases that it thinks are coreferential with each other, and then allow the author to accept or reject the pairings. Suggested pairings could include a pronoun and a preceding noun, the same noun in consecutive sentences, or nouns that are synonyms of each other. The tool could also attempt to determine the type of coreference involved, possibly using theories of anaphor resolution (Hirst 1981).

4.3.3 Requirements for an Interface to the Parser

Before the HealthDoc sentence planner repairs the text selected for an individual patient, the sentences of the master document must be converted to TSL, the internal representation used by HealthDoc. The first step in converting the text is to parse each sentence.

However, the parser will not be able to correctly parse every English sentence; some intervention by the author will be required. Sentences that are ambiguous must be explicitly disambiguated by the author, so the authoring tool must have an interface to the parser.

Basic Requirements

- The authoring tool should receive messages from the parser, indicating if the sentence is difficult to parse.
- The tool should query the author to help determine the correct parse of the

sentence, and should then return diagnostic information to the parser, so that the parser can correct the parse tree appropriately.

Advanced Requirements

- When querying the author to correct the parse tree, the authoring tool should word the questions in a manner that the author will understand. Since the questions posed by the parser currently being used in the HealthDoc project are cryptic,[3] the authoring tool must translate the parser's questions into a more understandable form.

4.3.4 Input/Output(TSL) Requirements

The format specification language for master documents used in the HealthDoc system is called *Text Specification Language (TSL)*. TSL uses a Lisp-like format and organizes the text of the master document into sets of variations that are annotated with the selection criteria and the extra linguistic information (rhetorical relations and coreference). A single TSL document can contain multiple representations of each sentence (e.g., ordinary text, an SPL representation, an HTML representation).

The HealthDoc authoring tool must read and write documents in TSL format.

[3] HealthDoc is currently using the Pundit parser (Dahl 1992).

Basic Requirements

- The authoring tool must be able to parse a document stored in TSL format, and re-create the document on the screen.
- The tool must also save master documents in TSL format.

Advanced Requirements

- As the TSL format evolves, the authoring tool must be updated to read and write the new format.

Optional Enhancements

- The text of the master document is currently stored in the TSL in both SPL form and HTML form. It is conceivable that the text could be stored in other forms as well. The authoring tool would need to be expanded to include these alternative forms of representation as they are added to the specification of TSL.

Chapter 5

The Design and Implementation of the Authoring Tool

In this chapter, I will describe the design and implementation of the HealthDoc authoring tool for customizable documents. I will first give an overview of the design of the authoring tool by explaining a screen shot of the authoring tool in use. Then, for the rest of the chapter, I will explain my design decisions in more detail.

The structure of this chapter mirrors that of the previous chapter (listing the requirements for an authoring tool). For each type of requirement, I explain how the requirement was met, and the design decisions that I made.

5.1 An Initial Look at the Authoring Tool in Use

Figure 5.1 shows a screen shot of the HealthDoc authoring tool in use. The text shown is part of a master document about the medical condition of diabetes.

On the left-hand side of the screen is an area titled "Selection Criteria". In this area, the author lists the variables being used to tailor the document, and the possible values of those variables. For example, the document could be tailored on the basis of technical reading level so there would be a variable called "Technical Level" which has the possible values "High", "Medium", "Low", or "Don't Know". The selection criteria area is editable, so the author can add or delete variables and values.

The main area of the authoring tool contains the text of the master document and the linguistic notations used by HealthDoc. The author enters the text of the document in the little boxes on the screen. Each little box contains two fields: the larger field contains the English text, and the smaller field above it contains the condition under which the text in the box (a particular variation) is to be selected. For example, the single box roughly in the centre of the screen contains the sentence *The condition that you have is insulin-dependent diabetes.*, and the selection condition "(diagnosis == insulin-dependent)".

The author can cut-and-paste variables and values from the Selection Criteria

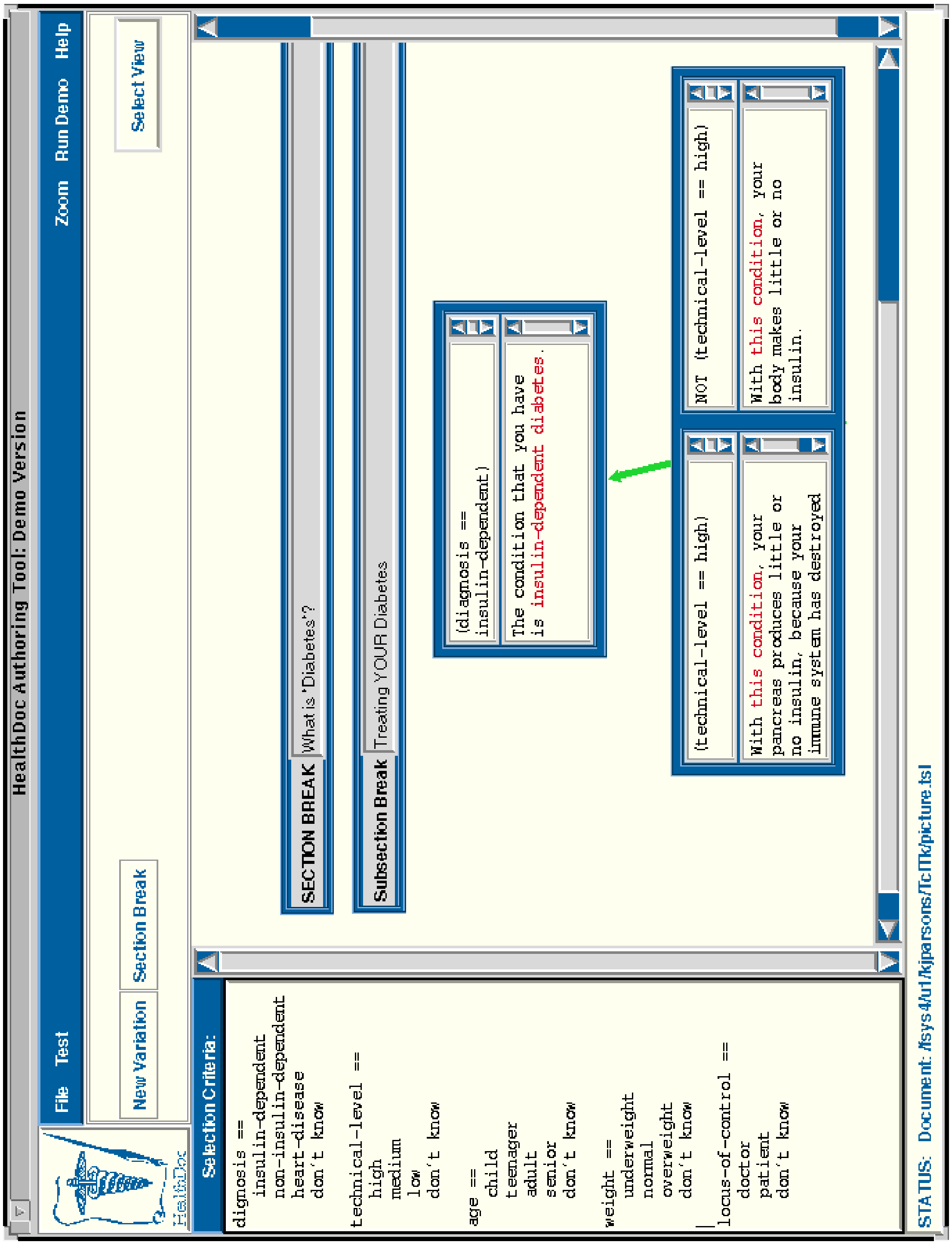


Figure 5.1: Screen shot of the HealthDoc authoring tool in use.

area into the condition field of the boxes.

A set of boxes that represent different variations of a piece of text are grouped together on the screen, and can be manipulated (e.g., moved around on the screen) as a unit. For example, the large box at the bottom of the screen contains two smaller boxes, each of which contains a sentence and a selection condition.

Sections and subsections are indicated by long, thin "Section Break" and "Subsection Break" boxes. These boxes contain the titles of the sections and subsections.

Rhetorical relations between pieces of text are indicated by arrows drawn between the boxes containing the texts. For example, the arrow between the single-text box and the double-text box in figure 5.1 shows that the author has indicated a rhetorical relation between the texts. The arrows are colour-coded to indicate the type of relation they represent. In the example shown, the green arrow indicates that the two sentences in the double text box are both "elaborations" of the sentence in the single-text box.

Words or phrases that are coreferential appear in the text in the same colour. For example, in a document about diabetes, all references to *insulin-dependent diabetes* could be in red, and all references to *non–insulin-dependent diabetes* in blue. In figure 5.1, the phrase *insulin-dependent diabetes* and two occurrences of the phrase *this condition* are all in red, indicating that they all refer to the same concept.

5.2 Implementation of Generic Authoring Tool Requirements

In this section, I will discuss the design decisions I made while implementing the requirements that are generic to any authoring tool for customizable documents. I implemented all of the "basic" requirements, but not the "advanced" requirements.

5.2.1 Software Development and Maintainability Requirements

Choice of Programming Language

A major decision in the design of the authoring tool was the choice of programming language. I considered Java, Lisp with CLIM, and Tcl/Tk. Each language had advantages and disadvantages, but I chose Tcl/Tk (Ousterhout 1994).

Java seemed like a reasonable choice for the authoring tool, since it is platform-independent and can run over the Internet. However, some of the user-interface objects were only available for full stand-alone Java applications, and not available for use in Java applets (that run over the Internet), so the authoring tool would not be able to run over the Internet after all. Additionally, some of the classes of user-interface objects (e.g., menus, windows) still contained errors, and the next version of Java was still in beta testing when I started to code the authoring tool. Rather than wait an unknown amount of time for the new release, I decided to consider other languages.

Since the majority of the HealthDoc system is written in Lisp, I considered writing the authoring tool in Lisp and CLIM (Common Lisp Interface Manager), an interface-builder for Lisp. Since CLIM is more difficult to use than Tcl/Tk, and Tcl/Tk can also be used to build an interface for Lisp code, Tcl/Tk seemed to be the better choice.

In the end, I decided Tcl/Tk was the best language for implementing the authoring tool. Tcl/Tk (which stands for Tool Command Language/Tool Kit) is a scripting language used to construct graphical user-interfaces for programs written in other languages. It is available on UNIX, DOS, Windows, OS/2, Windows NT, and Macintosh environments and is easy to port between these environments.

There are reference books published for Tcl/Tk (Ousterhout 1994, Welch 1995), and an Internet newsgroup (comp.lang.tcl) dedicated to it, so it is easy to learn the language and to find information about it. The language is stable and well-documented.

Tcl/Tk turned out to be a good language for the initial implementation of the authoring tool because it was designed to allow rapid prototyping of graphical user interfaces. Tcl/Tk was also designed to run compiled code written in other languages, so it can be used to piece together code in multiple languages.

Implementation Platform

The end-users of the HealthDoc authoring tool will be medical writers, who will most likely use PCs or Macintosh computers. However, I chose to implement the authoring tool in a UNIX environment, because that is what was more readily available to me and because that is the environment used by the rest of the HealthDoc system. Since Tcl/Tk is easy to port to other environments, the authoring tool will run on PCs or Macintoshes as well.

Revision Control

I used the RCS revision control system available on UNIX systems. RCS keeps track of all versions of the code in a database, not just the most recent version, so that any previous version can be retrieved. It keeps a history of the changes to each version, the programmer's name, the date, and any additional information the programmer provides.

Code Optimization

Since Tcl/Tk can include code written in other languages, some of the functions and procedures of the authoring tool can be rewritten in the future to optimize them. For the HealthDoc project, some of the authoring tool might be faster if rewritten in Lisp (e.g., for parsing TSL) or in a compiled language like C++ (e.g., for moving text boxes around the screen and redrawing them).

Running over the Web

The authoring tool can be run over the Internet if the end user has an "X11" emulator.

5.2.2 User Interface Requirements

Ease of Use

I believe the authoring tool is reasonably easy to use, especially for inputting text and associating selection conditions with the text. Including linguistic information in the document is more complicated, however.

In this initial implementation, the author must have enough linguistic expertise to differentiate between types of rhetorical relations and between types of coreference. This will need to change in future implementations: the amount of linguistic expertise needed should be reduced.

Reusing Existing Documents

To reuse text from an existing document, the author can cut-and-paste text from the original into the master document using the mouse. The left mouse button is used to copy text into a buffer, and the middle mouse button copies the contents of the buffer into the document.

The authoring tool cannot automatically read in a document in a format other than TSL.

Keyboard Shortcuts

For more advanced users, the authoring tool provides keyboard shortcuts for menu items. One letter of each menu heading is underlined. Holding down the "Alt" key and typing the underlined letter pops up the indicated menu, and the arrow keys can be used to move through the menu. The "Return/Enter" key is used to make the selection. For example, typing "Alt-F" pops up the File menu and then hitting the down arrow will move to "Quit." Pressing "Return/Enter" will quit the authoring tool.

Novice and Expert Modes

I did not implement novice and expert modes, but they would be particularly useful when defining rhetorical relations and coreference. The authoring tool could give hints and advice about which rhetorical relation or which type of coreference the author should specify.

Different Views of the Document

The master documents written so far for the HealthDoc project have been more linear than I initially expected. I first thought that the text boxes and rhetorical-relation arrows would form a fairly complex graph structure, so that it would be beneficial to include different ways of viewing a graph in the authoring tool.

Since experience has shown that the documents are still quite linear, I did not implement different views of the document. However, if the master documents *do* become more complex as the model of a master document develops, I recommend that the authoring tool include different ways of viewing the document, such as a full-document view versus a "fish-eye" view.

5.2.3 Word Processor Functionality Requirements

Entering Text

The author types ordinary text into the boxes on the screen. Cut-and-paste functionality is available by using the mouse: highlighting text with the left mouse button copies the text into a buffer, and clicking with the middle mouse button

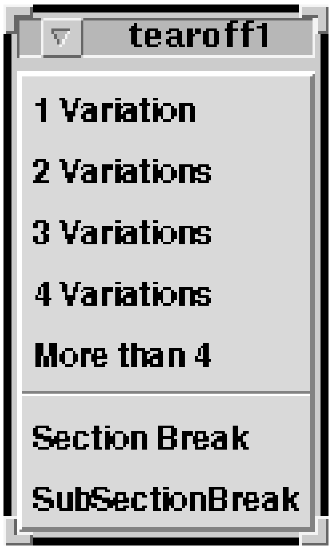


Figure 5.2: Pop-up menu in the main editing area.

pastes the text from the buffer at the current location of the mouse.

If the text is too long to fit in the box, the author can use the scroll bars to view the rest of the text.

To create a new text box, the author clicks on the "New Variation" button, which pops up a menu from which the author chooses the number of variations that will be grouped together. For example, if the author wants to write one text that will be selected for all readers, she chooses "1 Variation" from the menu, and a single text box is drawn. If the author wants to write three different variations of a text, for three different types of audiences, she chooses "3 Variations" from the menu, and a group of three text boxes is drawn.

The author can also draw text boxes on the screen by clicking with the right mouse button inside the main editing area of the authoring tool. A menu pops up at the location of the mouse, and the author can again choose the number of text boxes to be grouped together. The boxes are drawn at the current location of the mouse. Figure 5.2 shows the menu that pops up when the author clicks with the right mouse button inside the editing area.

If the author accidentally chooses the wrong number of variations that she requires, she can delete the box by clicking on the coloured frame of the box and typing “d” (for “delete”). She can then draw another variation-set box with the correct number of variations.

The boxes of text can be manually moved around the screen by clicking on the frame around the box with the left mouse button and dragging with the mouse. When the mouse button is released, the text box is redrawn at the new location of the mouse.

Indicating Document Structure

To indicate section and subsection breaks, the author uses a “Section Break” or “Subsection Break” box, which is a long, thin box on the screen. The author may also put a title for the section or subsection in the box. These boxes are visually distinct from the regular text boxes, to make it visually obvious where the section breaks are.

To create a section break on the screen, the author clicks on the "Section Break" button. A short menu pops up, which lets the author choose either "section" or "subsection", and the section-break box appears on the screen near the top of the portion of the document that is currently visible. The section and subsection boxes can be manually moved by clicking on the frame with the left mouse button and dragging them around the screen.

Alternatively, instead of using the "Section Break" button and menu, the author can click with the right mouse button inside the main editing area of the authoring tool. The menu which pops up includes selections for "Section Break" and "Subsection Break" (see figure 5.2). The section break boxes appear at the current location of the mouse.

File Manipulation

To save an existing document, the author chooses "Save" or "Save As" from the File menu. Documents are saved in TSL format, and have the filename extension ".tsl". To open an existing TSL document, the author chooses "Open" from the File menu.

The "Open" command pops up a file-selection window, to prompt the author for the name of the file to open. The user can either type in the filename directly, or can browse the file system to find the file. Figure 5.3 shows the window used for

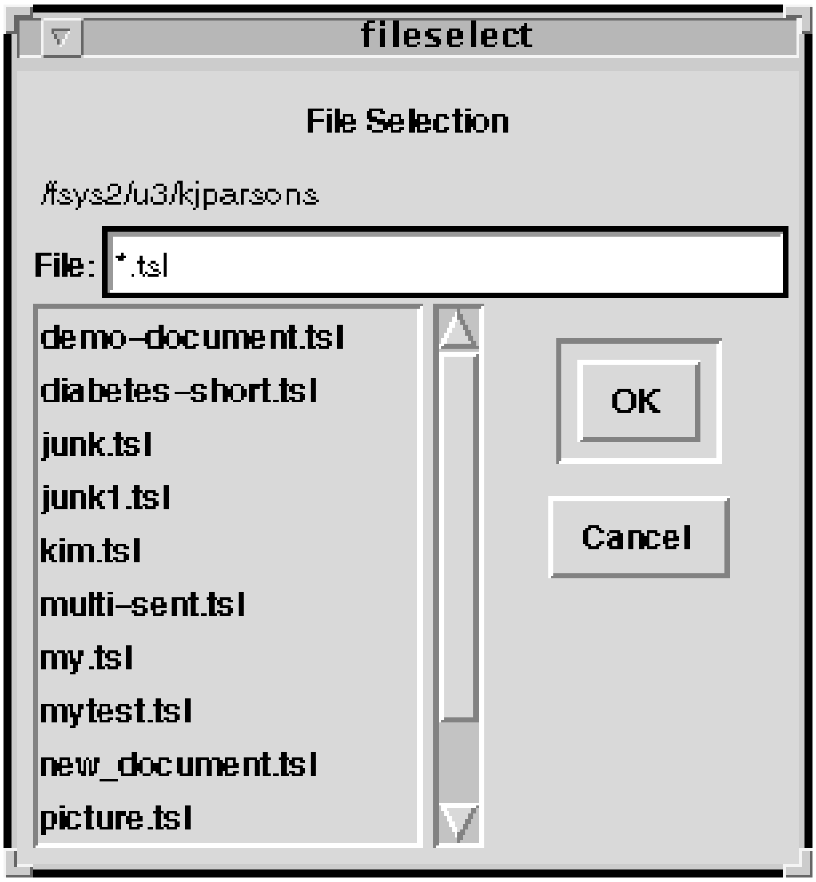


Figure 5.3: Window used to select a file.

file selection.

The "Save" command is a quick way to save the document. If the current document has a filename, the Save command uses that name. If the current document is unnamed, the Save command uses "new_document.tsl" as the filename.

The "Save As" command prompts the author to name the document. This command uses the same file selection window as the "Open" command (see figure 5.3).

A Spelling Checker

The authoring tool currently does not include a spelling checker, although one could be added by adapting an existing spelling checker such as *spell* or *ispell* to check the spelling of the English text in the variation-set boxes.

Text Mark-up

There is no explicit support for including text mark-up such as italics or different fonts in the authoring tool. This is because the rest of the HealthDoc system (i.e., the parser and the sentence planner) manipulates text only. Any extra text mark-up will need to be stored in the TSL as an additional representation of the text. For example, a sentence could be stored as ordinary text, as an SPL representation of the sentence, or as an HTML representation of the sentence.

The formatting for the sentence will need to appear in the supplementary representations of the text. However, the authoring tool does not currently allow the author to manipulate any representation other than the ordinary text. To add formatting to another representation (e.g., to the HTML), the author would need to manually edit the TSL.

Including Non-Textual Media

Non-textual media such as illustrations can be included in a master document, but they will not be repaired by the current sentence planner.

There is currently no specified format for including other media in the document. I am assuming that the software that performs the selection and then repairs the document will be able to recognize that a selection is a filename instead of a sentence, so that it can include other files in the document.

For example, to include an illustration in the document, the format might be "$FILENAME(picture.ps)" inside the text field of a variation box. Figure 5.4 shows how an author might indicate that she would like to include either the image *high.ps* (for readers who have a high technical-reading level) or the image *low.ps* (for readers who do not have a high technical-reading level) in a document.

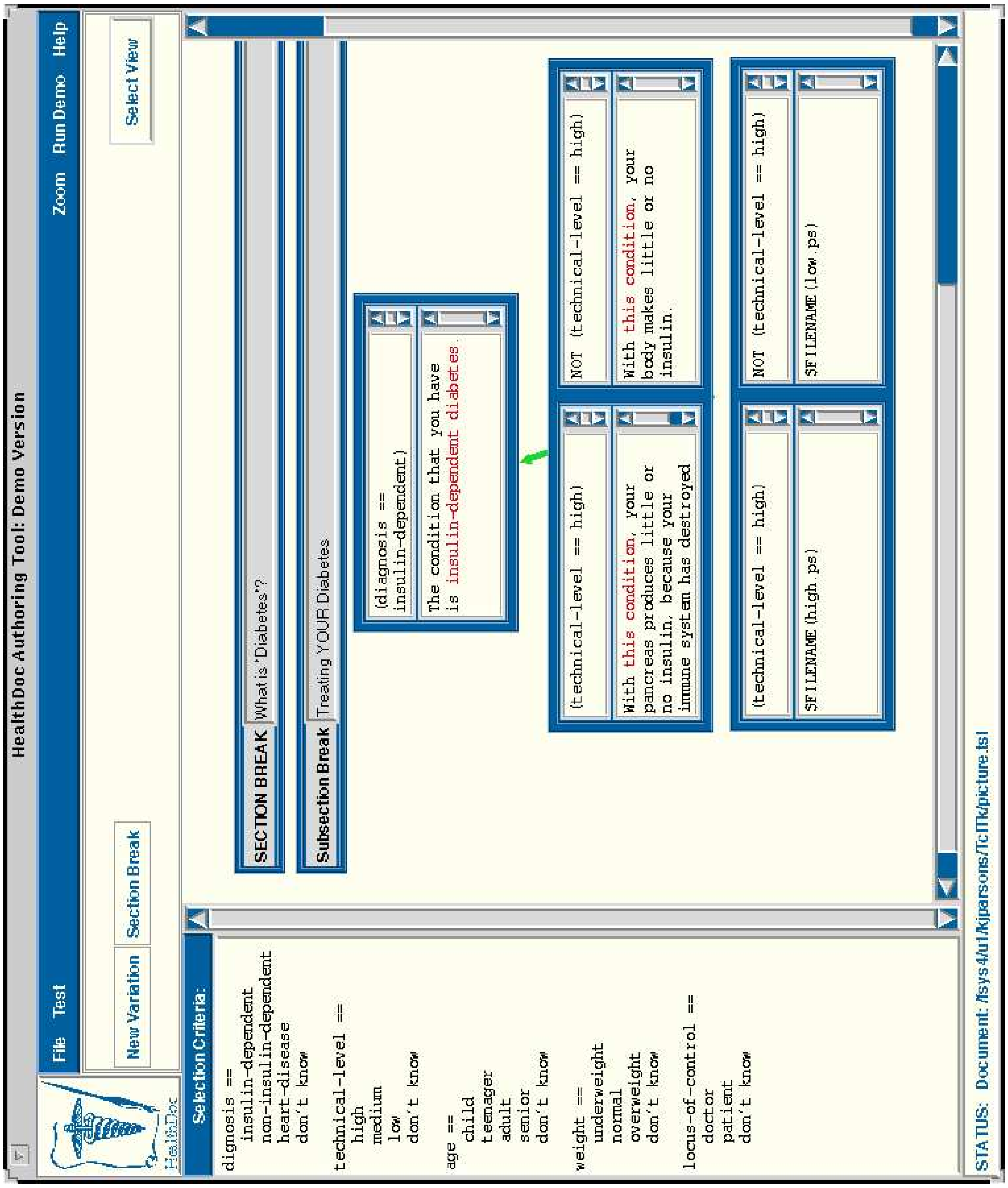


Figure 5.4: Including an illustration in the master document.

5.2.4 Customization Requirements

Listing the Selection Criteria

On the left side of the authoring tool interface is a section labeled "Selection Criteria." This is the area where the author can list all the variables being used to tailor the document and the possible values of these variables. The author can edit any part of the selection criteria, to add or delete variables or values. Thus, the author has precise control over the variables and values used to tailor the document. The list of selection criteria also provides a clear reference list of the variables being used for tailoring.

An example of the selection criteria being used to tailor a document about diabetes is visible in figure 5.1.

There is no specified format for the selection criteria yet. I suggest the format shown in figure 5.1: the variables are listed, and the possible values for the variable are listed below, indented for clarity. This format is easy for the author to edit and for the selection engine to parse.

The "Everyone" and "Don't Know" Cases

Since there is no specified format for the selection criteria yet, there is no rule specifying how the author should indicate that a piece of text should be selected for all readers or that a piece of text should be selected when the value of a variable is unknown.

Currently, if the author wishes to indicate that a piece of text should be selected for all readers, she can enter the word "ALL" in the condition field of the variation box. To indicate that a piece of text should be selected when the value of a variable is unknown, the author can include the value "don't know" in the list of possible selection criteria, and then use this value when constructing conditions. For example, if a default piece of text is to be used when a reader's technical-reading level is unknown, the condition for that piece of text might be "(technical-level == don't know)".

Attaching Conditions to the Text

To indicate the types of readers for whom a piece of text is relevant, the author places the condition and the text together in a text box. For example, the text boxes in figure 5.1 indicate that the sentence *The condition that you have is insulin-dependent diabetes.* will be selected for readers who have insulin-dependent diabetes. Then, the text *With this condition, your pancreas produces little or no ...* will be selected for readers who desire a high level of technical detail *or* the text *With this condition, your body makes little or no insulin.* will be selected for all other readers.

The conditions that the author enters in the condition field of the text box must conform to a pre-specified format, so that the automatic consistency checking and the software that selects pieces from the master document can read them. However, there is no specified format for the conditions yet in the HealthDoc project. The author can enter the conditions in any format she likes.

The format that will be used to specify conditions will likely depend on the software used to select texts from the master document: if the selection software is written in C or C++, the format of the conditions would be the format used by C and C++, such as "(technical_level == high)". If the selection software is written in Lisp, the conditions would use a Lisp-like format, such as "(= technical_level high)".

Indicating Variation Sets

When the author includes in the master document a set of texts that are all variations of each other, then these texts are grouped together on the screen. For example, the larger box in figure 5.1 contains two sentences that are variations of each other, one which is highly technical, and one which is non-technical. Each of these sentences has a condition attached.

The author can choose any number of variations to include in a variation set. However, if there are more than four variations in the set, it is difficulty to see them all simultaneously, and the author must scroll back and forth frequently.

Including Variables and Formulas

To include variables (e.g., $NAME) or formulas (e.g., $LIFE-EXPECTANCY – $AGE) in conditions or in the text, the author will enter them according to a specified format.

No specified format exists as yet, so I recommend that variables and formulas be preceded by a dollar-sign ("$"), as shown in the examples in the preceding paragraph. The selection engine will need to find these variables and formulas in the conditions and in the text, and will need to be able to find the values of the variables in the patient's medical record.

Testing the Document

The author cannot yet use only the authoring tool to "test" the document to see the text that would be selected for a given reader.[1] The next version of the authoring tool will need to be able to parse the selection conditions so that an author can test a document using only the authoring tool.

However, in the interim, the author can use the authoring tool in combination with other parts of the HealthDoc system to view the text for a given reader: she need only save the document and then use HealthDoc's selection engine to set the

[1]Appendix A shows a section of a master document about diabetes and the selected text for three sample patients.

values of some variables and view the selected text.

The Consistency Checker

I have not implemented a consistency checker in this prototype version of the authoring tool. Once a specified format for the selection criteria and conditions exists, a consistency checker could help ensure that the variables, values, and conditions entered by the author conform to this specified format.

5.3 Implementation of Requirements Specific to the HealthDoc Authoring Tool

5.3.1 Rhetorical Relation Specification Requirements

Rhetorical relations are indicated on the screen by coloured arrows between text boxes. The colour of the arrow indicates the type of relation, and the direction of the arrow indicates the direction of the relation.

To draw a rhetorical relation on the screen, the author clicks on the frame of the first text box, then types the number "1". She then clicks on the frame of the second text box, and types the number "2". A black arrow will be drawn from the first box to the second one.

Figure 5.5: The list of rhetorical relations.

Once the arrow is drawn, a window pops up with a list of all the possible rhetorical relations that the author can choose. Figure 5.5 shows the window containing the possible relations, listed alphabetically. The author chooses one of the relations in the list by double-clicking on it, or by clicking once and hitting Return.

Once the author chooses a relation, the arrow changes colour to reflect the relation that was chosen.

There can be only one rhetorical-relation arrow drawn between any pair of boxes on the screen, whether the box is a single text box containing only one text-condition pair or a multiple text box containing a set of variations. As a simplification in the current version of HealthDoc, if there is a set of text boxes grouped together into a variation set, then every text in the set must have the same rhetorical relation to other texts in the document. As the working model of the master document evolves, the authoring tool may need to be updated to include the ability for individual texts in a variation set to have different rhetorical relations to other texts.

Automatically Determining Rhetorical Relations

I did not attempt to automatically determine the rhetorical relations between sentences. The author must specify all of the rhetorical relations manually.

5.3.2 Coreference Specification Requirements

The authoring tool must keep track of groups of words and phrases that are coreferential. All phrases in a group of coreferences are coloured the same in the text. For example, in a document about smoking cessation, all references to *lung cancer* could be coloured green, and all references to *emphysema* could be coloured red.

The author indicates that she would like to mark a phrase as being coreferential with something else by highlighting the phrase with the mouse and then clicking the right mouse button. A window such as the one in figure 5.6 pops up.

The phrase currently under consideration is shown at the top of the window, so that the author can verify that the phrase was selected correctly. In the example in figure 5.6, the current phrase is *this condition*.

The list of phrases in the left portion of the window is the set of groups of coreferential words/phrases that have been marked thus far. The first word that was put into the group is one used in the list as an "index." In the example, only one group of coreferences (referring to the phrase *insulin-dependent diabetes*) has been marked thus far. The list in the right portion of the window is the list of possible types of coreference.

If the phrase under consideration is not part of one of the groups already listed, the author chooses "Not Listed" from both columns and clicks "Okay." The current

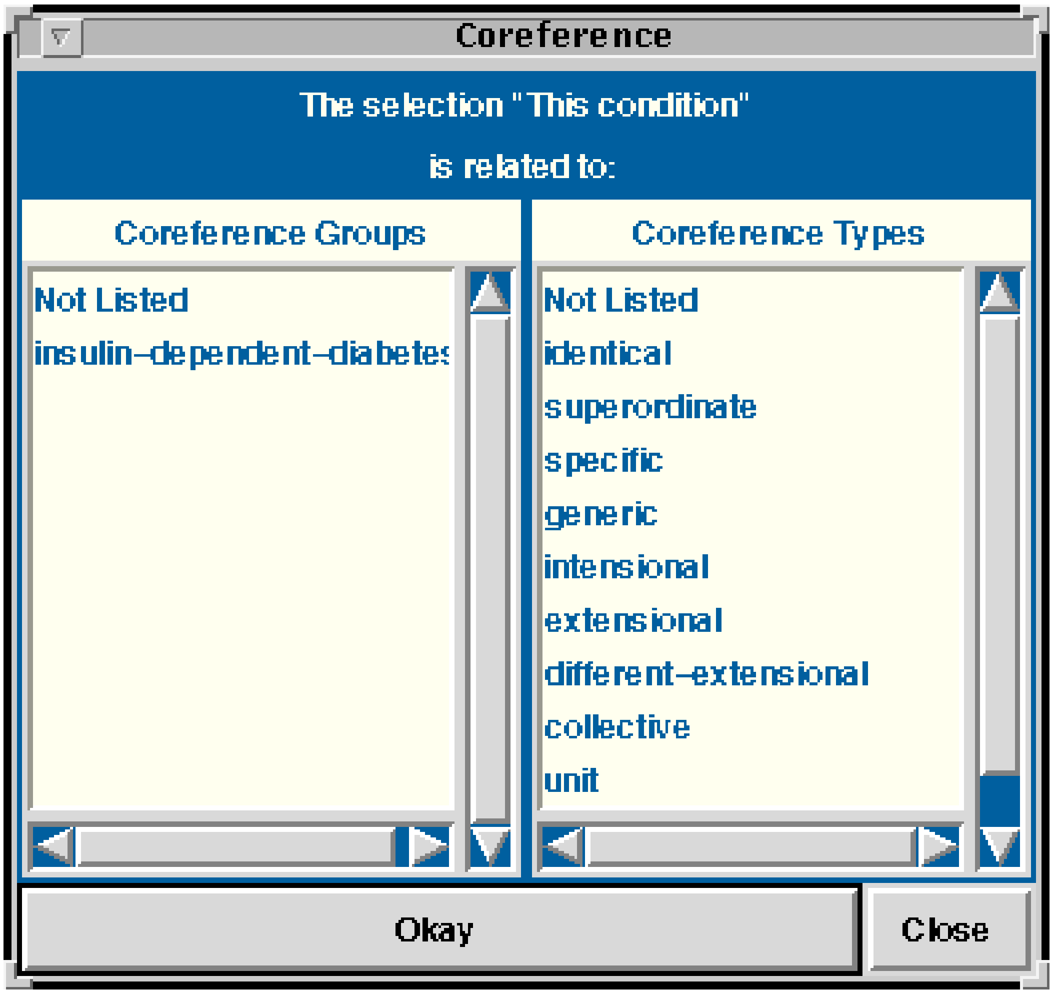


Figure 5.6: Window to indicate words or phrases that are coreferential.

phrase is added to the list of groups and the phrase changes colour in the text.

If the phrase under consideration is part of a group that is already in the coreference window, the author chooses the group that it belongs to, and chooses the type of coreference involved. For example, in figure 5.6, the author would choose "insulin-dependent diabetes" as the group, and "generic" as the type of coreference, because *this condition* is a generic reference to *insulin-dependent diabetes.* Clicking on "Okay" stores the coreference pair internally and changes the colour of the phrase in the text to match the colour of the group.

The type of coreference that the author chooses indicates the manner in which the current phrase and the "index" phrase of the group are related. If the author wishes to specify a coreference relation with a word that is not the index word of the group, she needs to create a separate group of coreferences with this other word as the index.

Automatically Determining Coreference

I did not attempt to automatically determine coreferential phrases in the authoring tool. The author must specify the coreferences by hand.

5.3.3 Requirements for an Interface to the Parser

At the current stage of the HealthDoc project, the parser and the software to convert the English text to an internal semantic representation (SPL) are not ready to be integrated into the authoring tool. The design of the tool, however, will allow reasonably smooth integration when the rest of the software is ready.

5.3.4 Input and Output Requirements

The authoring tool works with files that are in TSL format. All input files must be in this format, and all output files are saved in this format.

The specification of TSL is evolving, so the authoring tool will need to be updated as new versions of TSL are developed.

5.4 Testing and Evaluation

5.4.1 Testing the Code

I have tested the authoring tool by testing individual pieces and by using it to create a master document for HealthDoc. I used the tool to create the TSL for the first section of a document about diabetes. The text of this master document is available on the HealthDoc Web Page[2] and is shown in appendix A, along with the text that would be selected for some sample patients.

5.4.2 Evaluation by Technical Writers

I also wrote a tutorial about the authoring tool (shown in appendix B), which I gave to some health-communication students of Victor Strecher in the Health Media Research Lab at the Comprehensive Cancer Center, University of Michigan. Since this group uses tailored health-education materials, the writing students were already familiar with the concept of master documents and customized documents.

General Comments on the Authoring Tool

The students who evaluated the authoring tool were very pleased with it, and said they would definitely use it to author master documents if they had the chance.

The most important benefit they saw in using the tool is that it gives them a way to organize the variations in their master documents. Until now, the master documents used by the Health Media Research Lab were all authored by hand. The students explained that it was extremely difficult for them to keep track of all of the variations of the document. They said it was definitely worth the learning

[2]The HealthDoc homepage is at http://logos.uwaterloo.ca/~healthdo/ and the diabetes document is at http://logos.uwaterloo.ca/~healthdo/About/Demos/diabetes.html.

curve to be able to use a tool that would help them organize variation sets on the screen.

The students had not previously used rhetorical relations or coreference when constructing their master documents. They were very interested in learning how to use knowledge of rhetorical relations to better structure a document.

Suggested Improvements to the Authoring Tool

The recommendations that the health-communication students had for the authoring tool were the following:

- Include on-line help. For example, include explanations of the differences between the types of rhetorical relations and between the types of coreference.
- As part of the explanation of the differences between the types of rhetorical relations and coreference, include examples of each of the different types.
- To make the user-interface a little "friendlier," include little "pop-up" messages that appear as you move the mouse over top of items on the screen, like Microsoft Word does. For example, as you move the mouse over the "New Variation" button, a little note would pop up, saying "Create a new variation set on the screen". The little note would disappear when you

move the mouse off the "New Variation" button.

- Include the ability to "name" sections, subsections, and variation sets, so that they can be reused in other documents without having to type them in again. For example, if a section about the dangers of smoking is to be included in several master documents, then the section could be named (e.g., "Smoking-dangers-section") the first time it's written and then later master documents can just specify that the "Smoking-dangers-section" should be included in the current document.

  Eventually this would result in a library of pre-written, customizable chunks of text that can be re-arranged and re-used as desired.

- Develop a format for the selection criteria and for the conditions that is easy for "non-computer-people" to understand and to use.

- Leave the list of possible rhetorical relations up on the screen permanently, as a reference for the author.

- When an author is specifying the rhetorical relation between two variation-sets, if certain relations would be invalid choices (e.g., because of the direction of the relation or because of the order of the boxes chosen), then disable those relations in the list, so that the author can't choose a relation that is invalid.

- If an author is specifying coreferences and chooses two phrases that are somehow "obviously" incorrect, then give an error message and don't allow the author to mark the phrases as coreferential. Try to prevent the author from making "stupid" mistakes.

Complaints about the Authoring Tool

- It is too hard to see all of the variations in a variation set if there are more than four or five variations in the set.

- The students found a bug in the scrollbars, which I have now fixed.

- There's no simple way to add or delete a variation from a variation set. If you want to add a new variation to an existing variation set, you have to create a new variation set, copy the text from the old variation set into the new one, and then delete the old variation set. There should be a command to add and delete single variations.

Chapter 6

Evaluation and Conclusions

6.1 Evaluation

6.1.1 The Usability of the Authoring Tool

To test the basic functionality of the authoring tool, I used it to author the first section of a master document about diabetes. The text of this document is available on the HealthDoc homepage (http://logos.uwaterloo.ca/~healthdo/About/Demos/diabetes.html) and in appendix A.

The authoring tool is reasonably easy to use, but would be more useful if it had some additional functionality, such as different views (e.g., zoom, fish-eye) or a spelling checker.

The evaluation of the authoring tool by some health-communication students (section 5.4.2) supported my opinion that the tool is easy to learn and easy to use. The students were very pleased with the authoring tool and were eager to use it to create their master documents.

The recommendations from these students were essentially:

- to include on-line help; for example, information about the differences between the different types of rhetorical relations and between the different types of coreference.
- to add "intelligence" to the tool to prevent the author from making "stupid" mistakes when specifying rhetorical relations and coreference.
- to add the ability to name pieces of the document, so that the pieces can be reused in other documents.
- to add some finishing touches to the user-interface to make it look more professional.

6.1.2 Limitations of the Authoring Tool

There are several limitations of the current authoring tool that need to be corrected in the next implementation:

- There is no specified format for the list of selection criteria and conditions. The author can enter them in any format she likes. There should be a specified format to which the author adheres.

- There is also no specified format for including variables in the text (e.g., $NAME) or for importing other files into the document (e.g., including postscript files as illustrations).

- The selections can be made only at the sentence level. The author should be able to specify selection conditions for pieces of text at any level: word, phrase, sentence, paragraph, subsection, or section.

- There can be only one rhetorical relation between any pair of boxes in the document, so all variations in a "variation set" (a set of text segments that are variations of each other) must have the same rhetorical relation to any other variation set. In future implementations the author might wish to specify different relations between any pair of variations in different variation sets.

- A word or phrase in the text can only be part of one group of coreferences. Future implementations of the authoring tool should permit a phrase to be part of more than one group.

- Because the authoring tool uses colour to indicate rhetorical relations and coreference, it would be difficult to author a master document using a monochrome monitor.

- If there are more than four variations in a variation set, then the set extends past the edge of the screen, so it is difficult to view all the variations in a large variation set simultaneously.

- When the author is specifying a rhetorical relation, the list of possible relations is not colour-coded to match the colour of the arrows on the screen, so the author has no easy way to correlate the colour of the arrows with the type of relation they represent.

- If the author wants to add or delete a single variation from a variation set, she can't just add or delete a single text box. She needs to delete the entire variation-set box and then create a new variation-set box of the correct size.

- The author cannot yet use the authoring tool to "test" the document to view the text that would be selected for an individual patient. She needs to use the authoring tool in combination with HealthDoc's selection engine to test the document.

6.1.3 Choice of Programming Language

Tcl/Tk was a good choice of programming language for an initial prototype of the authoring tool, but it was not powerful enough to do everything that I wanted to try. I needed to install extensions to the language (Lam 1997) in order to implement parts of the functionality of the authoring tool. This means that the tool is no longer easy to port to new platforms because the extensions must also be present on the new platform.

Because Tcl/Tk is a scripting language, parts of the implementation of the authoring tool are more complicated than should be necessary (e.g., the parsing of files in TSL format). I recommend using a higher-level language for the next implementation.

If the idea of small boxes on the screen is to be re-used in the next implementation of the authoring tool, then I think it would be best to use an object-oriented programming language for implementation. The object-oriented-programming paradigm is more suited to the manipulation of visual objects on the screen than the procedural-programming paradigm.

6.1.4 Suggestions for Future Work

My suggestions for future work on the authoring tool include the following:

- Rewrite the tool in an object-oriented programming language such as Java or C++.

- Include the ability to specify selection conditions for pieces of text at *any* level of document granularity, from an individual word up to an entire section.

- Establish a format that will be used to enter the selection criteria and the conditions, so that the software that performs the selection can parse the selection criteria and conditions correctly.

- Establish a format for including variables (e.g., $NAME) in the text and for including other files (e.g., illustrations) in the document.

- Use techniques from natural language understanding to automatically (or semi-automatically) determine the rhetorical relations between pieces of text and the coreferential relations between different words and phrases. This is an open problem in natural language understanding research.

- Include a "consistency checker" to help the author ensure that the selection conditions specified are consistent and correct according to the format being used.

- Provide a simple way for the author to "test" selections from the document and to view the text that would be selected for a given reader.
- Implement the other "Advanced Requirements" and "Optional Enhancements" listed in Chapter 4. The most important of these extensions are:
  - Provide different views of the document, such as zooming or fish-eye views.
  - Include a spelling checker.
  - Include different modes of operation, depending on the expertise of the author (e.g., "novice" and "expert" modes).
- Include a "formatting-repair" module to repair the document formatting (e.g., fonts) after the pieces of text are selected from the master document.
- Include formal modes of rhetoric and argumentation to assist the author in communicating more effectively. This could involve including schemas for various rhetorical models.

6.2 Research Contributions

The authoring tool described in this thesis is used to author customizable documents for the HealthDoc system. It allows the author to enter variations of each piece of the document and to specify the conditions for selecting a particular variation. It also permits the author to specify the rhetorical relations between the pieces of text and to indicate words and phrases that are coreferential with each other.

This is the first software tool developed to create "master documents" for the HealthDoc tailoring system. Other authoring tools focus on authoring hypertext or multimedia presentations; this tool focuses on constructing English-text documents that can be customized for a particular reader. As seen in chapter 2, tailored documentation has applications in education, technical documentation, and marketing.

The HealthDoc authoring tool is one of a small set of tools that allow an author to explicitly indicate the rhetorical relations between pieces of text in the document. It is the only tool to include a way for the author to specify that groups of words or phrases are coreferential. By providing a method of specifying explicit linguistic information, the authoring tool provides the author with a way to more effectively organize the structure of the document.

The thesis also argues that there are certain benefits to using an authoring tool to create customizable documents. In particular, this authoring tool supports natural language generation using the model of generation by selection and repair. Pieces of text can be re-used in multiple documents, and the linguistic information can be used to repair the selected pieces of text.

In addition to demonstrating the direct benefits to the HealthDoc project, I have suggested how an authoring tool might be used to create customizable documents in other domains. In particular, I have specified what a programmer should do in order to design an authoring tool (sections 4.2 and 4.3), and I have shown a full walk-through for an author, describing what he must specify in order to use the authoring tool to perform text customization (section 4.1.1).

I have built a working version of an authoring tool for customizable documents, which is a reference for others who may want to design additional authoring tools for other applications. Moreover, I have provided a list of advanced requirements and optional enhancements to provide suggestions for alternate designs.

I have also presented a contrast to the snippet-based approach (Strecher et al. 1994), which requires the author to be responsible for consistency and coherence checking, compared to an authoring tool which can delegate these tasks to a sentence planner. Furthermore, I have provided an improvement over the standard approach of hand-authoring master documents, as supported by comments from the writers who evaluated the tool (section 5.4.2).

The authoring tool I have written can be used to author documents on any topic (not only health education) and it can be used both by those authors who wish to tailor the style of a document and by those who do not. An author who does not wish to include linguistic information in a master document is not required to do so. Since it may be difficult to find authors who have both the subject-domain expertise and the linguistic expertise necessary to create a master document, I have presented some suggestions to help reduce the amount of linguistic expertise needed (sections 4.3 and 6.1.4).

6.3 Conclusions

The authoring tool described in this thesis is a first step in providing support for the authoring of customizable documents for the HealthDoc project; subsequent implementations could be used to author customizable documents in other domains as well. The design of this initial implementation of the authoring tool has raised some issues to be resolved in future implementations, but the current authoring tool can be used to create master documents that support the HealthDoc model of natural language generation based on generation by selection and repair.

Appendix A

Sample Master Document and Selections

This appendix contains the introductory section of a master document about the medical condition diabetes. The master document containing information for all patients is shown in figures A.1 and A.2, and then the selected text for some individual patients is given in section A.1.

A.1 Selected Working Documents

The following sections show the text that would be selected for some individual patients. The first section gives the text for a patient with non–insulin-dependent

Basic information

What is diabetes?

basic (for everyone)
Diabetes is a group of conditions in which glucose levels (sugar levels) are abnormally high. Diabetes prevents your body from using glucose, an important product of digestion, for energy.

The two types of diabetes

basic	
There are two main types of diabetes. One type is insulin-dependent, also known as type I diabetes, and the other is non-insulin-dependent, also called type II diabetes.	
insulin-dependent	***non-insulin-dependent***
The condition that you have is insulin-dependent diabetes.	The condition that you have is non-insulin-dependent diabetes.
low-technical	***moderately technical OR highly technical***
Insulin-dependent and non-insulin-dependent diabetes have different causes and treatments.	Insulin-dependent and non-insulin-dependent diabetes are different disorders, so that the causes, short-term effects and treatments for the two types differ. However, both types can cause the same long-term health problems.

Figure A.1: A Section from a Master Document about Diabetes.

low-technical AND insulin-dependent	***(moderately technical OR highly technical) AND insulin-dependent***	***low-technical AND non-insulin-dependent***	***(moderately technical OR highly technical) AND non-insulin-dependent***
With insulin-dependent diabetes, your body makes little or no insulin.	With insulin-dependent diabetes, your pancreas produces little or no insulin. The pancreas is not able to produce insulin, because your body's immune system has destroyed the insulin-producing cells.	With non-insulin-dependent diabetes, your body makes insulin, but can't use it well.	With non-insulin-dependent diabetes, your pancreas may still produce some insulin, but your body cells can't efficiently use it, because they are resistant to the insulin.

insulin-dependent	***non-insulin-dependent***
Insulin-dependent diabetes most often develops in children and young adults. Sometimes, people over age forty get insulin-dependent diabetes, but it usually begins at younger ages.	Non-insulin-dependent diabetes is the most common type of diabetes. It usually develops in adults over age forty, and is most common in those who are overweight.

Figure A.2: A Section from a Master Document about Diabetes.

diabetes and a low technical reading level. The second section shows the text that this patient would receive if she had a high technical reading level instead of a low one.

The third section shows the text that would be selected for a patient with insulin-dependent diabetes and a high technical reading level.

A.1.1 Patient with Non–insulin-dependent Diabetes and a Low Technical Reading Level

Basic information

What is diabetes?

Diabetes is a group of conditions in which glucose levels (sugar levels) are abnormally high. Diabetes prevents your body from using glucose, an important product of digestion, for energy.

The two types of diabetes

There are two main types of diabetes. One type is insulin-dependent, also known as type I diabetes, and the other is non–insulin-dependent, also called type II diabetes. The condition that you have is non–insulin-dependent diabetes. Insulin-dependent and non–insulin-dependent diabetes have different causes and treatments. With non–insulin-dependent diabetes, your body makes insulin, but can't use it well.

Non–insulin-dependent diabetes is the most common type of diabetes. It usually develops in adults over age forty, and is most common in those who are overweight.

A.1.2 Patient with Non–insulin-dependent Diabetes and a High Technical Reading Level

Basic information

What is diabetes?

Diabetes is a group of conditions in which glucose levels (sugar levels) are abnormally high. Diabetes prevents your body from using glucose, an important product of digestion, for energy.

The two types of diabetes

There are two main types of diabetes. One type is insulin-dependent, also known as type I diabetes, and the other is non–insulin-dependent, also called type II diabetes. The condition that you have is non–insulin-dependent diabetes. Insulin-dependent and non–insulin-dependent diabetes are different disorders, so that the causes, short-term effects and treatments for the two types differ. However, both types can cause the same long-term health problems. With non–insulin- dependent diabetes, your pancreas may still produce some insulin, but your body cells can't efficiently use it, because they are resistant to the insulin.

Non–insulin-dependent diabetes is the most common type of diabetes. It usually develops in adults over age forty, and is most common in those who are overweight.

A.1.3 Patient with Insulin-dependent Diabetes and a High Technical Reading Level

Basic information

What is diabetes?

Diabetes is a group of conditions in which glucose levels (sugar levels) are abnormally high. Diabetes prevents your body from using glucose, an important product of digestion, for energy.

The two types of diabetes

There are two main types of diabetes. One type is insulin-dependent, also known as type I diabetes, and the other is non–insulin-dependent, also called type II diabetes. The condition that you have is insulin-dependent diabetes. Insulin-dependent and non–insulin-dependent diabetes are different disorders, so that the causes, short-term effects and treatments for the two types differ. However, both types can cause the same long-term health problems. With insulin-dependent diabetes, your pancreas produces little or no insulin. The pancreas is not able to produce insulin, because your body's immune system has destroyed the insulin-producing cells.

Insulin-dependent diabetes most often develops in children and young adults. Sometimes, people over age forty get insulin-dependent diabetes, but it usually begins at younger ages.

Appendix B

Authoring Tool Introductory Tutorial

B.1 Introduction

This tutorial provides an introduction to the HealthDoc authoring tool. It teaches a new user how to enter the text of a master document using the authoring tool, how to indicate selection conditions for the pieces of text, how to indicate rhetorical relations between pieces of text, and how to indicate words and phrases in the text that are coreferential.

B.2 Installation

B.2.1 Hardware Requirements

- You need a colour monitor.
- The authoring tool is written in Tcl/Tk, so you must have Tcl and Tk installed on your system.
- If you don't have Tcl/Tk installed on your system, you can run the authoring tool remotely on HealthDoc's "logos" machine, if you have X-Windows on your machine.

  Before you run remotely on logos, you must do the following steps:

  1. In one of your machine's x-terms, type "xhost logos.uwaterloo.ca". Then type "echo $DISPLAY". Record the host display name that the computer shows as a result (it will look something like "machine. domain:0.0").
  2. In the x-term that is connected to logos, login to logos, then type "setenv DISPLAY host-display-name:0.0".

B.2.2 Required Software Files

The following files *must* all be present:

- at.tcl
- fileselection.tcl
- coref.txt
- relations.txt
- health2.xbm

The following files must be present if you wish to go through the tutorial:

- demo-document.tsl
- manual.ps
- tutorial.ps

B.3 Running the Authoring Tool

The main executable program is in the file "`at.tcl`".

There are two ways to run the authoring tool:

1. *Recommended way:* type `at.tcl` or `./at.tcl` at the prompt. Try `at.tcl` first, and if that doesn't work, then try `./at.tcl`. The difference between these two is how your PATH environment variable is set: if it looks for executables in the current directory first, then `at.tcl` will work; otherwise, you explicitly indicate that the executable is in the current directory by adding the "./" before the `at.tcl`.

2. *For those who want to see the "guts":* Type "`wish`" at the prompt, which runs the command-line interpreter for Tcl/Tk. Inside the interpreter, type "`source at.tcl`" to run the authoring tool. Inside the interpreter, you can view the values of variables as the authoring tool is running. (See the documentation for the Tcl/Tk language and the Wish interpreter for more information.) Using `wish` is useful for debugging or for adding extensions to the tool, but isn't necessary for those who just want to *use* the authoring tool.

B.4 Starting the Tutorial

Click on the "`File`" menu item, and choose "`Open`". A new window will pop up, where you can select a file to open.

Open the file "demo-document.tsl" by double-clicking on "demo-document.tsl" in the list of files. The window for selecting files will disappear, and the demo document will appear in the authoring tool.

B.5 Look Before You Leap...

Before making any changes to the demo document, let's take a look at what's already there...

You may want to re-size the window so that more of the document is visible at once. You can also use the scrollbars to move around in the document.

B.5.1 What is All this Stuff?

The authoring tool has two main areas: the left-hand portion of the screen is labeled "Selection Criteria" and contains a list of the variables being used to tailor the document; the right-hand portion of the screen contains the master document that you're editing.

The Selection Criteria

The "Selection Criteria" area lists the following variables and values:

```
tech-reading-level ==
   high
```

```
   medium
   low
   don't-know
reader-role ==
   medical-professional
   computer-scientist
   don't-know
age ==
   child
   adult
   don't know
```

This indicates that you are going to tailor the document on the basis of the reader's technical reading level, role, and age.

The Main Editing Area

The main editing area in the right-hand portion of the screen shows one section of a master document about customized documents.

The title of this section is "*Introduction to Customization*". The section contains two subsections, titled "*Applications of Customization*" and "*How to Create Customized Documents*".

The subsection titled "*Applications of Customization*" contains two variation sets. The first variation set contains only one variation, starting "*Readers who receive ...*". The second variation set contains two variations, one for medical professionals, starting "*Medical Practitioners can use ...*", and one for computer scientists, starting "*Computer companies can use ...*".

The subsection titled "*How to Create Customized Documents*" contains one variation set, and this variation set contains two variations. The first variation is for readers who have a high technical-reading level and the other variation is for readers who don't have a high technical-reading level.

The Variation-Set Boxes

The text of the master document is entered in the "Variation-Set boxes". Each box can contain multiple variation boxes (as many as you'd like). For example, the demo document contains three variation set boxes. The first one contains only one variation, and the other two each contain two variations.

Each variation box consists of two fields: the upper (smaller) field contains the condition for when the text will be selected, and the lower (larger) field contains the text of the variation.

What Does the Arrow Mean?

You will see an arrow drawn between the two variation sets in the subsection "*Applications of Customization*". This arrow indicates that there is a rhetorical relation specified between those two variation sets. In this particular case, the relation is "enablement", indicating that the sentences in the lower variation set are both "enablements" of the sentence in the higher variation set.

Why are Some of the Words Coloured?

Words that are coloured the same in the text are coreferential with each other. For example, in the first subsection, the words "*document*", "*tailored documents*", and "*tailored product documentation*" are all the same colour because they are all coreferential with each other.

B.6 Editing a Master Document

You will now make some changes to the demo document to learn how to edit a master document.

B.6.1 Moving Boxes Around the Screen

You are going to add another variation set to the first subsection (the "*Applications of Customization*" subsection). So, you need to move the second subsection ("*How to Create Customized Documents*") down.

To move a variation-set box or a section/subsection-break box around the screen, click on the coloured frame of the box with the left mouse button and hold the button down.

While holding the button down, you can move the mouse around. You will see the mouse cursor (an arrow) move, but the box will not move until you let go of the mouse button.

Once you release the mouse button, the box will be re-drawn at the current location of the mouse.

You can drag a box past the edges of the window if you'd like. When the box is redrawn, the scrollbars will change to indicate that there is now part of the document that is past the edge of the currently visible area.

Boxes are re-drawn so that the centre of the bottom of the box is at the location of the mouse.

Move the second subsection-break box and the variation-set box down, so that there will be room for a new variation set in the first subsection.

B.6.2 Creating A New Variation Set

You will now add a new variation set to the first subsection. This variation set will contain three variations: for (age == child), (age == adult), and (age == senior).

There are two ways to draw a new variation set on the screen:

1. Click on the "`New Variation`" button near the top of the window. This will pop up a menu with the choices "`1 Variation`" through to "`4 Variations`" and "`More than 4`".

   Since you want to enter a variation set that contains three sentences that are variations of each other, choose "`3 Variations`".

   When you use this method of drawing variation sets, they will be placed on the screen at the top of the currently-visible area. You can move them around by dragging them (see section B.6.1).

2. The other way to draw variation sets is to click inside the editing area with the right-most mouse button and hold the button down. A menu will appear, from which you can choose from 1 to 4 Variations, or more than 4 variations.

   Choose "`3 Variations`".

   When you use this method of drawing variation sets, they will appear on the screen at the position where the mouse was when you clicked with the right mouse button to create the variation set. You can move them by dragging them (see section B.6.1).

The variations to enter are the following:

```
Variation 1:

   condition: (age == child)

   text: This is a really cool idea!


Variation 2:

   condition ((age == adult) || (age == don't-know))

   text: This will save you time: you don't need to wade through

         irrelevant information.


Variation 3:

   condition: (age == senior)

   text: This will be useful for seniors who have difficulty

         reading large amounts of text in small type.
```

Enter these conditions and texts in the variation-set box you created.

B.6.3 Creating a New Section or Subsection Break

You will now add a new subsection break to the current document.

There are two ways to draw a new section or subsection break on the screen:

1. Click on the "`Section Break`" button near the top of the window. This will pop up a menu with the choices "`Section`" or "`Subsection`".

   Since you want to add a new subsection break, choose "`Subsection`".

   You can then type the text of the subsection title into the subsection-break box.

   When you use this method of drawing section and subsection breaks, they will be placed on the screen at the top of the currently-visible area. You can move them around by dragging them (see section B.6.1).

2. The other way to draw section and subsection breaks is to click inside the editing area with the right-most mouse button and hold the button down. A menu will appear, from which you can choose "`Section Break`" or "`Subsection Break`".

   Choose "`Subsection Break`". You can then type the text of the subsection title into the subsection-break box.

   When you use this method of drawing section and subsection breaks, they will appear on the screen at the position where the mouse was when you clicked with the right mouse button to create the section or subsection break. You can move them by dragging them (see section B.6.1).

B.7 Updating the Selection Criteria

Did you notice a problem with the conditions you used in the new variation set? The value "senior" is not one of the choices for "age" in the Selection Criteria area.

So ... edit the Selection Criteria, and add the value "senior" to the list of values of "age". You do this by clicking with the mouse at the point where you want to enter more text, then typing the text you'd like.

B.8 Indicating Rhetorical Relations

Now, let's learn how to indicate rhetorical relations between variation sets. You will mark a rhetorical relation between the new variation set you've drawn and the first variation set in the subsection (the one that starts "*Readers who receive ...*").

The steps for drawing a rhetorical relation are the following:

1. Click on the coloured frame of the new variation set that you created, and then type the number "1" to indicate that this is the first variation set in

the rhetorical relation.

2. Click in the coloured frame of the first variation set and then type the number "2" to indicate that this is the second variation set in the rhetorical relation.

3. A black arrow will be drawn between the two boxes, from the new box to the first one, with the arrowhead at the first box.

4. A new window will pop up, containing a list of the possible rhetorical relations that you can choose from. Double-click on one of the relations, and the window will disappear. The arrow between the variation-set boxes will change colour to indicate the relation that you chose.

   For this example, it doesn't matter which relation you choose, since this is only a sample document.

If you can't see the arrow well because it's obscured by the rhetorical relation arrow that was already there, you can move the variation set box slightly to one side (see section B.6.1 on how to move a box). The arrow moves with the box.

B.9 Coreference

B.9.1 Indicating Coreference

Now let's indicate some words and phrases that are coreferential. In the first subsection of the document, you see that "*document*", "*tailored documents*", and "*tailored product documentation*" are all the same colour, which indicates that they are all coreferential.

You will add the word "*information*" in the variation that starts "*Computer companies can use …*" to this group of coreferences.

To indicate that "*information*" is part of the group, you perform the following steps:

1. Highlight the word "*information*" by dragging across it with the mouse.

2. Click inside the highlighted word with the right mouse button.

3. A new window will pop up. At the top of this window you can see that "*information*" is the current selection. The rest of the window is divided into two lists: a list of the groups of coreference that are already defined, and a list of the possible types of coreference.

   Only one group of coreferences has been defined so far: a group of references to "*document*".

4. Click on "document" in the first list to indicate that you are adding the word "information" to a group of references to "document". Click on "specific" in the list of types of coreference to indicate that "information" is a "specific" reference to "document".

5. Click on "Okay" to store the coreference. The word "*information*" will change colour to match the other words in this group of coreferences.

6. Click on "Close" to close the coreference window.

B.9.2 Creating a New Group of Coreferences

Let's move down to the second subsection "*How to Create Customized Documents*". Both sentences in the variation set start with "*The HealthDoc project*". You will indicate that these two sentences refer to the *same* HealthDoc project by marking the two phrases "*The HealthDoc project*" as being coreferential.

To do this, you perform the following steps:

1. Highlight "*The HealthDoc project*" in the first variation and click with the right mouse button.

2. When the coreference window pops up, choose "`Not Listed`" in both lists, to indicate that the current selection ("*The HealthDoc project*") belongs to a group of references that is not yet listed in the coreference window.

3. Click on "Okay". The phrase "*The HealthDoc project*" will be added to the list of groups of coreferences and it will change colour in the text.

4. Now, highlight the phrase "*The HealthDoc project*" in the *other* variation and click with the right mouse button.

5. Choose "`The HealthDoc project`" in the first list, and "`identical`" in the second list.

6. Click on "Okay", and this second reference to the HealthDoc project will change colour to match the first reference in the first variation.

7. Click on "Close" to close the coreference window.

B.10 Saving the Document

To save the changes you've made to the demo document, go to the `File` menu and choose `Save As`.

Choosing `Save As` will pop up a file selection window, where you can enter the name you would like the document to have. Do not use "demo-document.tsl" or you will overwrite the demonstration document. You can also use the file selection window to browse or traverse the file system, so that you can save the document in another directory.

Choosing `Save` performs a "quick" save. It does not pop up the file selection window: it saves the document using the existing filename. If this is a new document (so no filename has been given yet), it saves the document as "`new_document.tsl`". Please don't use `Save` for this document, or you will overwrite the demonstration document.

NOTE: BEFORE you save a document, be sure that you have specified section and subsection breaks (section B.6.3). If you don't you will get errors when you try to save, and the document will not be saved. In the demonstration document the sections and subsections are already defined, but you will need to be careful if you create your own document from scratch.

Note: Ensure that all sentences in the variation-set boxes have proper punctuation. If you leave a sentence that doesn't have punctuation at its end, the authoring tool will hang, and you will lose your document.

B.11 Quitting/Exiting the Authoring Tool

When you are finished with the authoring tool, go to the `File` menu and choose `Quit`. If there has been an error while the authoring tool was running and it is "hung", then `Ctrl-C` will kill the authoring tool.

Bibliography

Altavista Technology, Inc. (1996). Media Wrangler. http://www.altavista.com/corp/index.htm.

Banks, S. (1997). Automated generation of sentence plan language using stylistic parses (working title). Master's thesis, University of Waterloo, Waterloo, Ontario, Canada. Forthcoming.

Bateman, J. A. (1995, May). KPML: The KOMET-Penman multilingual linguistic resource development environment. In *Proceedings of the 5th European Workshop in Natural Language Generation*, Leiden, pp. 219–222.

Binsted, K., A. Cawsey, and R. Jones (1995). Generating personalised patient information using the medical record. In *Proceedings of the Conference on Artificial Intelligence in Medicine, Europe.* http://www.dcs.gla.ac.uk/~www/Piglet.html.

Buchanan, B. G., J. Moore, D. Forsythe, G. Banks, and S. Ohlsson (1992, November). Involving patients in health care: Explanation in the clinical setting. In M. E. Frisse (Ed.), *Symposium on Computer Applications in Medical Care*, pp. 510–514. American Medical Informatics Association: McGraw-Hill, Inc.

Campbell, M. K., B. M. DeVellis, V. J. Strecher, A. S. Ammerman, R. F. DeVellis, and R. S. Sandler (1994, May). Improving dietary behaviour: The effectiveness of tailored messages in primary care settings. *American Journal of Public Health 84*(5), 783–787.

CapSoft Development (1996). HotDocs and CAPS Author. http://www.capsoft.com.

Carenini, G., V. O. Mittal, and J. D. Moore (1994, November). Generating patient-specific interactive natural language explanations. In *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care*, Washington, D.C., pp. 5–9. American Medical Informatics Association.

Carenini, G. and J. Moore (1993, January). Generating explanations in context. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, Orlando, Florida, pp. 175–182.

Cawsey, A., K. Binsted, and R. Jones (1995, May). Personalised explanations for patient education. In K. de Smedt, C. Mellish, and H. Novak (Eds.), *Proceedings of the Fifth European Workshop on Natural Language Generation*, Rijks Universiteit Leiden, pp. 59–74.

Dahl, D. A. (1992, September). *PUNDIT - Natural Language Interfaces.* Logic Programming in Action. Heidelberg, Germany: Springer-Verlag.

Database Publishing Software (1996). Expert Publisher. http://www.databasepub.com.

DiMarco, C. and S. Banks (1997). Using subsumption classification on a stylistic hierarchy in the multi-stage conversion of natural language text to sentence plans. In preparation.

DiMarco, C. and M. E. Foster (1997). The automated generation of Web documents that are tailored to the individual reader. In *Proceedings of the 1997 AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, Stanford University, pp. 44–53.

DiMarco, C. and G. Hirst (1993a, September). A computational theory of goal-directed style in syntax. *Computational Linguistics 19*(3), 451–499.

DiMarco, C. and G. Hirst (1993b, September). Usage notes as the basis for a representation of near-synonymy for lexical choice. In *Proceedings of the*

Ninth Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary and Text Research, Oxford, pp. 33–43.

DiMarco, C., G. Hirst, and M. Stede (1993, March). The semantic and stylistic differentiation of synonyms and near-synonyms. In *Proceedings of the AAAI Spring Symposium on Building Lexicons for Machine Translation*, Stanford, pp. 114–121.

DiMarco, C., G. Hirst, L. Wanner, and J. Wilkinson (1995, August). Healthdoc: Customizing patient information and health education by medical condition and personal characteristics. In *First International Workshop on Artificial Intelligence in Patient Education*, Glasgow.

Document Science Corporation (1996). Document Science. http://www.docscience.com.

Eastgate Systems, Inc. (1996). StorySpace, A Hypertext Tool for Writers and Readers. http://www.eastgate.com/Storyspace.html.

Goffinet, L. (1996). Luc Goffinet's home page. http://www.info.fundp.ac.be/~lgo.

Harrow Software (1996). Formula Graphics. http://www.harrow.com.au/formula/Welcome.htm.

Hirst, G. (1981). *Anaphora in Natural Language Understanding: A Survey*, Volume 119 of *Lecture Notes in Computer Science*. Springer-Verlag.

Hirst, G. (1995, March). Near-synonymy and the structure of lexical knowledge. In *Working notes, AAAI Symposium on Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*, Stanford University, pp. 51–56.

Hirst, G. and C. DiMarco (1996, November). Automatic customization of health-education brochures for individual patients. In *Proceedings of the Conference on Information Technology in Community Health*, Victoria.

Ibrahim, B. (1991, May). Alice in Authorland! http://cuisung.unige.ch/CBL.papers/nato.report.html.

Information Sciences Institute, University of Southern California (1996). Intelligent Documentation (I-Doc). http://www.isi.edu/isd/i-doc.html.

Johnson, W. L. (1994). Dynamic (re)generation of software documentation. In *Proceedings of the Fourth Systems Reengineering Technology Workshop, APL Research Center Report RMI-94-003*, pp. 57–66. Johns Hopkins University Applied Physics Laboratory.

Lam, I. K. (1997). Tix programming guide. http://www.xpi.com/tix/.

Levy, D. M. (1988, December). Topics in document research. In *Proceedings of the ACM Conference on Document Processing Systems*, Santa Fe, New Mexico, pp. 187–193.

Lin, J.-K. (1992, November). MediaMosaic - a multimedia editing environment. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'92)*, Monterey, California, pp. 135–141.

Mann, W. C. and S. A. Thompson (1986, October). Rhetorical structure theory: Description and construction of text structures. Technical Report ISI/RS-86-174, Information Sciences Institute, University of Southern California.

MetaCard Corporation (1996). MetaCard. http://www.metacard.com/.

MicroMass Communications, Inc. (1996). IntelliWeb. http://www.micromass.com/.

Morris, J. and G. Hirst (1991, March). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics 17*(1), 21–48.

Osman, L. M., M. I. Abdalla, J. A. G. Beattie, S. J. Ross, I. T. Russell, J. A. Friend, J. S. Legge, and D. J. Graham (1994, February). Reducing hospital admission through computer supported education for asthma patients.

British Medical Journal 308, 568–571.

Ousterhout, J. K. (1994). *Tcl and the Tk Toolkit.* Addison-Wesley Professional Computing Series. Addison-Wesley Publishing Company.

Penman Natural Language Generation Group (1989). *The Penman User Guide.* Information Sciences Institute, University of Southern California.

Rösner, D. and M. Stede (1994, August). Generating multilingual documents from a knowledge base: The TECHDOC project. In *Proceedings of the International Conference on Computational Linguistics, 1994 (COLING-94), Volume I*, Kyoto, Japan, pp. 339–??

Schuler, W. and J. B. Smith (1990, November). Author's Argumentation Assistant (AAA): A hypertext-based authoring tool for argumentative texts. In *Hypertext: Concepts, Systems and Applications, Proceedings of the First European Conference on Hypertext*, INRIA, France, pp. 137–151.

Siglar, J. (1996). Multimedia authoring systems frequently asked questions (FAQ), Version 1.4.3. Web page. http://www.tiac.net/users/jasiglar/MMASFAQ.HTML.

Skinner, C. S., V. J. Strecher, and H. Hospers (1994, January). Physicians' recommendations for mammography: Do tailored messages make a difference? *Americal Journal of Public Health 84*(1), 43–49.

Strecher, V. J., M. Kreuter, D.-J. D. Boer, S. Kobrin, H. J. Hospers, and C. S. Skinner (1994, September). The effects of computer-tailored smoking cessation messages in family practice settings. *The Journal of Family Practice 39*(3), 262–270.

Texas A & M University (1996). Walden's Paths. http://www.csdl.tamu.edu/walden/.

University of Massachusetts (1996). Eon. http://www.cs.umass.edu/~tmurray/eon_www/eon.html.

VivaTEXTE (1996). VivaTEXTE. http://www.vivatexte.com/.

Welch, B. B. (1995). *Practical Programming in Tcl and Tk.* Prentice-Hall, Inc.