An Intensional Tool Applied in French Language Educational Software

By

Honglian Li
B. Sc, Nankai University, 1996

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
in the Department of Computer Science

Supervisor: Dr. W. W. Wadge

ABSTRACT

In this thesis the author presents a Web-based French language educational software system, French e-Flash Card (FFC), programmed using Intensional Markup Language (IML). The FFC site is, in some ways, a typical logic-based web application, which consists of a knowledge base of French grammar rules, a logic engine and a web based user interface. The FFC site is both interactive and dynamic, and incorporates Artificial Intelligence for French grammar. These special features, which distinguish FFC from most conventional language educational applications, are made possible by the intensional programming tool, IML, which allows rule-based markup in an intensional logic. The IML source files define whole families of pages indexed by parameters which specify, for instance, the particular subject matter, the degree of difficulty, and even the vocabulary choices and agreement constraints used to generate sample French sentences. This thesis describes the French e-Flash Card site, provides an overview of IML and defines a design approach for intensional programming which should enable web authors to find practical and systematic ways to program using intensional tools.

ACKNOWLEDGMENTS

I would like to thank Dr. W. W. Wadge, whose advice, brilliant creativity and enthusiasm have guided this thesis from its conception to completion. I also thank him for constant financial support.

I am grateful to G. Brown, Paul Swoboda, Qin Zhu, Xing Jin, R. Agarwal, and B. Moa for their contributions to Intensional Programming. Their excellent work forms the foundation for my research.

I would also like to thank Christine Wadge for her skillful wisdom of French education and her passionate influence which inspired me to learn French and, therefore, create this French language software.

I thank Ruth Dyck for proofreading my thesis and helping me with my use of English.

Finally, I am indebted to my family and friends for their unlimited love and encouragement.

Dedication

On ne voit bien qu'avec le coeur. L'essentiel est invisible pour les yeux.

Pour mon Petit Prince.

CONTENTS

ABSTRACT ---- ii

Chapter 1 Introduction ---- 1
1.1 Overview of this thesis ---- 1
1.2 Overview of the problem, the motivation and the approach ---- 2
1.3 Overview of Chapters ---- 6

Chapter 2 Background ---- 7
2.1 Intensional Logic ---- 7
2.2 Intensional Programming and Web Authoring ---- 8
2. 3 IHTML ---- 9
2. 4 ISE ---- 12

Chapter 3 French e-Flash Card design approach ---- 14
3. 1 Overview ---- 14
3. 2 FFC Approach ---- 15
3. 3 FFC design concepts ---- 19
3.3. 1 Malleable layout ---- 19
3. 3. 2 Concise contents ---- 20
3. 3. 3 Interactive interface ---- 20
3. 3. 4 Numerous examples ---- 21
3. 4 Examples ---- 22
3. 4. 1 Table ---- 22
3. 4. 2 Vocabulary table ---- 25
3. 4. 3 Menu ---- 27
3. 5. 4 Search Engine ---- 29
3. 5 Conclusion ---- 30

Chapter 4 FFC implementation tool: Intensional Markup Language ---- 31
4. 1 Introduction ---- 31
4. 1. 2 Chapter overview ---- 32
4. 2 The backbone of IML ---- 32
4. 2. 1 Evolutional process ---- 33
4. 2. 2 Publishing mode ---- 33
4. 3 Functions ---- 34
4. 3. 1 Macros for intensional links ---- 34
4. 3. 2 Macros for version control ---- 37
4. 3. 3 Initialize version dimension ---- 42
4. 3. 4 Macros for layout definition ---- 42
4. 3. 5 Macros for debugging ---- 48
4. 3. 6 Macros for menus ---- 48
4. 3. 7 Macros for extensibility ---- 50
4. 4 Define new macros ---- 50
4. 4. 1 Process of defining new macros ---- 51
4. 4. 1. 1 Start from generic macros ---- 51
4. 4. 1. 2 Start from primitive ISE ---- 53

4. 4. 2 More examples ---- 54

Chapter 5 French e-Flash Card implementation ---- 55

5. 1 Introduction ---- 55
5. 2 Application Architecture ---- 55
5. 3 Components Structure ---- 57
5. 4 Layout ---- 62
5. 5 Features ---- 63
5. 5. 1 Alternative menus ---- 63
5. 5. 2 Search Engine ---- 66

Chapter 6 Future Work and Conclusion ---- 68

6.1 Add Multimedia materials in FFC ---- 68
6.2 Include more grammatical units ---- 68
6.3 Enhancement to IML ---- 68
6.4 Conclusion ---- 69

Bibliography ---- 71

Appendix A: IML macro reference ---- 73

Appendix B: New Defined IML macro reference ---- 77

Appendix C: New Defined IML macro definition ---- 78

Appendix D: Comparison of IML and related web authoring languages ---- 82

1. IML vs. IHTML 2 ---- 82
2. IML vs. ISE ---- 82
3. IML vs. IXML ---- 83
4. IML vs. DHTML ---- 86

LIST OF TABLES

TABLE 3.1 AN ENGLISH SENTENCE STRUCTURE EXAMPLE 16
TABLE 3.2 SIX-DIMENSION ENGLISH SENTENCES TABLE 17
TABLE 3.3 SPECIFIED OPTIONS IN THE SIX-DIMENSION TABLE 17

LIST OF FIGURES

FIGURE 3.1 EXAMPLES OF THE GENDER OF OBJECTS 21
FIGURE 3.2 EXAMPLE TABLE OF THE ARTICLE SECTION (1) 23
FIGURE 3.3 EXAMPLE TABLE OF THE ARTICLE SECTION (2) 24
FIGURE 3.4 VOCABULARY TABLE OF POSITION OF ADJECTIVES. 26
FIGURE 3.5 VOCABULARY TABLE OF THE POSITION OF ADJECTIVES (2) 27
FIGURE 3.6 EXAMPLE OF MASCULINE NOUNS 28
FIGURE 3.7 SCREENSHOT OF POSSESSIVE ADJECTIVES 29
FIGURE 3.8 EXAMPLE OF SEARCH FUNCTION IN FFC 30
FIGURE 4.1 IML PUBLISHING MODE 34
FIGURE 4.2 SCREEN SHOT OF THE "COMMON AND PROPER NOUN" SECTION 36
FIGURE 4.3 SCREEN SHOT OF NATIONALITY MENU IN THE GENDER OF NOUN 38
FIGURE 4.4 SCREEN SHOT OF POSSESSIVE ADJECTIVE EXAMPLE (1) 41
FIGURE 4.5 SCREEN SHOT OF POSSESSIVE ADJECTIVE EXAMPLE (2) 41
FIGURE 4.6 SCREEN SHOT OF STRETCHTEXT EXAMPLE (1) 44
FIGURE 4.7 SCREEN SHOT OF STRETCHTEXT EXAMPLE (2) 45
FIGURE 4.8 EXAMPLE OF DROPTEXT (1) 46
FIGURE 4.9 EXAMPLE OF DROPTEXT (2) 46
FIGURE 4.11 EXAMPLE OF MENU IN FFC 49
FIGURE 4.12 POSITIVE AND NEGATIVE ANSWER BUTTON 52
FIGURE 5.4 FFC INTERFACE 62
FIGURE 5.5 EXAMPLE OF ALTERNATIVE MENUS 1 64
FIGURE 5.6 EXAMPLE OF ALTERNATIVE MENUS 2 64
FIGURE 5.7 THE USER SELECTION SCENARIO 1 72
FIGURE 5.8 THE USER SELECTION SCENARIO 1 72

Chapter 1 Introduction

1.1 Overview of this thesis

This thesis presents details of three interrelated topics: the French educational grammar software known as French e-Flash Card developed by the author, the design approach used for constructing this software and the programming tool behind it – Intensional Markup Language.

The software French e-Flash Card (FFC for short) was designed and implemented by the author using an intensional programming approach with IML as the programming tool. The FFC application, as a link, weaves the other two subjects into a complete work in this thesis. The parametric approach of FFC is inspired by Wadge's French Sentence Maker program [1] which is programmed directly with ISE, a low level Perl-like programming language with intensional features. FFC is a more ambitious application developed using IML, a high-level markup language. IML is a rule based extension of HTML, which allows large chunks of ISE code to be generated by simple macro calls, making it relatively easy to produce the source code for the grammar knowledge base and the logic engine.

FFC is a typical logic-based web application; which is composed of a French grammar knowledge base, a logic engine and a presentation layer to display the answers. FFC's ability to produce sentences dynamically in response to the user's demand transcends most conventional language educational software, which offer only static and predefined examples. Since the FFC rules embody knowledge of French grammar, it is capable of generating student questions and example utterances that have not been explicitly entered in a database. Furthermore, the intensional logic behind IML supports

an inheritance relation between contexts. This makes it easy to express the rules in terms of general principles and exceptions (and exceptions to exceptions, and so on). In subsequent chapters the author provides a detailed description of the tool's usage and compares it with other authoring tools.

1.2 Overview of the problem, the motivation and the approach

As a French beginner, the author found that current French class-based education focuses mainly on grammar knowledge built up from small, simple facts; for example, the form of a particular adjective takes in the feminine plural, or the conjugated form of a particular verb in a particular person, number and tense. Students are taught and tested on their knowledge of these kinds of atomic facts of French grammar. However, even simple expressions use surprisingly large amounts of these atomic facts. Therefore, French beginners get frustrated when attempting to use what they have learnt in class, and some are even unable to make well-formed sentences in simple conversation.

Most education experts are aware of this flaw in class-based language education and prefer a language-learning environment that improves the efficiency of acquisition. It is well known that imitation - as babies do when they learn their mother language - is the most efficient and natural means of learning a language. Therefore, the author decided to create a simulated language environment for adult students to learn French with the aid of computer software.

There are a number of French grammar websites or applications already available. But in reviewing them, the author found that many were still employing the class-based educational model. They usually move the contents of French grammar textbooks online

and include exercises that are also based on the questions in particular grammatical units. These websites are simply electronic versions of the textbooks, and have the same problem as their paper-based counterparts: they provide only a limited supply of examples of complete sentences, namely the ones explicitly written, one by one, by the authors. There may be hundreds of these sample sentences, but a student who spends even one week in France, immersed in French, will hear or read thousands and thousands of sentences. Electronic versions of textbooks often have AI which automates the production of atomic grammatical facts, but not complete utterances or even complete phrases. As a result, these conventional websites, like their paper-based counterparts, are not really practical for students who wish to practice their new language.

The author investigated how to provide a language environment with the aid of a computer. Researchers find that learning efficiency is improved if the learning unit extends beyond isolated grammatical units with vocabularies to complete sentences. This is because it is more efficient to remember a sentence as a single language unit than it is to remember the atomic grammar facts and vocabularies separately. Particularly in oral communication, if the storage unit consists of separate grammar and vocabularies, students have to search their vocabulary, apply the grammar rules and calculate the resulting sentence mechanically. This is not a practical way to use a foreign language in daily communication.

Therefore, the author decided to improve upon the existing educational software by adding a large quantity of 'virtual' or dynamically-generated sample sentences. The essential concept of French e-Flash Card is to provide as many sentences as possible in each grammatical unit, in order to present typical usage of each grammatical concept. For

example, in the possessive adjective section, the author built a set of examples to cover all possible uses of possessive adjectives: singular, plural, masculine and feminine.

In designing and implementing this sentence-making function, it was necessary to solve the following technical problems:

- How to ensure that users, with only limited knowledge of French grammar and vocabulary, can make both grammatically correct and semantically meaningful sentences.
- How to allow for customization of sentence examples.
- How to improve interaction between the user and the software
- How to logically arrange the contents in a web interface, in order to make the large quantity of sample sentences easily accessible.

To solve these problems, the author utilized a state-of-the-art approach to design the FFC software. Through FFC, the author provided the major elements of a sentence, such as subjects, predicates and objects. The user can then choose appropriate vocabulary items which they want to use in the sentence and FFC will select and apply all relevant grammar rules and generate the sentence for the user. The user can choose different vocabulary items or move to different grammatical sections to repeat the sentence-making process, and can therefore, learn grammar by making sentences.

In order to improve customizability and interactivity, FFC uses the parametric method to provide the user with a customized version of the web pages. Sentence elements are listed in a table, or in a menu, to allow the user to customize his or her sentences with appropriate vocabulary items. Each option in a menu, or each item in a table, is an intensional link which a new valaue for a particular parameter. Once selected

by a mouse click the mouse, the parameter's current value will be changed. Despite the fact that these pages are not predefined in the server-side database, the server side will calculate the best-fit version according to current version expressions. This intelligent function comes from the Intensional Markup Language, which is the tool behind FFC implementation.

Each example sentence formed by combining the selected elements is a version of the sentences determined the settings of key parameters. These parameters are a subset of those which determine the version of the web page being displayed. When the user changes an element selection FFC generates a new version of the web page, using the new parameter settings, to present the new example to the user..

The reason that FFC can exceed the limitations of conventional web applications in providing customization and multiversion solutions is the tool, IML, used in its implementation. IML is a simple markup language that extends HTML. It is constructed on top of ISE, a Perl-like CGI language with run-time parameterization. The IML source is translated (once) into a corresponding ISE program which, when run with specific parameter settings, produces the HTML which renders the desired version of the requested page. [2]

Therefore, IML utilizes the parametric approach. It makes it possible for many pages of a website to share generic elements by sharing parameters. Furthermore, it makes customization of web pages possible by altering parameters.

FFC is the combined result of both the approach and the tool. It is an innovative test of the intensional programming approach and tool. Further details on this application are documented in the following chapters, which are précised in Section 1.3.

1.3 Overview of Chapters

Chapter 2 provides background information on the topics which underpin the work in this thesis including: intensional logic, intensional programming, versioning systems and intensional web authoring tools e. g. IHTML and ISE.

Chapter 3 demonstrates FFC's design approach and provides a comparison with conventional French grammar educational software.

Chapter 4 describes and evaluates IML, the tool used in the implementation of FFC, from the perspective of its function, uses, advantages and shortcomings compared to other web-authoring tools such as IHTML, ISE, IXML and DHTML.

Chapter 5 describes the implementation of the author's educational language software French e-Flash Card. Specific detail is also provided on its architecture, version and navigation control, and data structure in this chapter.

Future work and conclusions are presented in Chapter 6.

Appendix A is a complete reference manual for IML; Appendix B provides a list of newly defined macros in FFC; and Appendix C contains the definitions of the newly defined macros.

Chapter 2 Background

2.1 Intensional Logic

Because Intensional logic forms the foundation of intensional programming we provide an overview of this subject within the context of this thesis.

Intensional logic is the logic of assertions and expressions which vary over different collections of contexts or possible worlds. It is very common in natural language. For instance, we can't judge the statement: "The average temperature is over 30 Celsius" as true or false unless we know its context, i.e. when and where the temperature was recorded. "Entities which vary according to context are called *intensions*, and each particular value determined by a particular context is called an *extension*." [3]

Intensional logic employs a framework of semantics called possible worlds, attributed to Honderich [4], Leibniz by Chellas [5] and others. Many variations of intensional logic have been described using possible worlds semantics, such as temporal logic, in which the possible worlds are time-points and the meaning of expressions therefore changes with time. For the FFC site, the set of possible worlds is a set of possible versions of pages or parts of a page. The accessibility operator is "$\subseteq$", with the meaning "is refined to". $A \subseteq B$ means that the version B is a refinement of version A. The refinement operator defines a directed graph of possible worlds, which will be used to find the closest match to a version.

2.2 Intensional Programming and Web Authoring

Intensional programming is programming using intensional logic. One obvious characteristic of intensional programming is using intensional operators to manipulate different versions of entities of almost any kind.

The first intensional programming language is Lucid, invented by Wadge and E. A. Ashcroft in 1974. [3] Its collection of possible worlds is a programmer-defined multidimensional space, where each dimension is defined as non-negative integer. It has several operators, allowing access to the value at the next index in a dimension, the previous index, or all indices meeting some criteria. Although there is no direct connection between Lucid and FFC, they share the same fundamental system of demand-driven multidimensional processing.

The World Wide Web is a very important application of intensional programming as Bill Wadge states in [28]: "The World-Wide Web is the first large-scale experiment in Intensional Programming and Education. ... It should be clear then, that the Web is exactly an intension. The Web is nothing more or less than an indexed family of pages, the indices being the URLs." The web contains many examples of families of pages which exist in different versions. For instance, Canadian Government websites generally have English and French versions which may share graphics or text, such as headers and footnotes, with other pages. The entire family of web pages can therefore be considered as two different versions of a single web page and these related pages can share source. A version control system, which allows code sharing between versions, is desirable (though often they are maintained manually).

The following sections give a brief review of IHTML and ISE as two typical intensional programming languages used in web applications.

2. 3 IHTML

As an extension of HTML, IHTML facilitates the authoring of multiversion sites that vary according to a number of parameters. Wadge described IHTML's major innovation that allows it to exceed HTML in [9]: "It allows the web authors to create source files for pages and parts of pages which, in general, are generic - valid for a whole family of versions of the component in question."

Wadge pointed out in [6] that: "Initially, the most important feature of IHTML was that it allowed multiple source files for the same page, each labeled (as above) with a version expression. When a request arrives for a particular version of a particular page, the IHTML server (an Apache server plug-in) uses the best-alternative rule described above to select the appropriate source file."

IHTML improved version control in the invention of version expressions which have global and uniform significance. In IHTML the version expression usually contains two parts: a version dimension identifier and a corresponding value, separated by colon. The expression values are not necessarily numbers. [8] For example, the expression `language:french` indicates that the language parameter of the document is French. Generally, IHTML also supports the sum of such expressions by specifying that each parameter identifier has its own respective value. For instance:

```
language: french + OS: 8 + processor: PPC
```

might specify the french language version for a PPC Macintosh running OS 8.

There has been a succession of refinements to the basic idea of combining versioned modules using a best-fit approach to software configuration. IHTML employed the basic notion of a set of versions of a file, and the use of refinement ordering of versions in a best-fit approach to select the appropriate version when constructing a complete system.

IHTML has four major components: the intensional link, server-side includes, intensional image and intensional case statement.

The most important feature of IHTML is the intensional link which allows authors to specify, with a single source tag, whole families of links between different versions of a page.

As in HTML, IHTML uses anchors to provide the link function. The simplest intensional link is identical to a conventional HTML. For example, if the IHTML source for `intro.html` contains the following tag:

```
<a href="intro.html">Introduction</a>
```

Tags will show corresponding versions of an English intro.html or a French version intro.html with the version expression `lg:en` or `lg:fr`. If English is the default language version (vanilla version) of `intro.html`, the version expression `lg:en` is usually omitted as showed in the above example.

```
<a href="intro.html<lg:fr>">Introduction</a>
```

In the above statement, the absolute version value was given by the expression `lg:fr`, which set the language of instruction value to French.

IHTML has server-side `include` function, which interpolates included files into current pages according to the customer's request. In the following example of a server-side include,

```
<!--#include file="extra.html" -->
```

if the current page is requested in its English version, the file included should be the English version of extra.html if it is available. Otherwise, a more generic version will be included instead according to the Best Fit Algorithm, or, if even the most generic/vanilla version cannot be found, an error message will be sent to the user.

The Intensional Image tag has similar parameters to the intensional hyperlink, for example:

```
<img src="picture.jpg" VERSION="Language:English">
```

Defines the use of the absolute version `Language:English` for `picture.jpg`.

The result will be the same in the following statement:

```
< img src="picture.jpg" VMOD="Language:English">
```

Which demonstrates the use of VMOD="`Language:English`" to change the current page version.

Finally, IHTML has a kind of intensional case statement which, when evaluated, selects the text for inclusion based on the particular version being produced. Here, for example, is the IHTML code which provides links between the French and English versions of a bilingual page:

```
<iselect>
  <case version="lg:fr">
    <a href=home.html vmod="lg:fr">Version
française</a>
```

```
    </icase>
    <case version="lg:en">
      <a href=home.html vmod="lg:en">English version</a>
    </icase>
  </iselect>
```

This same identical piece of code works for both the French and English versions. [8]

2. 4 ISE

IHTML worked well enough for small sites, but as a markup-only system it proved to be impractical for larger projects. [9] In 1998 Wadge and Paul Swoboda, designed and implemented ISE, Intensional Sequential Evaluator, a Perl-like scripting language. Swoboda, the implementer, describes the language as follows: "ISE is an attempt to create the first imperative scripting language with versioned identifiers. The basic notion in ISE is that everything that can be identified, including user-defined functions, can also be defined in multiple versions. When constructs such as variables and functions are accessed in expressions, intensional best fits are used to find the appropriate version of the entity referred to" [10]. "Briefly, this means all entities in the language (variables, arrays, functions etc) can be versioned; that programs execute in an implicit context which determines which versions of the relevant entities are used; and that there are also explicit mechanisms for accessing and modifying the context" [8]

Through ISE, Wadge found an alternative way to produce an HTML file - using the `print` statement. Using the conditional construct (`vswitch`), ISE can output certain versions of HTML files at runtime. A different value of the version parameter following the URL, determines a different versioned HTML file section being printed out. The print statements are used to output the HTML file sections and a conditional construct (`vswitch` statement) is used to decide which version of the HTML file section

will be output at runtime. Differences in value of the version parameter following the URL determines which version of a HTML file section is printed out.

Chapter 3 French e-Flash Card design approach

3. 1 Overview

French e-Flash Card (FFC) is an educational application designed to help French beginners learn grammar through a series of topical units including noun, adjective, adverb, verb and pronoun. It covers many important aspects of grammar including word order, conjugation of verbs, tenses, agreement of adjectives, comparison of adjectives and adverbs, direct and indirect objects, negation, the use of possessive adjectives, formation of adverbs, and the use of definite and indefinite articles. FFC is easy to operate with its user-friendly web-based interface and can be run from any web browser on any operating system.

The most important concept behind FFC is that it enables users to create a large number of example sentences and therefore learn French in an interactive and motivating environment. To achieve this, FFC provides several operations for users to create sentences. For example, it allows specific vocabularies and other sentence parameters to be selected from a set of tables or menus by simply clicking several hypertext links. Users may also look for the conjugated form of a verb using a search engine provided by FFC. In addition, sentences can be checked by moving the mouse over the questions. And finally, users are able to control the number, tense or other parameters of a sentence by using predefined buttons or menus. With these operations, even users with very limited knowledge of French grammar are able to create correct sentences and so learn French grammar by example.

FFC has a number of characteristics that distinguish it from conventional French grammar applications. Firstly, it is designed to be example-oriented: It creates a language

environment with plenty of examples. Secondly, it aims to encourage the user to grasp French through sentence construction, not just through a knowledge of grammatical theory and principles. This contrasts with conventional French applications which are usually presented in isolated grammatical units with limited example sets tailored to demonstrate specified topics.

Besides the above differences noted from a user's point of view, FFC differs from conventional French grammar applications from a technical perspective through its innovative scheme of web authoring. It is a multiversion web application that utilizes a parametric scheme to present different versions of a web page family. Moreover, it provides a mechanism allowing different members of a web page family to share common resources, e.g. layouts, data or contents. French e-Flash Card (FFC) is also, in some ways, a typical logic based language educational application, which consists of a knowledge base containing French grammar rules and a logic engine to analyze and repond to user's requests. It uses a default logic in defining and applying the rules of sentence construction. These logic-enabled features overlayed on top of a conventional web application are implemented with the intensional programming language tool IML.

The chapter is organized in the following order: Section 3.2 will describe FFC's design approach. Section 3.3 will introduce the basic design ideas behind FFC. And in section 3.4, some examples are provided to explain the design approach.

3. 2 FFC Approach

French e-Flash Card (FFC) avoids the problems of conventional French language educational web pages by using the intensional programming tool IML to specify

dynamic and multiversion Web pages. The basic idea behind FFC is that it enables computers to generate grammatically correct sentences for students and therefore aid language acquisition. The problem, however, is how to maintain control over generated sentences if students are allowed to input whatever they want. To solve this dilemma, the author provides relatively small and well-defined fragments of French in tables or menus. The user can specify fragments and other sentence parameters simply by clicking on them. FFC will then create a sentence according to the user's specified parameters. These French fragments are designed to be indexed using a simple multidimensional scheme.

In fact, French, like the other natural languages, is easy to divide into multidimensional fragments. Consider, for example, the following English sentence:

The boy likes the cheese.

We can see that there are three major parts in the sentence: the subject *the boy,* the predicate *likes* and the object *the cheese.* If we allow choices for the subject, verb and object, we can produce a table as follows:

Table 3.1 An English sentence structure example

The subject	The predicate	The object
Boy	Like	cheese
Girl	Adore	wine

Users can then choose different fragments to make sentences and from the table above, any combination of choices, one per column, will form a sentence. Therefore, in this instance the user can make 8 sentences. For example:

The girl likes cheese.

The boy adores wine.

If we wish to show the user English grammatical knowledge regarding tense, number and affirmation/ negation, we can add three more dimensions to the table.

Table 3.2 Six-dimension English sentences table

The subject	The verb	The object	Tense	Form	Number
Boy	Like	Cheese	Present	Affirmative	Singular
Girl	Adore	Wine	Past	Negative	Plural

For example, making the following choices:

Table 3.3 Specified options in the six-dimension table

The subject	The verb	The object	Tense	Form	Number
Boy	Like	Cheese	Present	Affirmative	Singular
Girl	**Adore**	**Wine**	**Past**	**Negative**	**Plural**

results in the sentence:

The boys didn't adore the wine.

Using the table above, users can make 64 different sentences which cover numerous aspects of English grammar, such as:

- The plural of *boy* is *boys.*
- The ordering of a negated sentence is *the subject-negative form of verb-object.*
- The negative of past tense form of verb *adore* is *didn't adore*.
- The definite article *the* is used for both singular and plural nouns.

We could make even more aggressive efforts to create additional dimensions to the table, allowing users to produce more sentences and cover more complex grammatical topics.However, the strategy of incorporating more and more grammar and vocabulary in

an increasingly complex sentence generator is impractical leading to the problems described in [1]:

More features means more dimensions and more choices and user interface (not to mention the implementation) could become bewilderingly complex. Also, more choices mean more possibilities for nonsensical combinations – such as "the house reads the car". These sentences may be grammatically correct but they are confusing to the learner and must be avoided.

Therefore, the author follows the suggestion proposed in [1], which is to work on a family of grammatical web pages. With each web page focusing on a specified grammatical topic and the inclusion of a modest sentence generator for each page, it is easy to control the complexity and correctness of sentences. Therefore, it is easy to control the learning progress.

Using this supposition the following five grammatical units are presented in FFC:

- Nouns
- Pronouns
- Verbs
- Adjectives
- Adverbs

Each unit is divided into several sub-sections necessary to fully comprehend the main topic. For example, the noun unit includes three sub-sections: gender, number and articles. FFC also supplies each unit with the means to enable users to create sentences to clarify the grammar. For example, in the position of adjective section, users can make

about 200 sentences to learn and gain confidence in putting adjectives around a noun. Besides examples, FFC still provides concise grammatical facts in each unit and gives explanations to more complex examples as would be expected from a conventional language education aid.

3. 3 FFC design concepts

The essential idea of French e-Flash Card is to provide concise grammar tuition along with plenty of examples in order to create a simulated language environment for users. To do this FFC employs simple layouts, concise contents and optional answers to help users test their recollection of grammatical knowledge in much the same way as old-fashioned paper-based flash cards. Since the application is analogous to this traditional teaching aid, its design style was inspired by the idea of flash cards. And as it is electronic, with a large quantity of user-selectable examples, it was named French e-Flash Card.

3.3. 1 Malleable layout

FFC uses a malleable layout that can be transformed by the individual user until it is tailored to their requirements. The transformations can vary among different levels of information, each of which provides different examples. This feature is very important for language learners. As Wadge points out in [7]:

"In other contexts, however, this same immutability causes serious problems. A case in point is the use of documents in classroom teaching (the English word *document,* which is descended from the same Greek root as *doctor*, originally meant a *lesson.*) Every teacher knows that elaborate support materials are often

more trouble than they are worth, because they make it difficult to adjust the lesson (in real time) to the strengths and weaknesses of each individual class."

3. 3. 2 Concise contents

The author tried to use concise words to explain the important grammatical issues in each section, after observing that superfluous words hindered students' concentration. Instead of tediously long descriptions of grammatical facts numerous examples are used to explain the rules of French grammar. This idea of teaching by example is the major design concept of French Flash Card.

3. 3. 3 Interactive interface

FFC presents French sentences and grammar explanation via an interactive interface that supports the customized selection of vocabulary and target grammar issues. In the following example (see Figure 3.1), which looks at the gender of objects in French, users can choose any subject, predicate and object from the popup menus on the right. FFC will then make sentences using the selected words so that all results are produced from the user's commands.

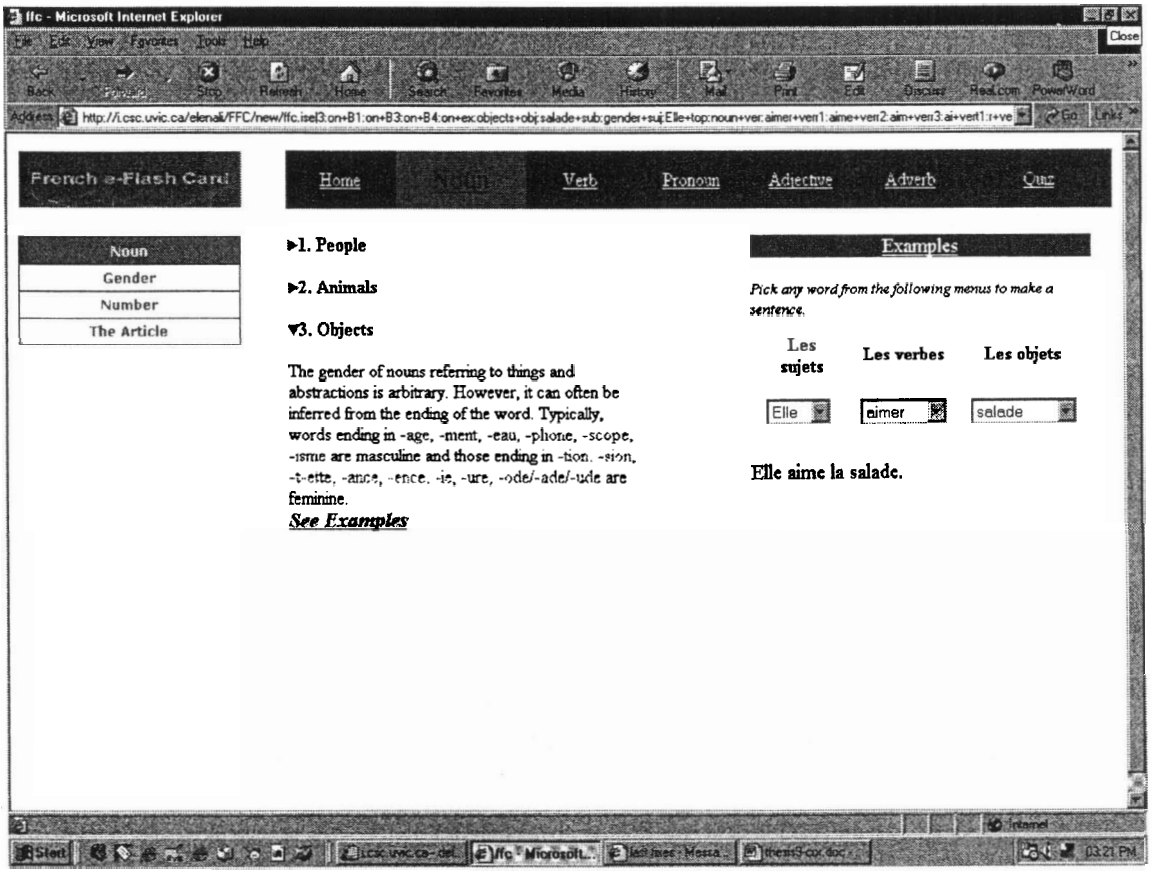

Figure 3.1 Examples of the gender of objects section

3. 3. 4 Numerous examples

The objective of FFC is to provide ample examples, which is also the distinguishing characteristic that separates it from other French language educational software. In the above example there are six choices in the subject menu: *je, tu, elle, nous, vous and ils*; four choices in the predicate menu: *aimer, manger, utiliser* and *regarder*; and 7 items in the object menu. This means that users are able to make approximately 148 different sentences.

However some of those sentences, while correct in grammar, are semantically meaningless, for example, *vous mangez la culture. (You eat the culture)*, and are automatically filtered out by FFC. But even after screening the above example creates

approximately one hundred sentences from which users can learn about the grammar of an object's gender.

3. 4 Examples

3. 4. 1 Table

The following example is extracted from the article section of FFC which covers the grammar of definite, indefinite and the partitive articles in French. The author has used a three-dimensional table to list fragments including subjects and verbs, articles and objects, for students to use in sentence generation

If, as shown in Figure 3.2, the user selects the subject and verb *je choisis* and the object *fromage*, FFC will automatically choose the masculine indefinite article *un*, the indefinite article cell is highlighted in red and so the user obtains the grammatically-correct sentence:

Je choisis un fromage.

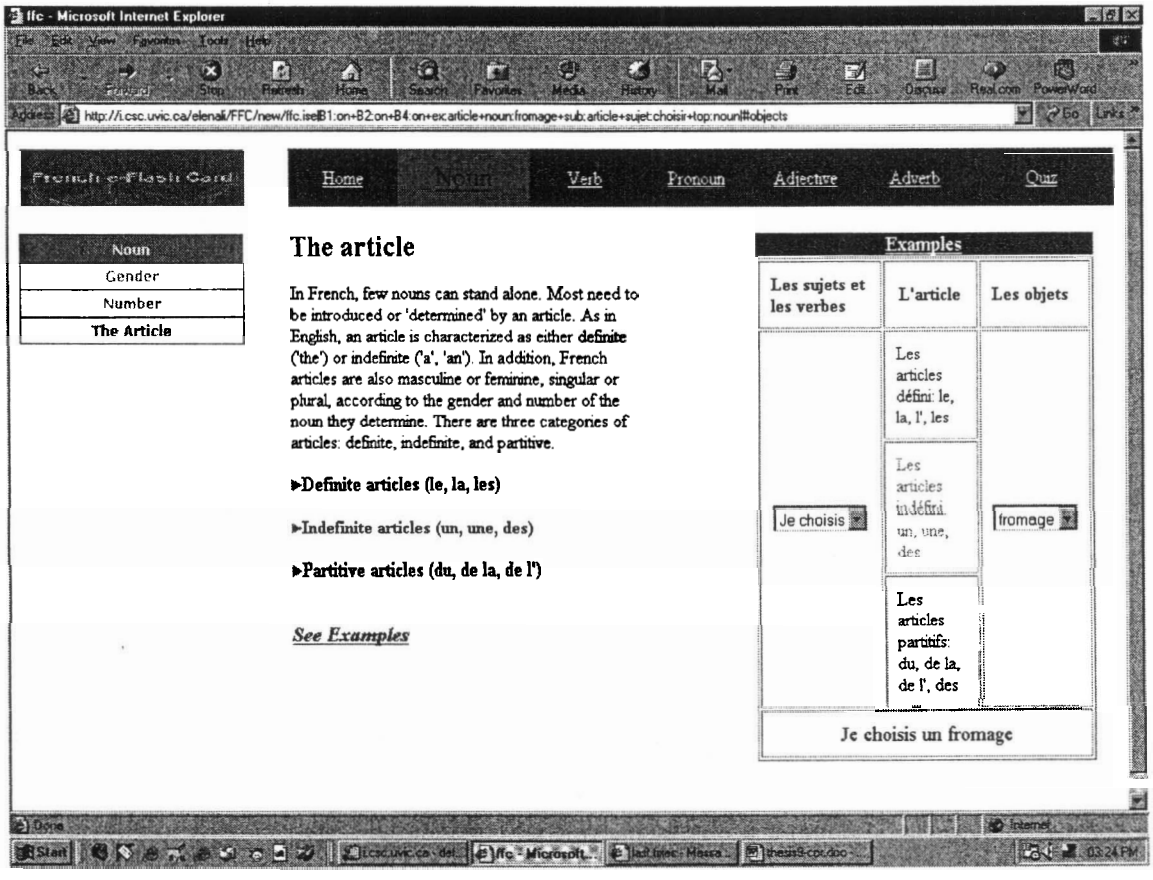

Figure 3.2 Example table of the article section (1)

If the student selects a different option from the fragment menus,for example, the subject and verb *j'achéte*, and the object *Fromage*. FFC then calculates the best-fit article to be the masculine partitive article *du*. (See Figure 3. 3). In the table, the partitive article cell's color changes to red highlighting the correct choice and ensuring that the user generates the grammatically-correct sentence:

J'achéte du fromage.

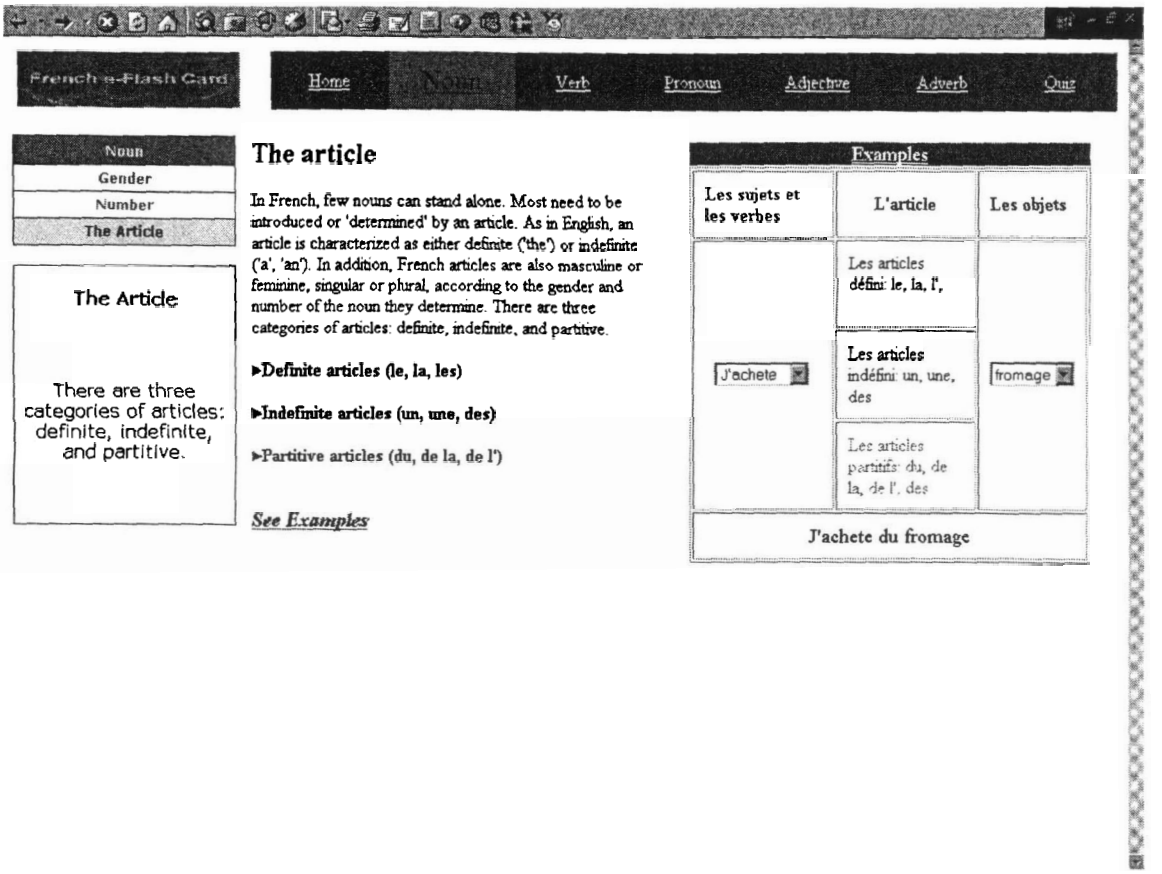

Figure 3.3 Example table of the article section (2)

In the example table, there are a total of three options in the subjects and verbs menu and six options in the objects menu. Each combination, one per menu, makes a sentence. The user can produce 18 sentences that present all possible uses of the definite, indefinite and partitive articles and cover the following grammar points:

- Definite and indefinite article in singular, plural, masculine and feminine modes.
- The definite article is always used with the verbs that express a strong feeling such as *aimer, préférer, détester*.
- Contractions with an article.
- The general use of the partitive article.

- The formation of the partitive article.

3. 4. 2 Vocabulary table

FFC uses a vocabulary table to list words which users can select to make a sentence. The following example explains the position of adjectives. The vocabulary table on the right side of the screen provides the user with 16 adjectives and 14 nouns. Once the user selects a noun and an adjective, FFC will put the noun and the adjective in the right position and add the subject and the verb.

For example, when the user chooses the adjective *beau* and the noun *garçon*, FFC determines that the adjective *beau* should be put before the masculine noun *garçon* (See Figure 3.4). In addition, it will insert *c'est* and the indefinite article *un* (for a masculine noun) to generate the following sentence:

C'est un beau garçon.

Taking another example from the vocabulary table, if the user chooses one of the adjectives *pauvre, vieux, ancient, grand* and any noun from the list, FFC will make two sentences because these adjectives could be put before or after the noun to give two different meanings. See Figure 3. 5.

C'est une pauvre femme. (This is a poor woman.)
C'est une femme pauvre. (This is a needy woman.)

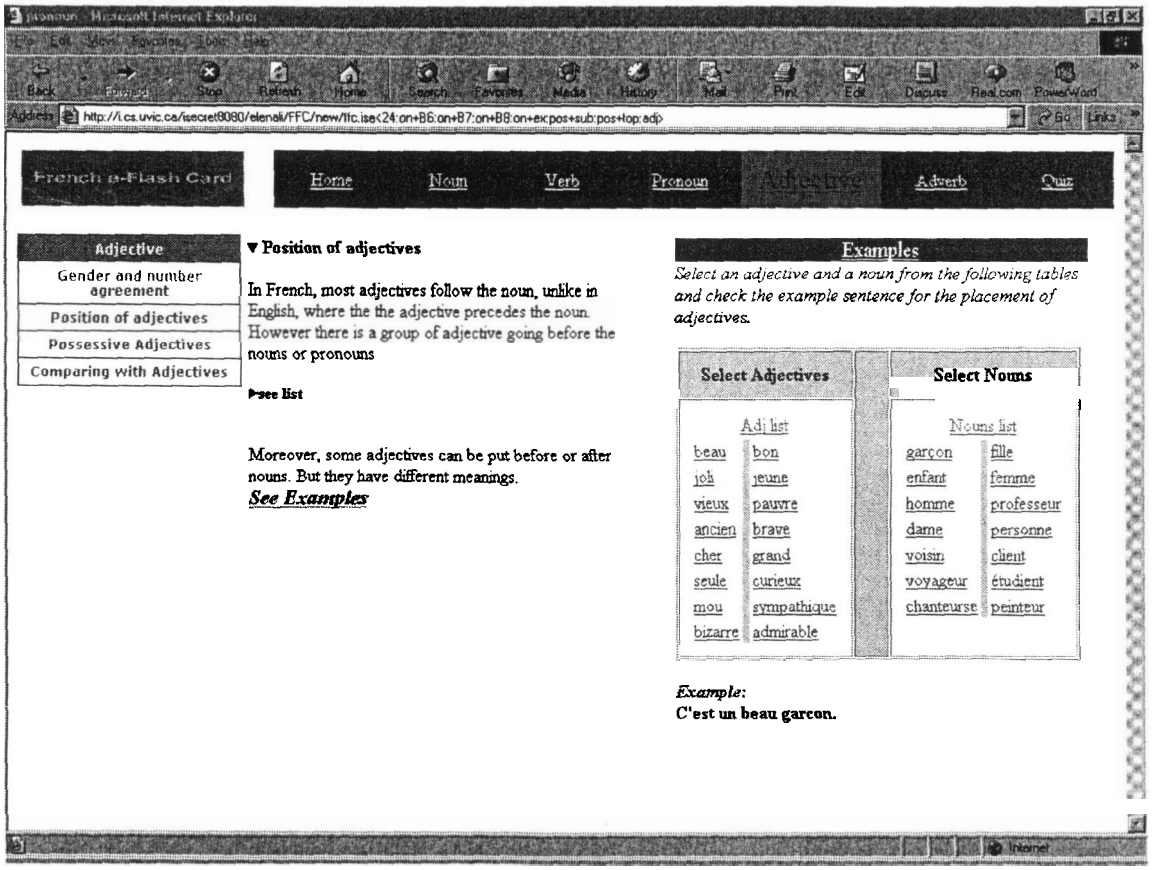

Figure 3.4 Vocabulary table of Position of Adjectives.

Users can develop more than 200 sentences to illustrate the the following grammatical rules:

- The majority of adjectives follow the noun.

  The position of some adjectives can change a sentence's meaning.

- The agreement of adjectives with noun in gender and number.

  The agreement of articles with the noun in gender and number.

- The formation of feminine adjectives.

- The identification and the presentation with *c'est.*

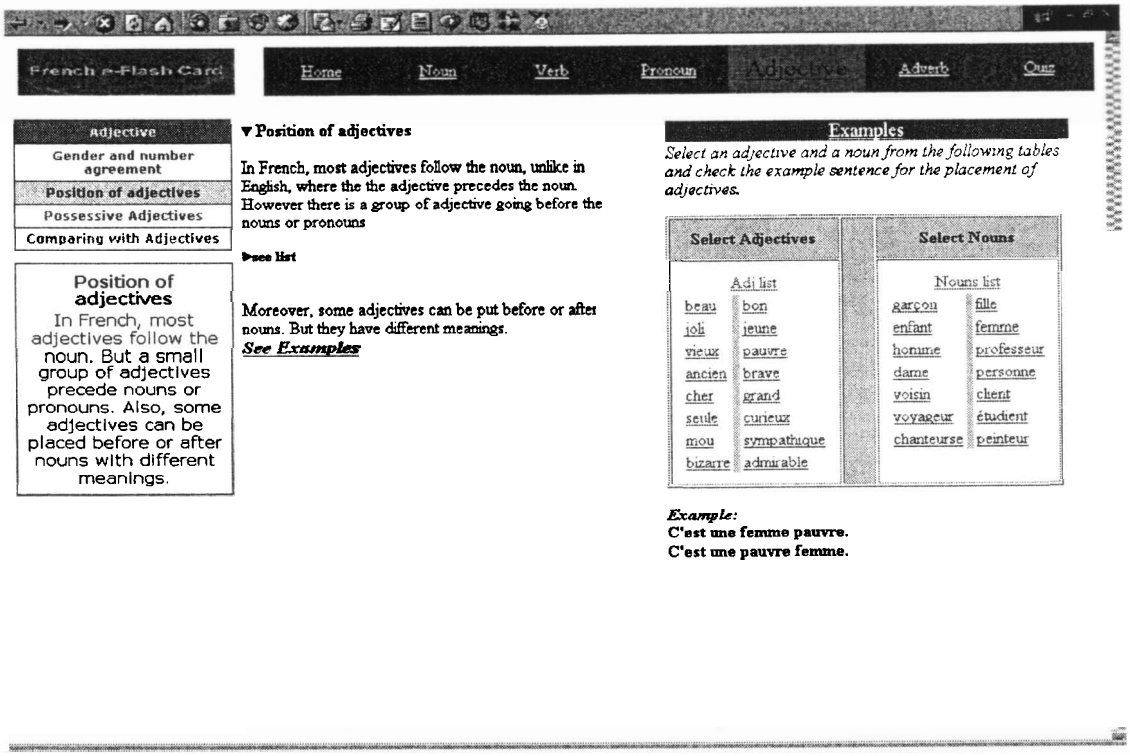

Figure 3.5 Vocabulary table for the Position of Adjectives (2)

3. 4. 3 Menu

FFC makes use of menus to enable users to choose vocabularies. For example, in the example kit for the gender of noun section, users can select professions and nationalities from menus, and the sentence will change according to their choices. If the user clicks on the girl's image, then all options in the two menus will change into the feminine format and the student will view the following introduction:

Margot: (the girl's name)

Je suis française.

J'espère devenir chanteuse.

But if the man's image is clicked, users will get an introduction in the masculine format as follows (see Figure 3.6):

Renaud: (the guy's name)

Je suis français.

J'espère devenir chanteur.

These two menus display 11 professions and 11 citizenships in both masculine and feminine formats. The menus are flexible making it easy to add in more options and options change their gender format when the web version context is changed by the user clicking between the images. This is because these images depend on the version parameter `gender`. Once it is clicked, the transversion link will swap the gender context of the web page and FFC will automatically produce the correct menu options and sentences.

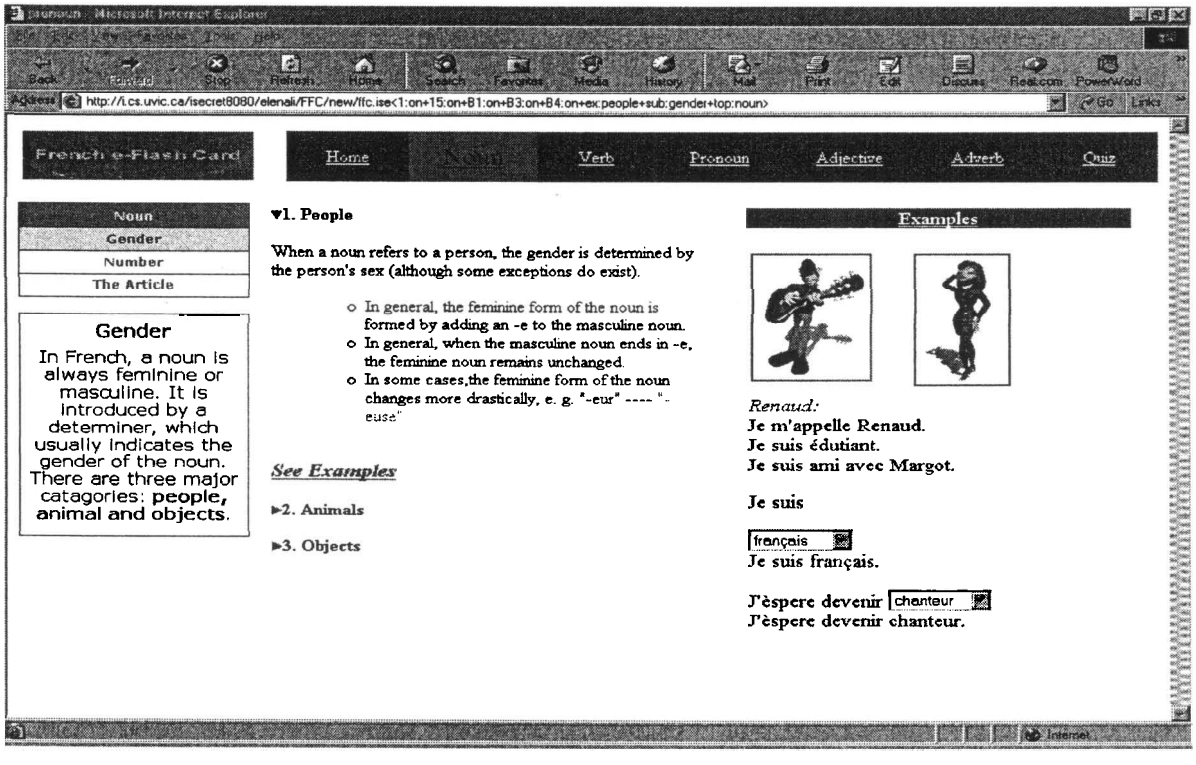

Figure 3.6 Example of masculine nouns

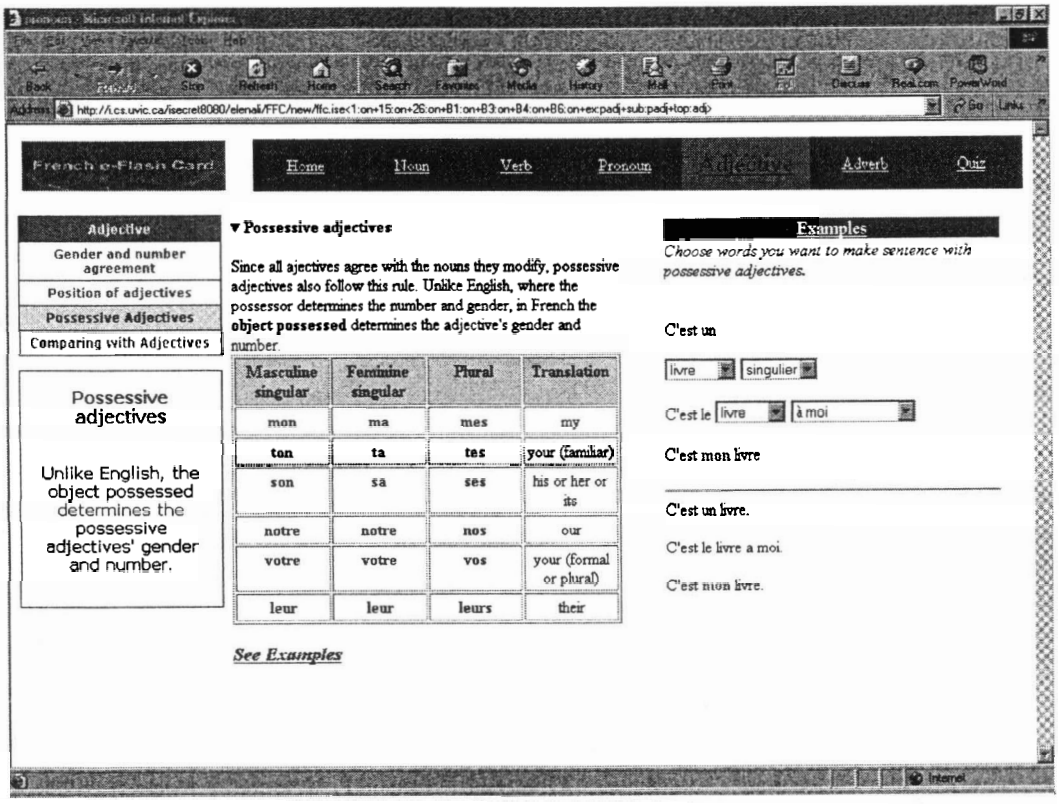

Figure 3.7 Screenshot of possessive adjectives

Figure 3.7 provides another example from menus used in the possessive adjectives section of FFC. In this instance users can make about 60 sentences to cover all possessive adjectives.

3. 5. 4 Search Engine

Besides the principle methods outlined above, user interaction is further aided by the author's provision of a Verb Conjugation Engine unit (see Figure 3.8). Using this facility users can input a French verb and the verb conjugation engine will provide a set of conjugations. This has obvious advantages for students but created a problem for the author – namely that of controlling errors in the user's input. In order to overcome this, FFC employs a filtering mechanism to avoid wrong or meaningless inputs (See section 5. 3. 4 for more details).

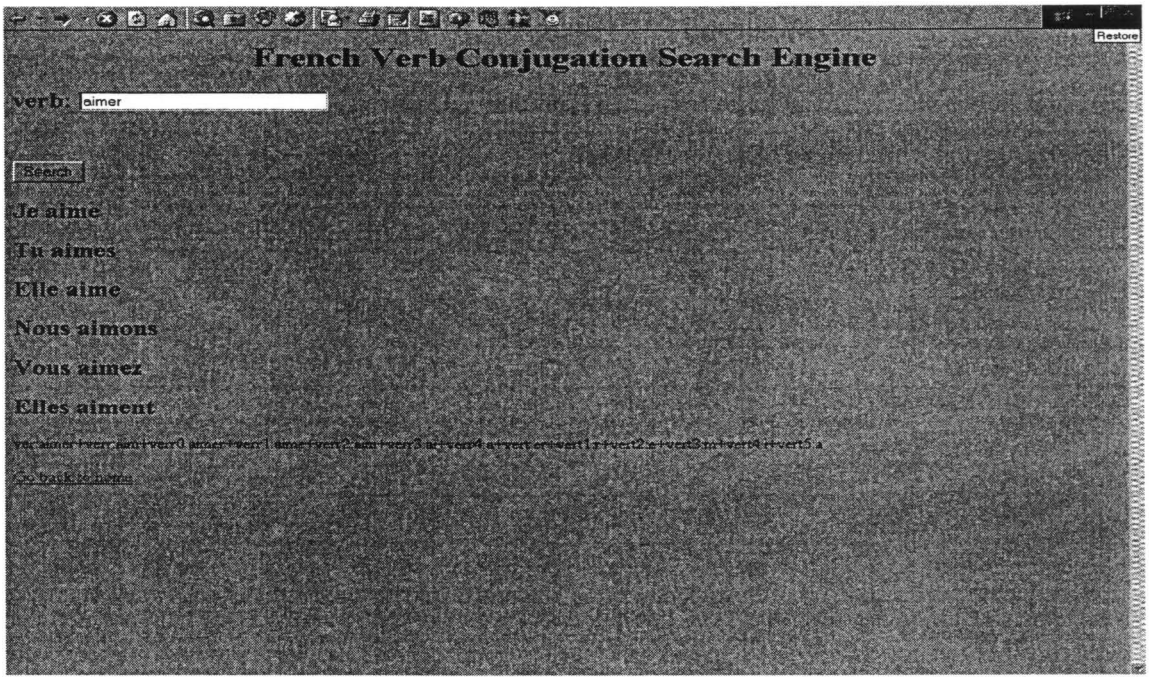

Figure 3.8 Example of search function in FFC

3. 5 Conclusion

The problem with existing formal French education methods and conventional French grammar software is that the atomistic and mechanical method impedes students from learning French in a natural manner. To overcome this failing, the main design idea of FFC is to provide students with extensive customized sentences in order to simulate a learning environment. Several methods for generating sentences are introduced in the chapter as a user manual. In addition, the author expounded the characteristics of FFC's presentation, navigation, and contents.

An explanation of the IML tool and the technical details of FFC implementation will be undertaken in Chapter 4 and Chapter 5 respectively.

Chapter 4 FFC implementation tool: Intensional Markup Language

4. 1 Introduction

Intensional Markup Language (IML) is a simple markup language that extends HTML. It provides multidimensional website developing functions through the setting of a collection of parameters.

IML has the following major characteristics that distinguish it from the other web authoring tools.

- Use of a parametric approach.
- Implementation on top of ISE, a full programming language
- Extensibility
- Ease of use

The author designed French Flash Card (FFC) to be a multi-version, dynamic, artificial intelligence-involved and web-based French language educational software. These specifications require that the programming tool should support multiple dimensions in a website developing domain. Moreover, the tool can produce a customized version on demand.

IML satisfies FFC's programming criteria in the following ways:

- It supports multi-version website development by using a parametric approach. The author controls the extent to which the reader can alter the parameters, whether by following links or by using menus and forms. An important advantage of the parametric approach is that parts of a website can share generic elements by sharing parameters. Furthermore,

individuals can customize their shared pages by altering some of the parameters. [2]

- IML is implemented on top of ISE, a Perl-like CGI language. Authors can embed ISE code in their markup, so there is no a priori limit to its expressiveness
- Compared to ISE, IML is simpler, which makes it practical for authors to produce multidimensional web pages in a short time.
- IML is an extensible language. It permits the skilled author to define new macros in order to extend its functional capability.

For these reasons, the author has chosen IML as the programming language for FFC. As the old Chinese saying goes, the right tool will help one to reach one's goal more easily. That explains the importance of choosing the right tool.

4. 1. 2 Chapter overview

The next section of this chapter will explicate the backbone technologies which are the underpinnings of IML. In addition, the author also describes the evolutional process leading from IHTML to ISE and then IML. Section 3 will list the functions of IML. Section 4 will present the new macros created in the process of developing FFC. The evaluation and a comparison of IML with other web-authoring tools such as IXML and DHTML is presented in Appendix D.

4. 2 The backbone of IML

Intensional Markup Language is the front end authoring tool to implement multidimensional web-sites. IML's backbone is ISE, a Perl-like CGI language with run-time parameterization. [2]. An IML package is a collection of Groff macro definitions. It

conceals the complexity of ISE from the general web authors by using Groff macro definition instead of the intensional constructs of ISE. When all these macros are replaced by their definitions, the result is an ISE program. (Groff, unlike troff, does not restrict macro names to two letters.)

4. 2. 1 Evolutional process

The intensional (possible-worlds) approach to versioning was originally developed by Wadge and Plaice [8] for use in configuring families of programs from families of components.

The first such system was Intensional HTML [11], a minimal extension of ordinary HTML. It is not powerful enough to be used in the development of large-scale web pages. Therefore, Swoboda designed and implemented ISE, which is to Perl as IHTML is to HTML [12].

ISE is much more powerful than IHTML. However, it is more difficult for the web author to use efficiently without tedious effort. To improve practical usage of ISE, Wadge [2] provided a set of macro definitions, which translate text with high-level (intensional) markup into ISE source. The essential purpose of IML is to reduce the complexity of ISE syntax for web authors, thus providing a comparatively simple intensional authoring tool, which makes it more feasible to build large scale websites.

4. 2. 2 Publishing mode

The publishing process of IML has three steps:

1. Edit IML source files.
2. Translate IML source files into ISE intensional files which have .ise as the file extension.

3. Interpret ISE files.

When users submit their requests to the particular web pages, the web server will compute the best-fit version of the pages and return them to the users. Figure 4.1 signifies the process of IML publishing process.

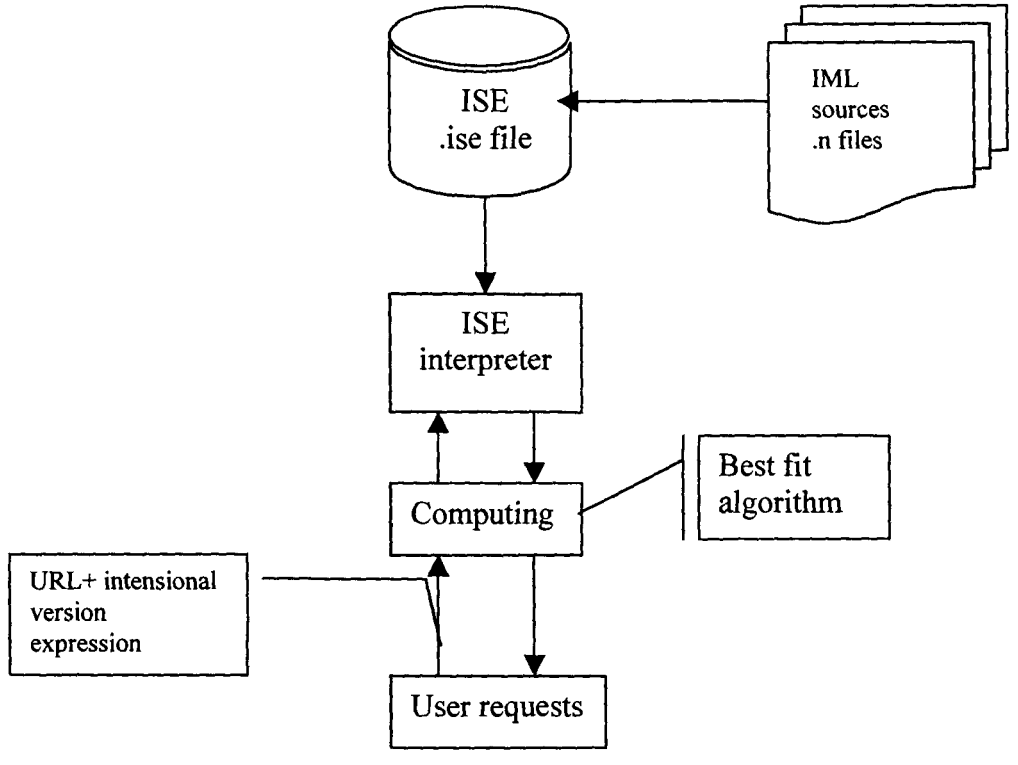

Figure 4.1 IML publishing mode

4. 3 Functions

This section explicates the functions of IML including intensional links, version representation and transformation, layout presentation, menu, debugging, defining and invoking functions, and so on.

4. 3. 1 Macros for intensional links

- Automatic link macro

Automatic link is used to change the current value of a version parameter. As described before, different versions of a web page family are presented and invoked with different version parameters. Therefore, automatic links can be used for inter-version swap.

The format of automatic link is:

```
.balink version_dimension:version_value
anchor_name
.ealink
```

The phrase in italic typeface is user-supplied. An automatic link consists of a pair of tags `.balink` (begin a link) and `.ealink` (end a link) as well as the anchor name. A version expression, which consists of a version dimension and its value follows the tag `.balink`. Once the user clicks the automatic link anchor, the version expression is sent with the original URL to the webserver.

FFC uses a lot of alinks. For example, in the section of "The introduction to nouns" uses alinks to present different examples. Please see Figure 4.2.

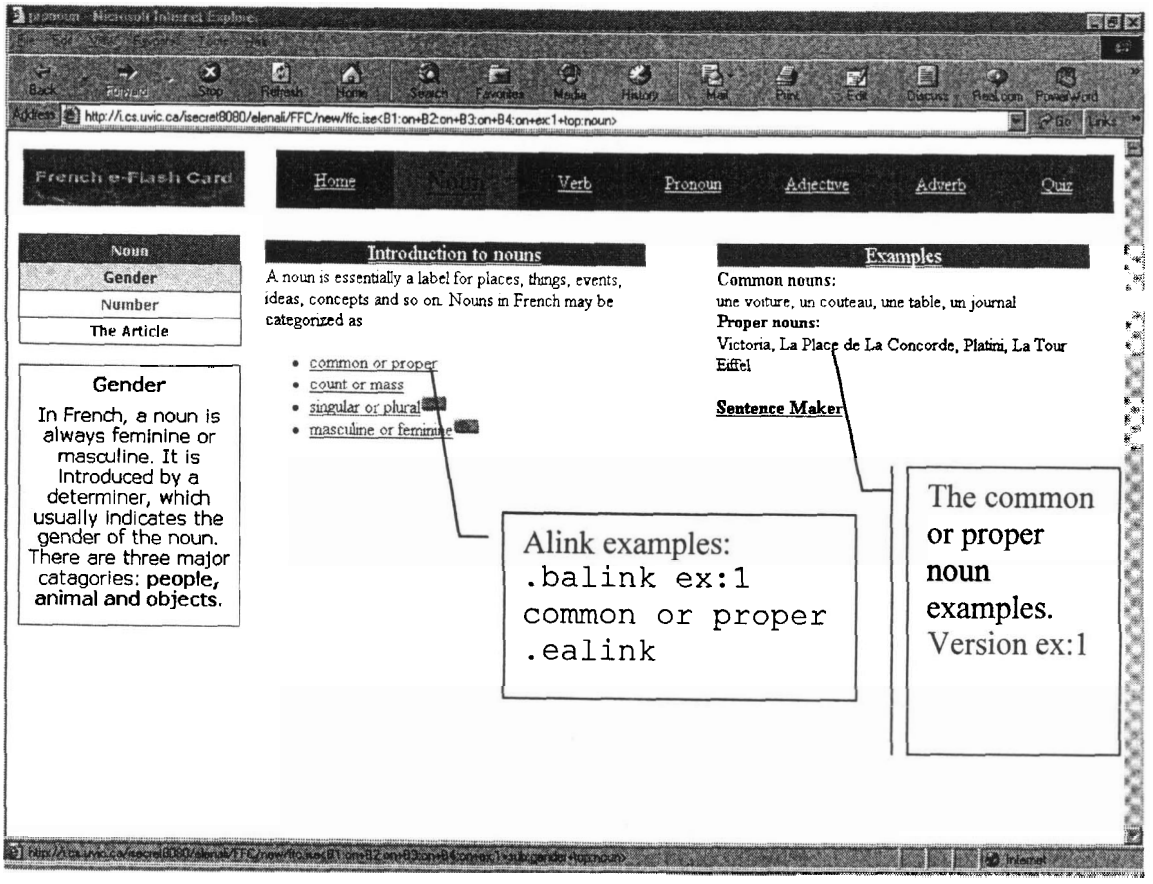

Figure 4.2 Screen shot of the "common and proper noun" section

The author uses an alink to show the user the examples of "common or proper noun".

```
.balink ex:1
common or proper noun
.ealink
```

When the user clicks this link, the current version context is changed to `ex:1`. The corresponding HTML code is as the follow:

```
<a href = "./FFC/new/ffc.ise|ex:1|"> common or
proper</a>
```

4. 3. 2 Macros for version control

- Intensional URL

Macro `.iaurl` is used to set up the context with a version dimension and its value. The format of intensional URL is:

```
.iaurl version_dimension:version_value
```

It will change the current *version_dimension* value to the value it defines. For example, we use

```
.iaurl lg:fr
```

to change the current language dimension `lg` into French `(fr)`. If there is no specified language version dimension in the current context, then it will add this dimension into the current version expression.

The typical usage of `.iaurl` in FFC is to set the version context when the user selects an option from a menu. `.iaurl` is embedded in a menu macro to fulfill this function. For example, the nationality menu in the "gender of noun" section has 11 options for nationality (See Figure 4.3). When the user selects the option *anglais* from the menu, the current version expression will change to `nation:anglais`. Please see menu macro for details.

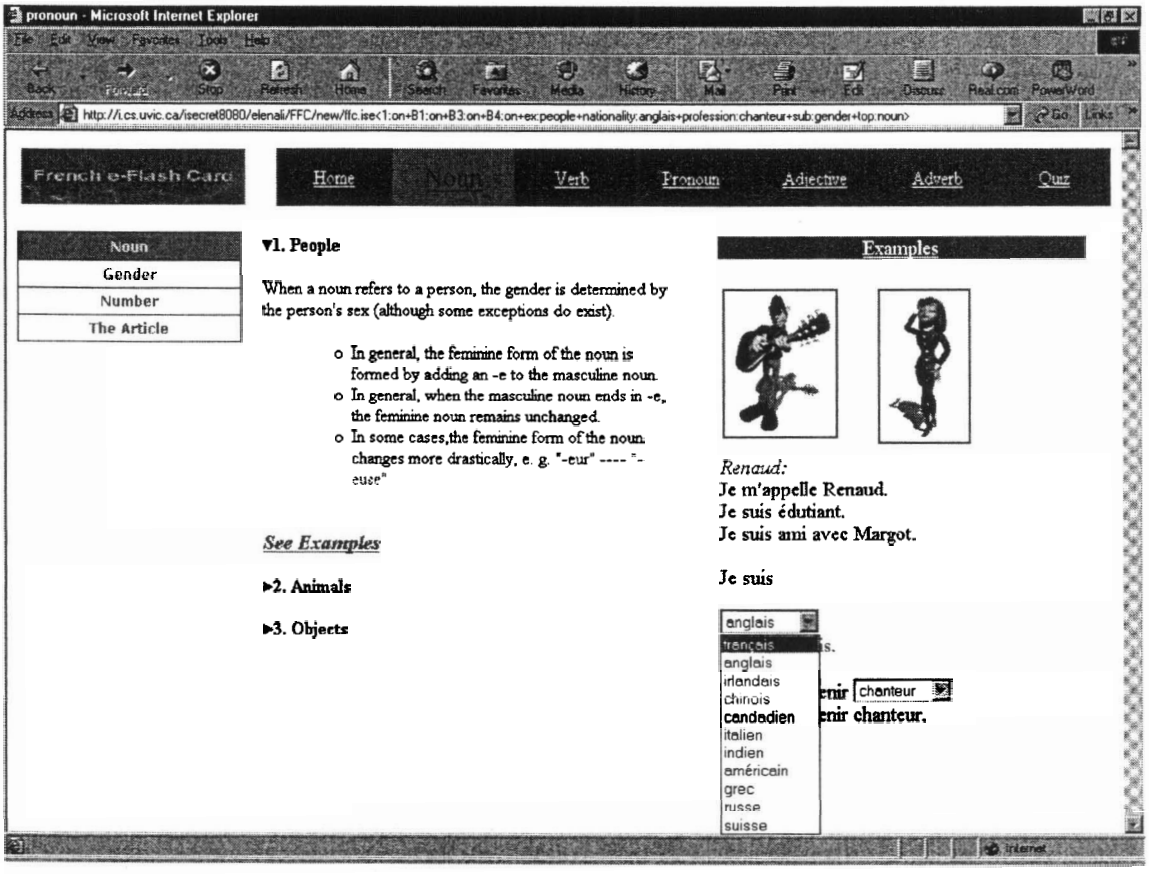

Figure 4.3 Screen shot of nationality menu in the "gender of noun" section

- Switch current version context with macros `.gmod` and `.gcase`
- The macros .gmod and .gcase are used, typically at the beginning of a source file, to specify rules for assigning values to new dimensions (not present in the original url) in terms of the values of dimensions that already have values

Macros `.gmod` and `.gcase` are used to set the version space of a collection of pre-defined `vmod` version expressions, one of which is chosen to be the current vmod version by the best-fit algorithm.

The format is:

```
.bgmod
.gcase current_version_dimension:value set_version_dimension:value
.egmod
```

If the current version dimension is null, then the set version dimension will be the vanilla version expression. Also, multiple gcase phrases can be embedded between `.bgmod` and `.egmod`.

The following code in the "possessive adjective" section of FFC is to specify the gender of the nouns. For example, when the user selects the noun *maison* by clicking the alink which changes the current version as `nom:maison`, this block of code returns the gender version expression as `gen:e`, which explicates that *maison* is a femine word in French.

```
.bgmod
.gcase "" gen:-
.gcase nom:maison gen:e
.gcase nom:chaise gen:e
.gcase nom:voiture gen:e
.egmod
```

- Macros for defining and evaluating intensional variables

To define an intensional variable, we use a pair of macros: `.bdef` to start the definition and `.edef` to end the definition. The format is:

```
.bdef variable_name version_dimension:version_value
variable_value
.edef
```

For example, this pair of macros is used to define layout variables left, right and middle in ffc.n.

```
.bdef middle top:hom
.so homM.i
.edef
```

This block of code will call homM.i file when middle is invoked in a context consistent with `top:hom`, i.e. one in which the paramter `top` has the value `hom`.

Moreover, this pair of macros is always used together with macro `.val`, which is used to invoke a variable. The format of macro `.val` is:

```
.val variable_name
```

- Macro `.biselect` and `.bicase`

They are used as a kind of case statement to choose between pieces of pre-defined code, one of which is chosen by the best-fit algorithm to execute. The format is:

```
.biselect
.bicase version_name:version_value
pre-defined code
.eicase
.eiselect
```

Multiple `.icase` codes can be embedded between `.biselect` and `.eiselect`. One of these blocks of code will be selected to execute when the version expression is the best fit for the situation. If there is no version expression followed `.bicase`, then the code that is defined is considered as the vanilla version.

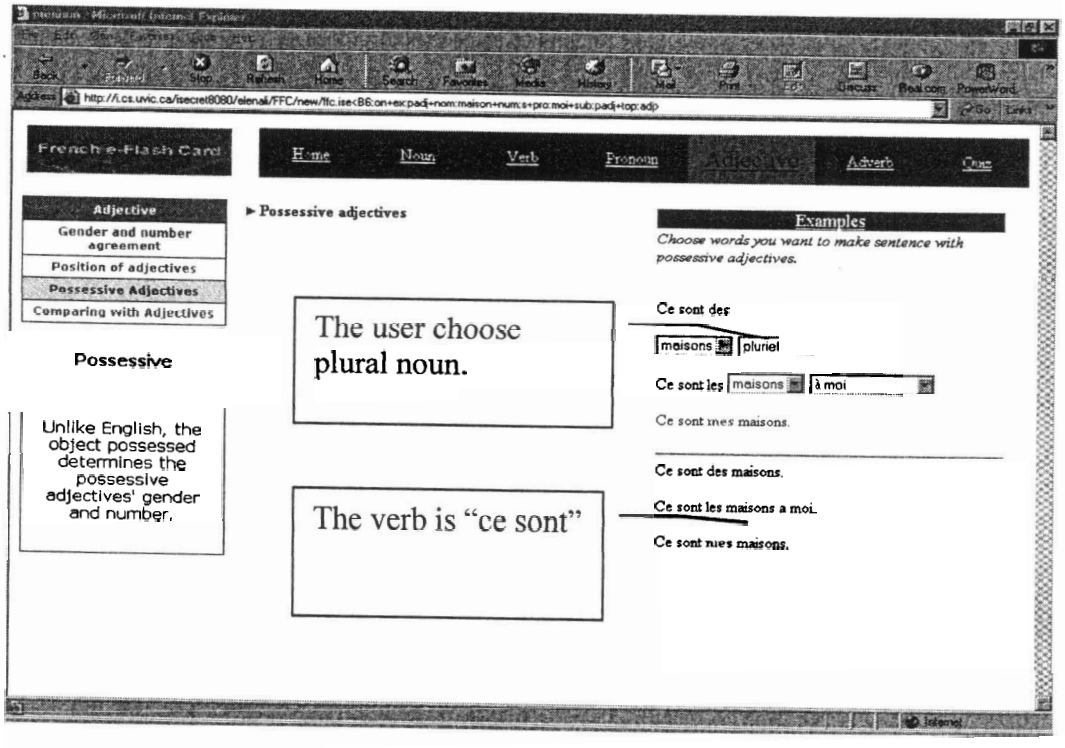

Figure 4.4 Screen shot of Possessive adjective example (1)

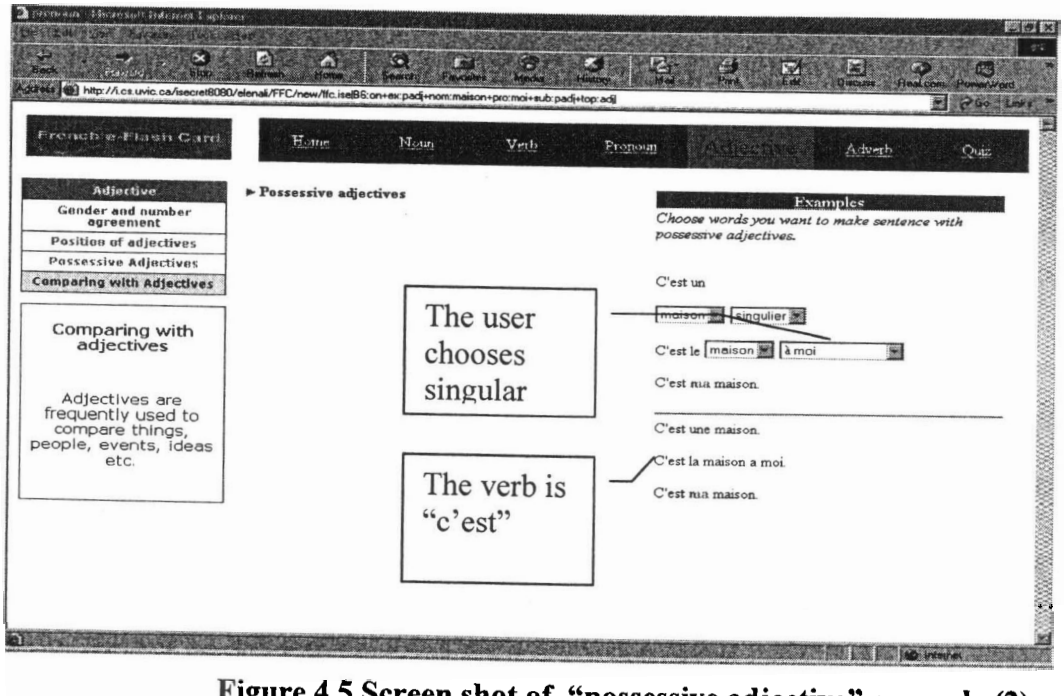

Figure 4.5 Screen shot of "possessive adjective" example (2)

Figure 4.4 and Figure 4.5 show the examples in the "possessive adjective" section. The author uses the macros `.biselect, .eiselect, .bicase` and `.eicase` to construct a set of verb options and the conditions under which the particular verb should be chosen. The conditions are represented by version parameters, which go after `.bicase`, e.g. `num:s` in the following block of code. The right choice of the verb is defined between `.bicase` and `.eicas`, e.g. *Ce sont*. The following code defines: the default value of verb is *C'est*, the singular format of *ce+etre* when the object is in its default singular format (See Figure 4.5). If the object is plural, the version expression becomes `num:s`. Then the sentence should use *ce sont*, which is the third person plural format (See Figure 4.4).

```
.biselect
.bicase
 C'est
.eicase
.bicase num:s
 Ce sont
.eicase
.eiselect
```

4. 3. 3 Initialize version dimension

Macro `.initdim` is used to initialize version dimensions. The format is:

`.initdim version_dimension:value`

For example, in order to initialize the version of gender as feminine, which is represented as `gen:e`, one can use the following format:

```
.initdim gen:e
```

4. 3. 4 Macros for layout definition

- Macros for stretchtext

As defined in [9], "stretchtext is variable-length text, ... where ... the length of the document can be varied according to a global parameter. When the parameter of a stretchtext document is increased, new text appears (not necessarily only at the beginning or end)."

We can use IML to implement stretchtext in a very easy way. The format is:

```
.BC
title
.BD subtitle
contents
.BE
```

Figure 4.6 and Figure 4.7 shows the examples of stretchtext used in "Introduction to adjectives" section of FFC. The initial stretchtext is only a title bar where the title "Introduction to adjectives" is showed in Figure 4.6. If the user is interested in the topic, he or she can click the title bar, then the full text will appear as showed in Figure 4.7. This is a typical use of stretchtext. It is implemented by the following code.

The multiple subtitle blocks can be embedded between macros `.BC` and `.BE`. Let's see an example from FFC.

```
.BC
<font size=+1>Introduction to adjectives</font>
.BD
<p>An adjective is a word that (simplified) ……
.balink sub:gen
Agreement in gender and number of nouns or pronouns
.ealink
(simplified)… …
.BE
```

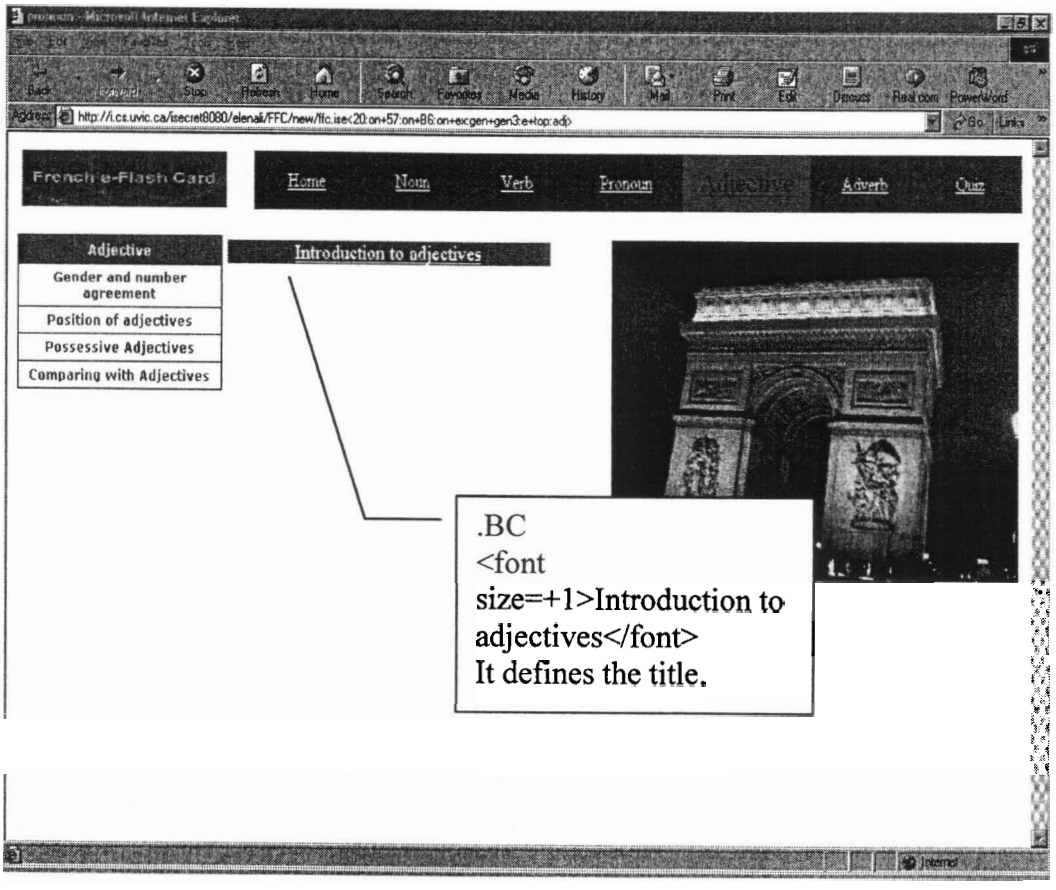

Figure 4.6 Screen shot of stretchtext example (1)

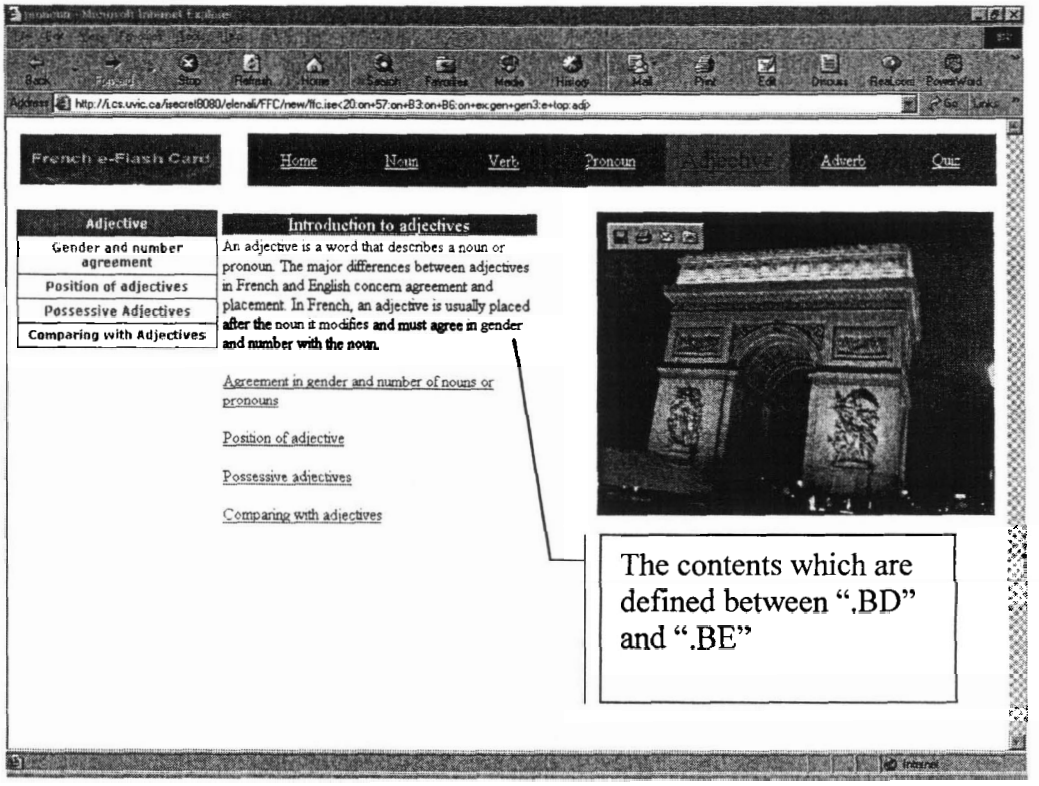

Figure 4.7 Screen shot of stretchtext example (2)

- Macros for droptext

As defined in [9], "droptext initially consists of section headings alone, and the reader chooses which sections to unfold and read. This corresponds roughly to the outline view provided by many word processors (the folding sections can be nested)."

The macro format of droptext is:

```
.bdt heading_size "title"
contents
.edt
```

Figure 4.8 and Figure 4.9 show the examples of droptext used in "Number of noun" section of FFC. Figure 4.8 lists the all topics related to "number of noun" using droptext headings. The user is able to unfold the sections and read the contents by clicking the arrow beside the droptext headings as showed in Figure 4.9.

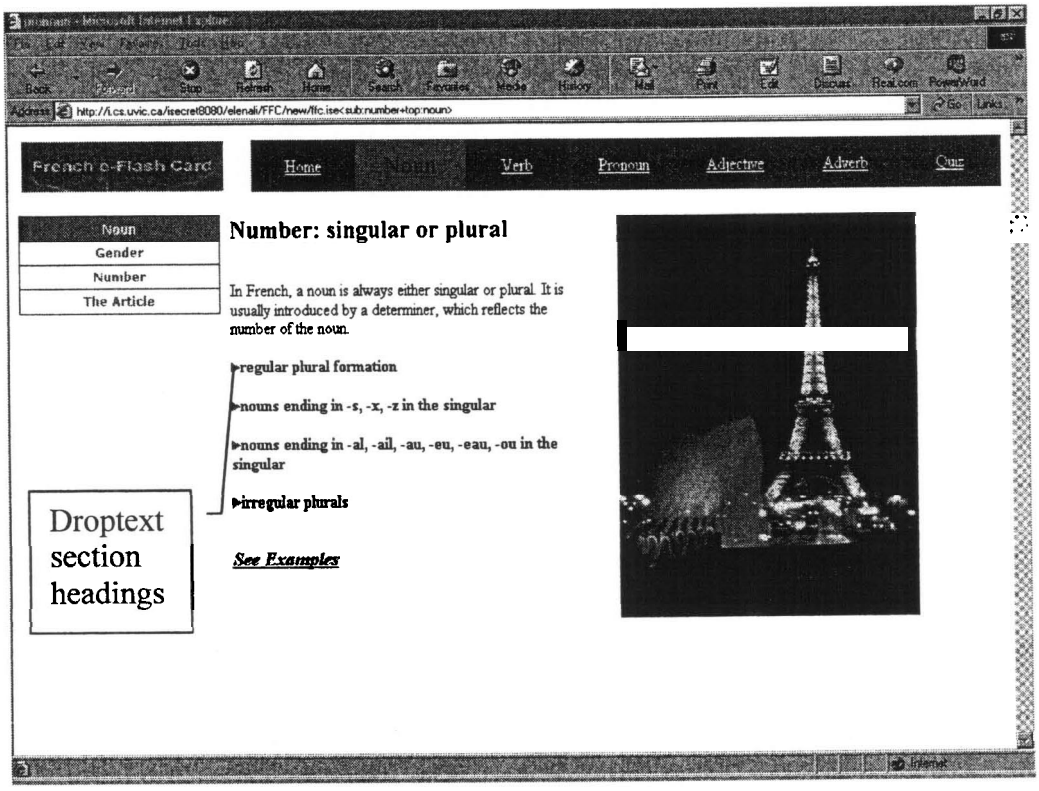

Figure 4.8 Example of droptext (1)

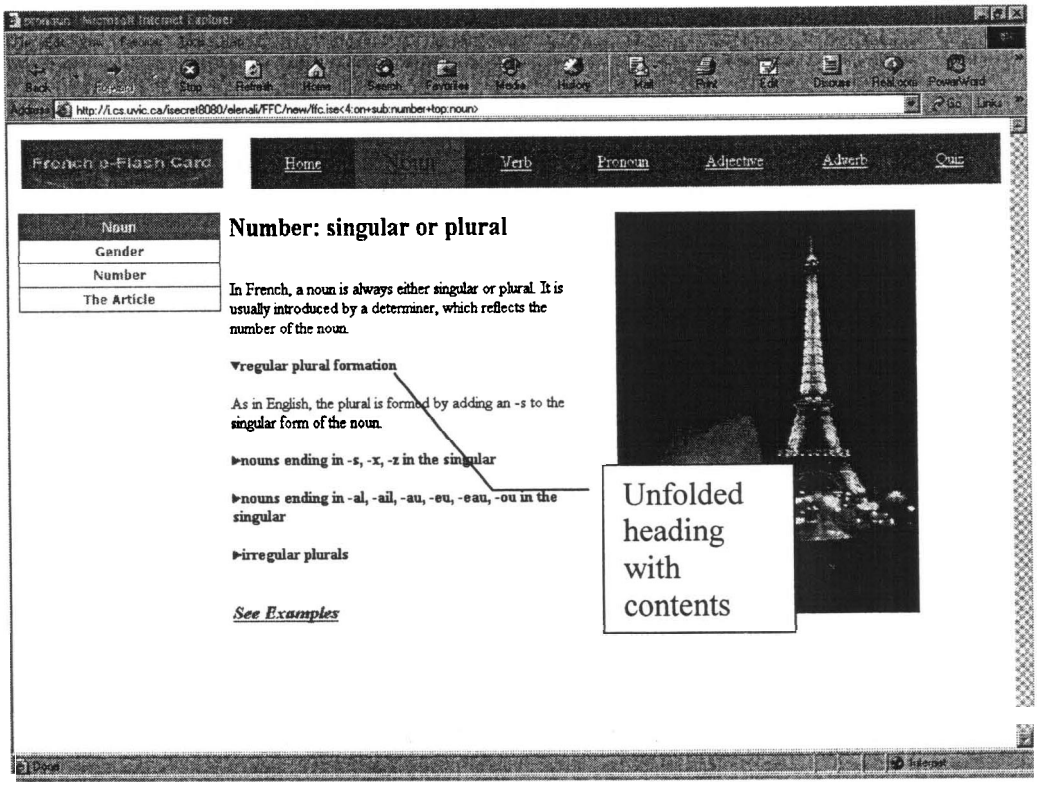

Figure 4.9 Example of droptext (2)

The following IML code is used to implement the above droptext.

```
.bdt 4 "regular plural formation"
As in English, the plural is formed by adding an -s...
.edt
.bdt 4 "nouns ending in -s, -x, -z in the singular"
Nouns ending in -s, -x, or -z do not change in the plural.
.edt
... ...
```

Stretchtext and droptext share the same idea that the user is able to control the length of the text in a web page. In fact, we can use the same IML code as stretchtext to implement droptext macros, except that each section in droptext has its own dimension. By using separate dimensions we can unfold individual captions (hence the name *droptext*). Droptext gives us much finer control over the document than does stretchtext, with its single global dimension.

- Macros for poptext

Poptext shares the same function as droptext, except that the content of section utilizes different layouts. The pop-out content will appear on the right of the section headings instead of underneath the headings (see Figure 4.10). It is a useful way for the web author to control the layout of a page.

The macro format is:

```
.SC
heading
.SD
contents
.SE
Let's see an example.
.SC
Contact information
.SD
RM 331,
Engineering  office wing
.SE
```

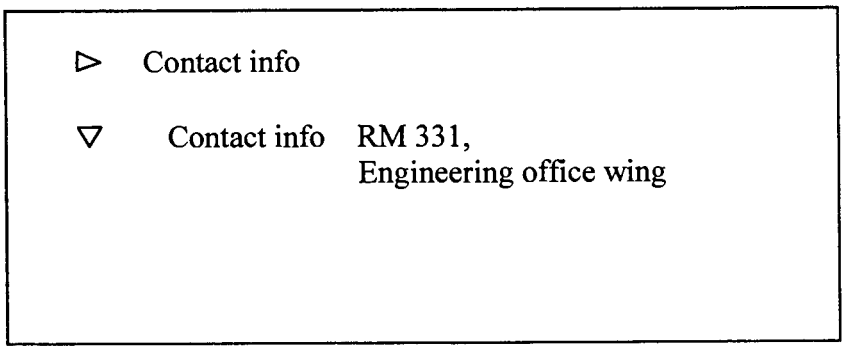

Figure 4.10 Example of Poptext

4. 3. 5 Macros for debugging

The macro `.context` is used to display the current version dimensions and their values. It is useful for debugging. The format is:

```
.context
```

4. 3. 6 Macros for menus

- Menu

IML provides the author with macros `.bmenu`, `.emenu` and `.option` to fulfill the function of menu. The user is able to choose from the options listed in the menu to make sentences. The version parameters are decided by the selection. Different menus are presented as different version dimensions. Each option in the same menu is presented as a different version value.

The format of macros for menu is:

```
.bmenu version_dimension
.option version_value1
option1
.option version_value2
option2
.emenu
```

Multiple options can be defined between .bmenu and .emenu. Figure 4.11 shows an example of menu – the menu of professions in the "gender of noun" section of FFC. The simplified IML code is showed as follows.

```
.bmenu profession
.option chanteur
.option vendeur
.....
.bmenu
```

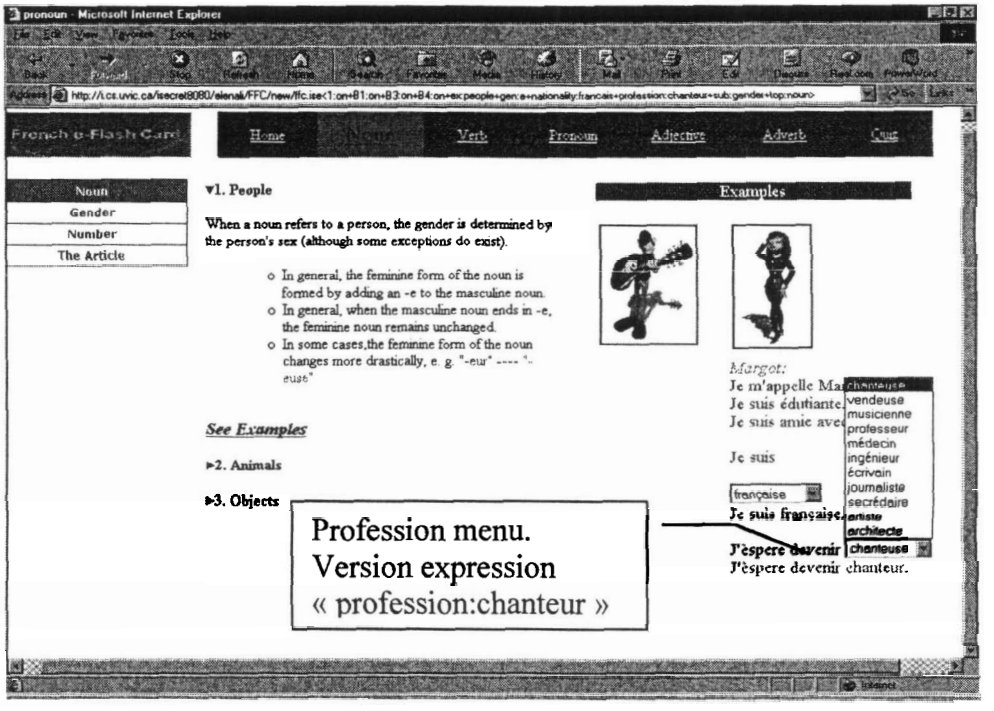

Figure 4.11 Example of Menu in FFC

Bi-gender menu

The menu options can be displayed in masculine or feminine format. The author defined a new macro .gender to fulfill this function. The format is:

```
.gender noun_masculin noun_feminine
```

For example, we can change the above profession menu into bi-gender menu.

```
.bmenu profession
.option chanteur
.gender chanteur chanteuse
```

```
.option vendeur
.gender vendeur vendeuse
.emenu
```

- Bilingual menu

The menu options can be displayed in English or French in a bilingual menu. The format of macros of bilingual menus is :

```
.bling English français
```

4. 3. 7 Macros for extensibility

IML provides web authors with a flexible way to embed HTML, JavaScript or ISE code into their IML code. The authors can import their code in HTML into IML more easily in this way. It also improves the reusability of the legacy system.

- Macros for inserting HTML code

One can insert HTML code into IML using the following macro:

```
.bhtml
html_code
.ehtml
```

- Macros for inserting ISE code

One can insert ISE code into IML by using the following macros:

```
.bise
ise_code
.eise
```

4. 4 Define new macros

IML provides high-level extensibility to web authors. Web authors can not only utilize the available macros as introduced above to create multidimensional websites, they can also define their own tags with the available macros or with ISE primitive tags.

4. 4. 1 Process of defining new macros

IML is implemented on top of ISE as a set of GROFF macros which on expansion produce fragments of ISE code. There are two ways to create new macros, one is to construct a new macro with generic macros, the other is to start from the very beginning that is wrapping primitive ISE commands with Groff standard syntax. For the second scenario, one has to understand ISE as well as Groff in order to create new macros. So it is special for the expert-level user.

No matter which method is used, the definition of a macro always starts with `.de` which means "to define" and ends with a dot which stands for a separation between definition blocks. The format is showed as follows:

```
.de macro_name
macro_definition_block
..
.
```

4. 4. 1. 1 Start from generic macros

The available macro package has about forty predefined macros. Some of them can be combined to fulfill new functions. Moreover, macro definition can include HTML, DHTML or JavaScript code. One can combine existing macros with other language code. For example, the new macro for the intensional (context-switching) button uses the existing `.iaurl` macro and several lines of JavaScript code.

```
.de button
<form><input type="button" name="ibutton"
onclick="window.location.replace(
.iaurl \\$1)" value=\\$2>
..
.
```

Macro for intensional URL `.iaurl` has been defined in macro package "ismacs". Therefore, we can use it directly in our new definition of macro `.button`. The format of `.button` is :

```
.button version_dimension:value button_value
```

When the button is clicked, the version parameter which is assigned with the button is changed and the user accesses a new web page.

For example, to create negative and positive answer buttons, the author used the following code:

```
.button pa:y Oui
.button pa:n Non
```

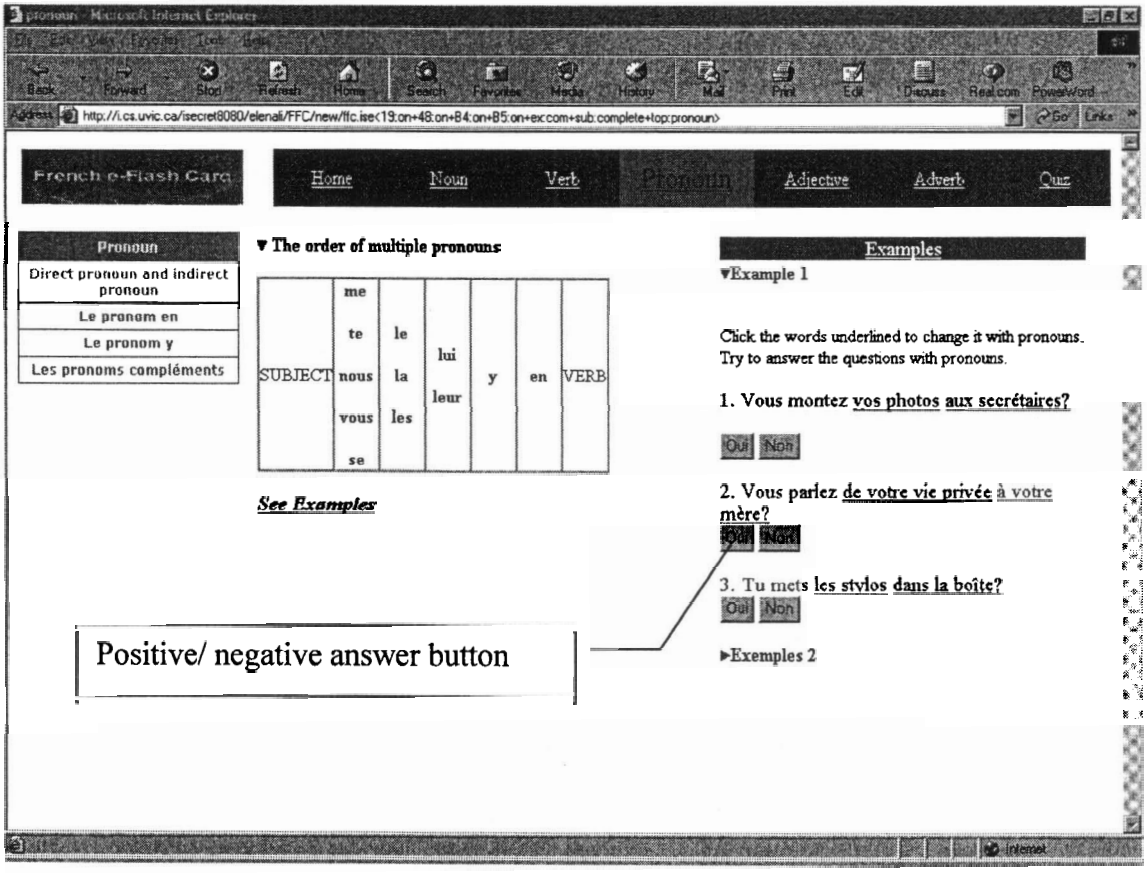

Figure 4.12 Positive and negative answer button

4. 4. 1. 2 Start from primitive ISE

P. Swoboda explained the detailed functions and syntax of ISE in his thesis [12]. In the other hand, users can refer to the online manual of Groff [13] to get necessary information in order to create macros. Here, the author explains the method used to create new macros from ISE with examples.

For example, we use the following ISE code to print a greeting word in English or in French, which depends on the current language version context. In fact, we use the `vswitch` function of ISE.

ISE code:

```
vswitch {
<@[]@><@[lg:fr]@>{print(@[Bonjour!]@);};
<@[lg:en]@>{print(@[Hello!]@);};
}
```

As we know, we can also use a pair of IML macros `.iselect` and `.icase` to fulfill the "greeting".

IML code:

```
.biselect
.bicase
Bonjour!
.eicase
.bicase lg:en
Hello!
.eicase
.eiselect
```

Comparing these two blocks of code, we can get the definition of macros `.b/eiselect` and `.b/eicase`.

The definition of macros:

```
.de biselect
.ehtml
```

```
vswitch{
..
.de bicase
<@[\\$1]@>{
.bhtml
..
.
.de eicase
.ehtml
}
..
.
.de eiselect
};
.bhtml
..
.
```

4. 4. 2 More examples

Please refer to Appendix C for more examples of new definitions of macros.

Chapter 5 French e-Flash Card implementation

5. 1 Introduction

As described in chapter 4, the author used IML as the authoring language for FFC. In this chapter we give the implementation details of the application structure, the data structures, layout and functionality and comparison with traditional educational software.

5. 2 Application Architecture

The FFC French Grammar Knowledge Base handles not only the rules for atomic facts but also those (e.g. for subject-verb agreement or word order) needed to combine the atoms into a complete sentence.

In order to improve the customizability and interactivity, FFC also uses the parametric method to provide the user with customized versions of its web pages. These sentence elements are listed in a table or in a menu. It allows the user to customize his or her sentences with appropriate vocabulary items. Each option in a menu or each item in a table is an intensional link which consists of a version dimension and its version value. Once it is selected by clicking the mouse, the current version expressions will be changed. These different customized versions of the pages are not pre-stored on the server side; instead, they are generated on demand by the same sort of parametric (intensional) logic that the IML implementation uses to generate the French sentences. Figure 5.1 shows the architecture of the FFC.

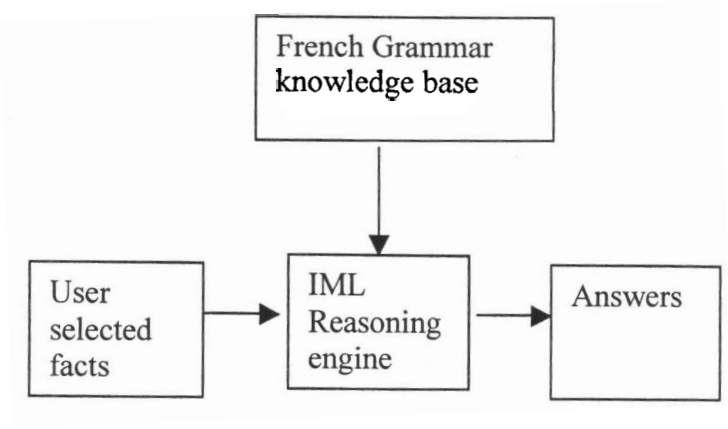

Figure 5.1 FFC Architecture Diagram

There are six grammatical topics in FFC. Each topic has its subtopics. The application structure is presented by Figure 5.2.

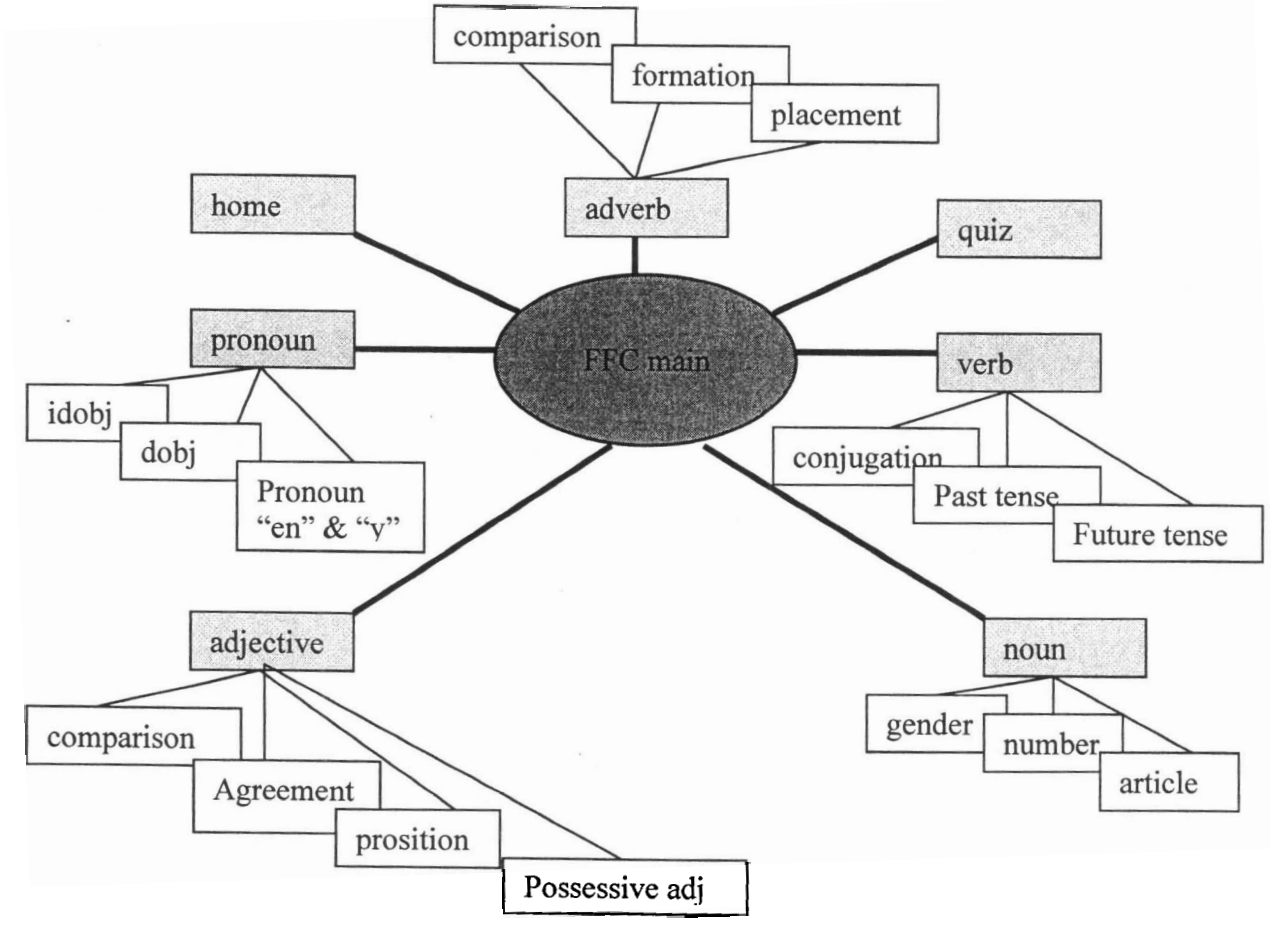

Figure 5. 2 FFC application module

5. 3 Components Structure

The FFC source has a simple tree structure. There are a total of four levels of files. The file in the top level is ffc.m. The second level files include JavaScript menu files *topic_menu_array.js*, the layout files including *topicM.i* and *topicR.i*. The third level files are the example sentence generators for each subtopic, e.g. *enR.i* file is used to create examples for pronoun *en*. The lowest level files are generic macrocodes, e.g. noun analysis macro definition file *announ.i*. Please refer to Figure 5.3 at the end of this section, which shows the date structure.

- The topmost file ffc.m

The top, so-called root file, is *ffc.m*, in which we define the grammatical topics and the layout of the software. It presents the grammatical topics with a set of transversion (context-switching) links and the files which will be invoked. The layout of FFC is identical for all the pages of different topics. It divides the web page into three parts: the left part is JavaScript menus; the middle part is grammar facts for the specified topic; the right part is the example sentence generator. All the files and the version values are defined in ffc.m. For example: the topic of pronoun is defined in *ffc.m* in the following code:

```
{def left top:pronoun}
     {jsmenu/ pronoun}
{/def}
{def middle top:pronoun}
     {so/ pronounM.i}
{/def}
{def right top:pronoun}
     {so/ pronounR.i}
{/def}
```

- The second level files

Under *ffc.m*, the next level of data files includes JavaScript menu files *topic_menu_array.js*, the layout files including *topicM.i* and *topicR.i*. For instance, we defined the subtopics in pronoun as direct pronoun, indirect pronoun and pronoun *en*, and pronoun *y*.

The function of the JavaScript menu file *pronoun_menu_array.js* is to choose among the subtopics and provide a short explanation of each subtopic on the left side.

The file *pronounM.i* contains the grammatical facts for pronouns and each subtopic. For example, we list direct pronouns and indirect pronouns in the middle layer of the presentation. Please see the following screenshot in Figure 5.3.

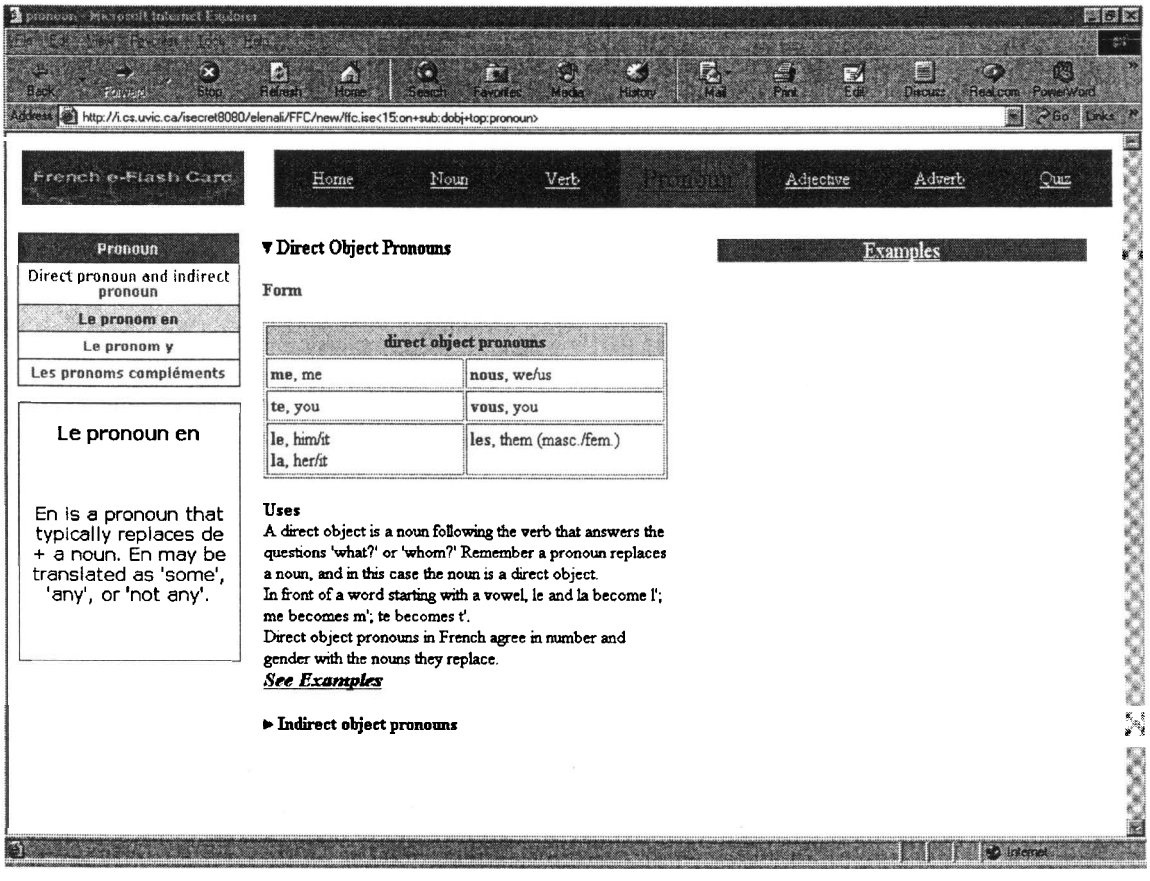

Figure 5. 3 Screenshot of pronoun with direct pronoun table

The file *pronounR.i* contains a set of case statement (`.iselect`) and versioned includes, e.g. `ex:dobj`. The display sentence is computed by invoking the sentence generator files, e.g. *dobj.i.* For example, the following code uses `.iselect` to define the set of version includes and corresponding files invoked. It contains the set of case statements and version includes of the subtopics: direct object pronoun, indirect pronoun, *en* and *y*.

```
.biselect
.bicase ex:dobj
.so dobj.i
.eicase
.bicase ex:iobj
.so iobj.i
.eicase
.bicase ex:en
.so en.i
.eicase
.bicase ex:y
.so y.i
.eicase
.eiselect
```

- Example generators files

These files define the French fragments which can be selected by the user and the mechanism used to generate sentences. From the above code, *dobj.i*, *iobj.i*, *en.i* and *y.i* define the sentence generator of the direct object pronoun, the indirect object pronoun, the pronoun *en* and the pronoun *y* respectively.

These sentence generators focus on one or two subtopics of grammatical knowledge. They correspond to the grammar facts present in the middle layer under the same topic. They are implemented in IML with transversion links `.alink` – each unselected choice is a link to a new version of the page in which the coordinate in one dimension is altered to record the choice made.

Those files have the case statements and versioned includes which compute the sentences. Let's see a simple example code from *iobj.i*. It gives a French sentence *Vous téléphonez a votre mere le dimanche?*. The indirect object *votre mére* is a transversion link. If the user clicks the link, she or he will get the new sentence to answer the question with a pronoun: *Oui, je lui téléphone le dimanche*. What's more, an explanation will be given according to the relevant rules.

These generator files are stored separately to modularize the overall design.. Also it makes it is easy to reuse and revise those files.

```
Vous téléphonez
.balink iobj1:mere
 à votre mére
.ealink
 le dimanche? </font>
.bicase iobj1:mere
<p>Oui, je lui téléphone le dimanche.
<font color="red"><I><b> Explanation</I></font></b>
<p>The indirect pronoun "lui" replaces the noun "mére"
 after the preposition "à".
.eicase
.eiselect
```

- The generic macro definition files

The generic macro files contain the definitions of new macros. These macros are invoked by the other files to fulfill certain functions. For example, announ.i as the noun division macro will be invoked to analyze a noun's root and ending in order to compute gender or number endings. It can be invoked by several programs globally.

```
$noun = @[##noun##]@;
$nount1 = substr($noun,-1,1);
$nount2 = substr($noun,-2,1);
$nounr1 = substr($noun,0,1);
$nounr2 = substr($noun,0,2);
vmod( @[nounr1:$$nounr1$$+nounr2:$$nounr2$$]@ );
vmod( @[nount1:$$nount1$$+nount2:$$nount2$$]@ );
```

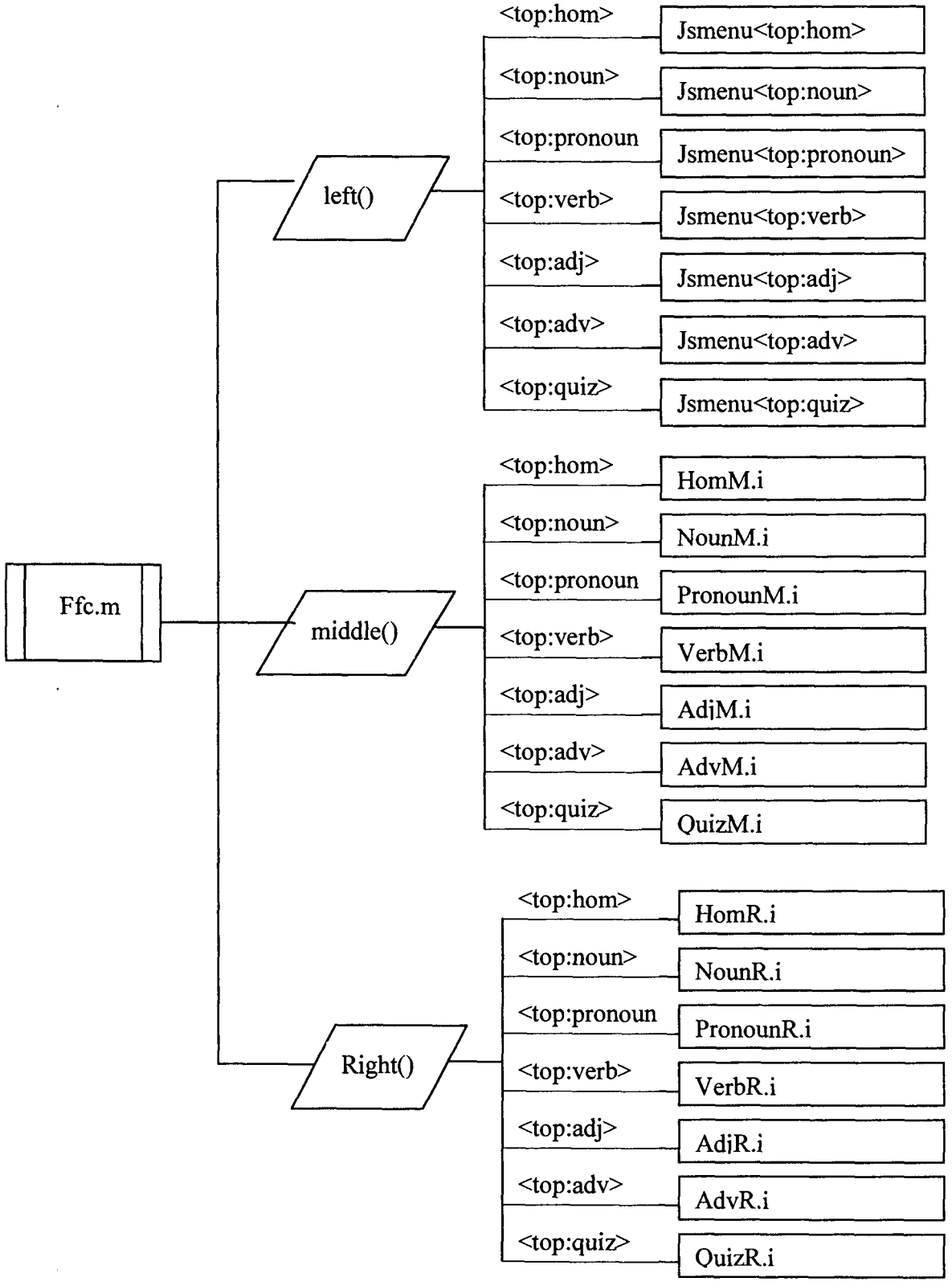

Figure 5.4 Data structure of FFC

5. 4 Layout

As described at the beginning of this chapter, a simple layout is one of the design concepts of FFC. In order to avoid scroll bars, FFC employs a horizontal layout instead of a vertical layout. There are three parts expanding from left to right. See Figure 5.5.

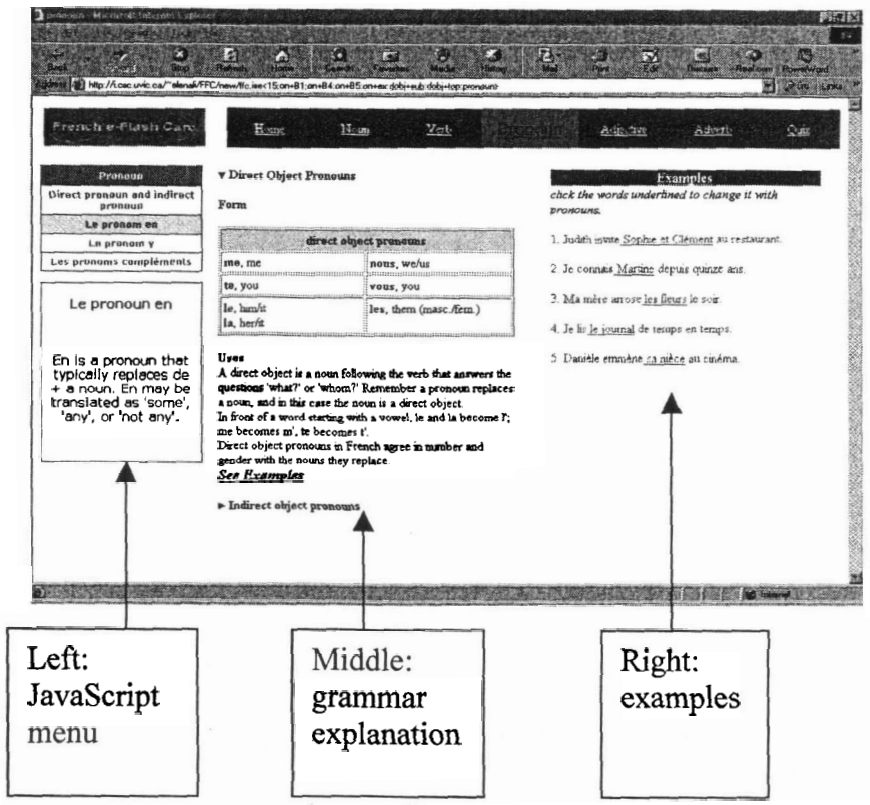

Figure 5. 5. FFC interface

Moreover, the version parameter can determine layout, including background color, font size and color and level of information to be included. The author uses droptext and stretchtext to keep the interface concise and tidy.

5. 5 Features

5. 5. 1 Alternative menus

An alternative menu is a dynamic menu in which the options will be changed automatically depending on the user's previous selections from other related menus. This is one feature that would be hard to do without IML. The author used two kinds of alternative menus: vocabulary-alternative menus and number-alternative menus.

- Number-alternative menu

The options in a series of menus can alternate between the singular form and the plural form. The choice is determined by the current context.

For example, the possessive adjective example generator employs an alternative menu. There are three menus: the object menu, the possessive adjective menu and the number menu, which controls the object in its plural or singular format. Any of these three menus can interact with the others. For instance, if the user chooses the plural option from the third menu, the options in the object menu and the possessive adjective menu will change into the plural form. Figure 5.6 shows the initial status of the menus and Figure 5.7 shows the changes when the user chooses the plural instead of the singular from the "number of object" menu.

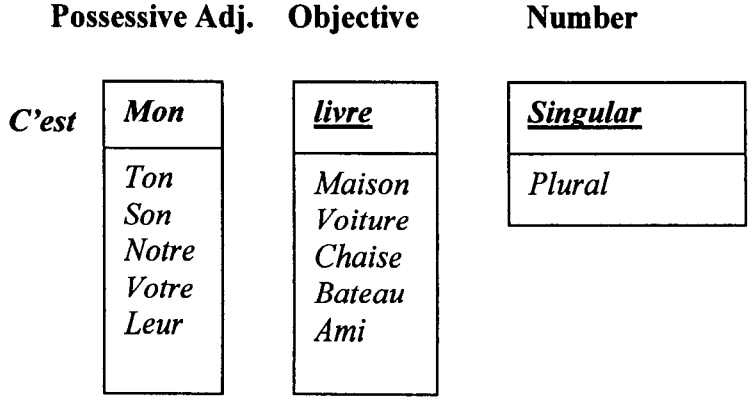

Figure 5.6 Example of Alternative Menus 1

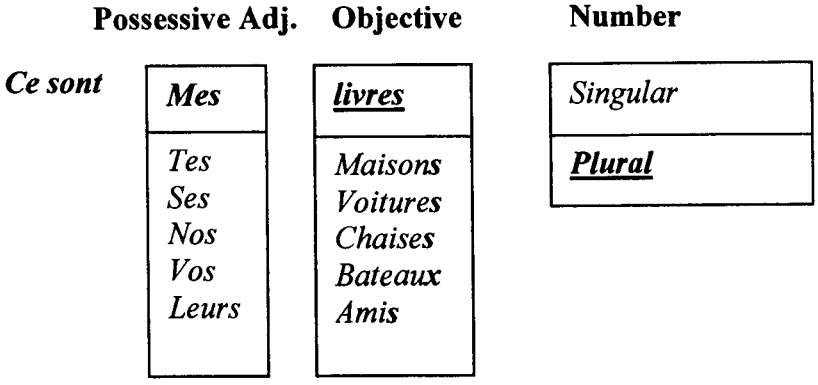

Figure 5.7 Example of Alternative Menus 2

To implement this feature, the basic idea is to define context switching conditions and statements with `.iselect` and `.icase`. The author defines the following macro to implement the number-alternative menu function:

```
.de number
.bnumber
\\$1
.bplu
\\$2
.enumber
..
.de bnumber
.biselect
.bicase
..
.de bplu
.eicase
.bicase num:s
..
.de enumber
.eicase
.eiselect
..
```

- Vocabulary-alternative menu

The options in the vocabulary-alternative menus can be changed into completely different vocabularies according to the previous selections of the user.

For example, the example sentence generator of gender.m in the French noun topic employs the vocabulary-alternative menu. The user can choose verb and object from the verb menu and the object menu respectively. Each verb matches with a different set of objects. Once the user selects one verb, then the corresponding set of options for the object will be presented in the object menu. What's more, the verb menu will present the corresponding conjugated verb according to the selection of the subject. Figure 5.8 and Figure 5.9 show the examples of vocabulary-alternative menus. Figure 5.8 is the initial status of these three menus. The underlined selection is the user's choice. Figure 5.9 shows the updated status of the menus and the resulting sentence after the user choosing different verb.

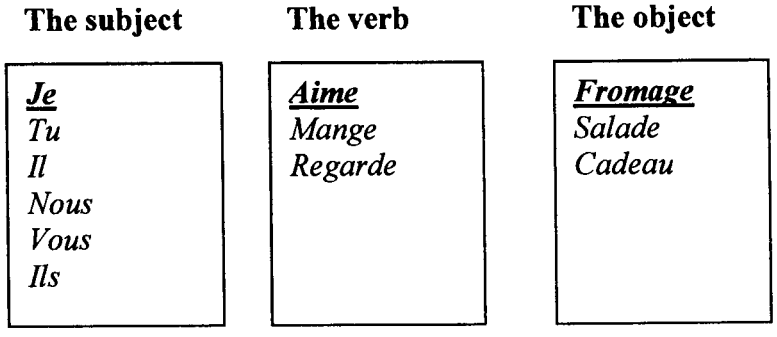

Figure 5.8 The user selection scenario 1

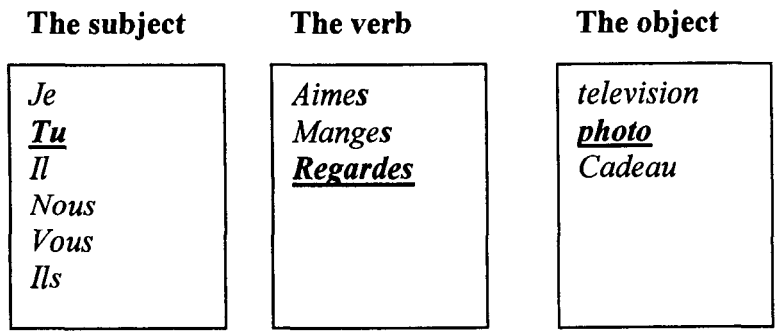

Figure 5.9 The user selection scenario 2

The author utilizes the version switch and transversion links to fulfill the function. Please refer to the macro code in Appendix C.

5. 5. 2 Search Engine

FFC allows the user to look up the conjugation of the verb the in present tense in the Conjugation Search Engine. The user can input his or her inquiries in the blank, and FFC takes the inquiry and searches for the conjugation form.

The search engine works in the following way:

- First, it captures the user's input and stores the string as the value of a parameter.
- Second, the verb analysis process is invoked to cut the string into single letters and store the last two to six letters into a word endings array, and the other letters into word a root array.
- Third, FFC will analyzes the verb ending according to the grammatical rules of French verb conjugation [27].
- Fourth, FFC goes through the irregular verb conjugation database to check whether the verb is irregular. The server side will use the best-fit

algorithm to facilitate the search. If the verb falls into one of the irregular conjugation categories, it will be conjugated according to the specific rule. If the verb is just a regular verb, then the normal conjugation rule will be applied.

- Fifth, the conjugated verb ending will be added to the verb root in order to form the conjugated verb.
- The last step is that FFC returns the result.

Let's see two examples. First, the regular verb *aimer* will be separated into two parts, the verb root "*aim*" and the verb suffix "*er*". After checking through the irregular verb database, there is no record for *aimer*. Therefore, it is automatically defined as a regular verb. The regular conjugation rule is applied to *aimer*. Finally, FFC returns the result to the user. (See Figure 3.8)

The second example invovles the irregular verb *manger*. Verbs ending in *–ger* are irregular verbs which are defined in the irregular verb database. So FFC will conjugate the verb according to the specific rule and return the result to the user.

Chapter 6 Future Work and Conclusion

6.1 Add Multimedia materials in FFC

FFC would be improved in term of functionalities if voice or video clips were included with examples. In fact, IML has the potential to develop multidimensional multimedia resource-sharing mechanism. IML has the capability of embedding JavaScript code. Therefore, we can easily embed the scripts to put voice or video into FFC. What's more, the obvious advantage is sharing the multimedia resource by combining generic slips at run-time instead of preparing all specified voice and video slips and saving them on the disk. This will not only save disk space but it will also simplify the maintenance and reusability. However, the quality of voice and video performance may be affected by network performance. Therefore, more research and investigation may be needed.

6.2 Include more grammatical units

FFC covers some but not yet all of the major grammatical topics of French. More topics would be useful for the French beginner, for example, interrogative adjectives, demonstrative adjectives, the imperative mood of verb, more tenses, such as *le passé composé, l'imparfait, le plus-que-parfait*, and the conditional expressions, and so on.

In addition, FFC could be developed into a comprehensive French language educational application by integrating French vocabulary topics and phonetic topics. Then it could provide a one-stop service to French language students.

6.3 Enhancement to IML

In the process of developing FFC, the author found that there are many possibilities to improve IML, especially the runtime speed and separation of data and

layout. There are many means mentioned in [22], [23], [24] to improve the functionalities of IML. The author is interested in ICC [23] as presented in X. Jin's master thesis. It may be a promising alternative tool to develop a multidimensional web application such as FFC.

6.4 Conclusion

Compared with current French language educational web applications, FFC has the following advantages:

- FFC breaks through the limitations of the traditional model of conventional language web applications, which are all basically web interfaces to a database. In contrast, FFC allows users to immerse themselves in a large collection of demand-generated sentences.
- FFC has dynamic contents and layouts. In addition, it supports customization.
- It has many more examples which can produced with the different options selected by users themselves.
- FFC employs all kinds of changeable menus, which can change the options according to the context. Moreover, it allows a simple form of semantic filtering. It incorporates some simple AI
- Because the sentences change according to different selections chosen by the user, the user is rarely able to go through all the examples at one time. So the longer the user uses the application, the more contents and examples they can extract from the application. It allows users to

generate to new sentences and examples, thus keeps their interest longer. Therefore, FFC promote an active learning curve.

- FFC is easy to maintain and be reused.

The methodology that the author used to create this multiversion French educational application is an authoring formalism in which one can specify whole families of related pages with only modest additional effort to that required to specify a single page. After setting up the grammar rule for some cases, it can produce sentences after taking input from users. This is the artificial-intelligence function of FFC, which is the distinctive characteristic of FFC.

The objective of the thesis is to propose an innovative application of intensional programming in educational language software. The objective is achieved through the development of the intensional multiversion web application French e-Flash Card. In this thesis the author presented FFC in terms of prototype design, intensional web engineering approach and implementation. In the process of describing the development of FFC, the author also explicates that the use of the intensional programming tool Intensional Markup Language. Moreover, the advantages and drawbacks of the IML tool were discussed through the comparison with other intensional programming tools and conventional web authoring tools. What's more, the author demonstrated the methodology of developing multidimensional web application through the developing of the application FFC.

Bibliography

[1] W. W. Wadge and C. Wadge, "Multidimensional French," in Proceeding of the Eleventh Annual International Symposium on Language for Intensional Programming, May 1998, Palo Alto, California, USA, W. W. Wadge, Ed., pp.109-117.

[2] W. W. Wadge, "Intensional Markup Language," in Distributed Communities on the Web, Third International Workshop, DCW 2000, Quebec City, Canada, June 19-21, 2000, Proceedings, ser. Lecture Notes in Computer Science, P. G. Kropf, G. Babin, J. Plaice, and H. Unger, Eds., vol. 1830. Springer, 2000.

[3] W. W. Wadge, "Intensional Logic in Context," Intensional Programming II, World Scientific Publishing, Singapore, 2000, pp1-13.

[4] Ted Honderich, editor. The Oxford Companion to Philosophy. Oxford University Press, 1995.

[5] Brian F. Chellas. Model Logic: An Instruction. Cambridge University Press, 1980.

[6] W. W. Wadge, m.c. schraefel, "A Complementary Approach for Adaptive and Adaptable Hypermedia: Intensional Hypertext," 2001, online available: http://citeseer.nj.nec.com/wadge01complementary.html

[7] G. D. Brown, "Intensional HTML 2: a Practical Approach," M. Sc. Thesis, University of Victoria, Canada. 1998.

[8] J. A. Plaice and W. W. Wadge, "A new approach to version control", IEEE Transactions on software engineering, March 1993, pp. 268-276.

[9] W. W. Wadge, "The Malleable Document," 1999, Online available: http://citeseer.nj.nec.com/530443.html

[10] B. Russell, "The Philosophy of Logical Atomism," Open Court Publishing Company, March 1985, pp188.

[11] T. Yildirim, "Intensional HTML," M. Sc. Thesis, University of Victoria, 1997.

[12] P. Swoboda, "Practical Languages for Intensional Programming," M. Sc. Thesis, University of Victoria, Canada (1999)

[13] Home of Groff, online available: http://www.gnu.org/software/groff/

[14] P. Swoboda, "The Language ISE," unpublished, 1999.

[15] W3C school web page, online available: http://www.w3schools.com

[16] International Organization for Standardization, Geneva. ISO 8879:1986(E). *Information processing – Text and office Systems – Standard Generalized Markup Language (SGML),* 1986.

[17] International Organization for Standardization. ISO/IEC 10179: *Document Style Semantics and Specification Language (DSSSL),* 1996

[18] World Wide Web Consortium. A Proposal for XSL, 1997.

[19] V.T. Phong, *Intensional XML*, M. Sc. Thesis, University of Victoria, 2000.

[20] DHTML tutorial online available: http://www.javascriptkit.com/howto/dhtmlguide2.shtml

[21] World Wide Web Consortium. Document Object Model Specification, 1998. Online available: http://www.w3.org/TR/WD-DOM

[22] Z. Qin, "A Windows GUI for Intensional Web Authoring," M. Sc. Thesis, University of Victoria, 2001.

[23] X. Jin, "Authoring Tools for Intensional Markup," M. Sc. Thesis, University of Victoria, 2003.

[24] L. Liu, "Temporal HTML," M. Sc. Thesis, University of Victoria, 2000.

[25] B. Moa, R. Agarwal, "IML 1.5 Documentation," unpublished, 2002.

[26] A. A. Faustini and W.W. Wadge, "Intensional Programming," Technical Report-DCS-55-IR, Department of Computer Science, University of Victoria, 1986.

[27] Collection Bescherelle, Complete Guide to Conjugating 12000 French Verbs, ISBN 2-89045-808-3, Editions Hurtubise HMH, 1999

[28] B. Wadge and A. Yoder, "The possible-world wide web," in *Intensional Programming* I, M. A. Orgun and E. A. Ashcroft, Eds. World Scientific, 1995.

Appendix A: IML macro reference

```
.rem
     remarks

.bdoc
     generates the boilerplate at the head of every html
page

.edoc
     closing macro corresponding to bdoc

.bhtml
     begin a section of quoted html code

.ehtml
     closes off a section of quoted html code

.bdef
     begin the definition of an intensional demand-
evaluated variable

.edef
     end the definition of an intensional demand-evaluated
variable

.val
     evaluate an intensional variable

.initdim
     initialize a dimension

.bmenu
     begin a menu

.option
     declare one of the menu options

.emenu
     end the menu

.biselect
     begin an intensional case statement

.bicase
     begin an iselect case
```

.eicase
    end a icase

.eiselect
    end the whole iselect

.bise
    begin a chunk of raw ise

.eise
    end a chunk of raw ise

.balink
    begin an auto link

.ealink
    end the auto link

.bdo
    begin a do section

.edo
    end the do section

.vmod
    modify the current context

.pause
    break up a section

.context
    display the current context, mainly for debugging

.randi
    generate a random integer from given range

.randc
    select in a random fashion a string from a given list of strings

.randfoo
    convert a list of strings into array.

.bforeach
    begin a foreach section

.eforeach
    end the foreach section

.ivar
    print the argument supplied to it

.iseval
    print the argument supplied to it

.BC
    begin a stretchtext section

.BD
    begin the text that pops in and out

.BE
    mark the end of stretchtext section

.SC
    begin a poptext section

.SD
    begin the text that pops in and out

.SE
    mark the end of a poptext section

.bdt
    begin a droptext section

.map2
    map its first argument to each two consecutive arguments

.bnf
    begin a negative filtering section

.enf
    end a negative filtering section

.bfi
    begine a postive filtering section

.efi
    end the positive filtering section

```
.bincdim
     increment dimesion link

.eincdim
     end of increment dimension link

.bsec
     begin a new section

.more
     to go the next section

.esec
     end section

.bslides
     begin a new presentation

.fstslide
     go to the first slide

.nxtslide
     go to the next slide

.eslides
     end slides section
```

Appendix B: New Defined IML macro reference

```
.popmenu
     start a pop menu

.poption version_dimension:value
     define a pop menu option under certain version context

.gender
     define masculine and feminine format of the options in
   a menu.

.bquote
     begin a quotation

.equote
     end a quotation

.number
     define singular and plural options in a menu

.baltmenu
     begin an alternative menu

.altoption
     define option in an alternative menu

.ealtmenu
     end an alternative menu

.button
     define a intensional button

.jsmenu
     define a JavaScript menu
```

Appendix C: New Defined IML macro definition

1. popmenu

```
.de popmenu
.BC
..
.de optiontitle
</font>
.ealink
.biselect
.bicase B\\nb:
.eicase
.bicase B\\nb:on
..
.de option
</td></tr>
<tr><td align=cente border=1>
.balink B\\nb:-+\\$1
<font color=black>
..
.de BE
.ealink
.eicase
.eiselect
</font></tr></td></table>
..
```

2. gender

```
.de gender
.bgender
\\$1
.bfem
\\$2
.egender
..
.de bgender
.biselect
.bicase
..
.de bfem
.eicase
.bicase gen:e
..
.de egender
.eicase
```

```
.eiselect
..

3.bquote

.de bquote
.ehtml
vswitch{
..
.de equote
};
.bhtml
..
.de item
<@[\\$1]@>{print(@[\\$2]@); print(@[ ]@); print(@[\\$3]@);}
..

4.number

.de number
.bnumber
\\$1
.bplu
\\$2
.enumber
..
.de bnumber
.biselect
.bicase
..
.de bplu
.eicase
.bicase num:s
..
.de enumber
.eicase
.eiselect
..

5.vocabulary-alternative menu

.de baltmenu
.ds md \\$1
.ehtml
vswitch{
<@[]@>{print(@[]@);}
<@[\\$2]@>{print(@[
```

```
<form>
<select name=ialtmenu
onChange="window.location.replace(options[selectedIndex].va
lue)">
..
.
.de option
<option value=
.iaurl \\*(md:\\$1
.ehtml
vswitch{
 [] {print(@[]@);}
 [@[\\*(md:\\$1]@] {%" selected  "%;}
};
.bhtml
>
..
.de ealtmenu
</select>
.ehtml
}
};
.bhtml
..
.de iaurl
.ehtml
print("\e"http://i.csc.uvic.ca");
print($ENV{"SCRIPT_NAME"});
print("|",[@[\\$1]@],"|");
print("\e"");
.bhtml
..
```

6.button

```
.de button
<form>
<input type="button" name="ibutton"
onclick="window.location.replace(
.iaurl1 \\$1
)" value=\\$2
>
..
.
.de iaurl1
.ehtml
print($ENV{"SCRIPT_NAME"});
```

```
print("|",[@[\\$1]@],"|");
.bhtml
..

7.jsmenu

.de jsmenu
<SCRIPT language=JavaScript src="\\$1_menu_array.js"
type=text/javascript></SCRIPT>
<SCRIPT language=JavaScript src="mmenu.js"
type=text/javascript></SCRIPT>
..
```

Appendix D: Comparison of IML and related web authoring languages

1. IML vs. IHTML 2

As mentioned before, IHTML1 [11] is the first trial with Wadge's version control approach [8] on intensional markup. It was later revised into IHTML2 [7].

IHTML2 brings the multi-version concept into HTML. Many HTML tags, e.g. links, images, font, background and so on, can be versionalized. The most important foundation for IHTML is the best-fit algorithm used to compute the specific version of the web page according to user's requirements. The best-fit algorithm is also the foundation of ISE and IML.

However, IHTML2 is not a programming language. It has many limitations compared with IML. As Wadge pointed out in [2]: "The first is that the extra markup required for even simple effects can be impracticably complex when faced with a document with a large number of droptext sections. The second problem is that IHTML is not a programming language. There are no provisions for evaluating expressions - this rules out constructs because the author cannot generically specify a link in which a parameter is incremented. There is no iterative construct, so that sequences of a similar marked-up copy have to be (re)produced by hand. There are no functions, and this means (among other things) that copy must appear on the rendered page in the same order it appears in the marked up source."

2. IML vs. ISE

In order to remedy the limitation, P. Swoboda implemented a Perl-like scripting language ISE. "ISE is an attempt to create the first imperative scripting language with

versioned identifiers. The basic notion in ISE is that everything that can be identified, including user-defined functions, can also be defined in multiple versions. When constructs such as variables and functions are accessed in expressions, intensional best fits are used to find the appropriate version of the entity referred to" [14]. "Briefly, this means all entities in the language (variables, arrays, functions etc) can be versioned; that programs execute in an implicit context which determines which versions of the relevant entities are used; and that there are also explicit mechanisms for accessing and modifying the context" [9].

However, as Wadge described in [2]: "ISE, however, is still a programming language, and therefore is inherently more complex than a markup system. We therefore developed the Intensional Markup Language IML as a kind of front end to retrieve the obvious advantages of markup authoring."

Therefore, IML makes use of the advantages of ISE as a powerful programming language. At the same time, it simplifies the job of programming for web authors.

3. IML vs. IXML

- Basic concepts of XML

E*x*tensible *Markup L*anguage (XML) is a markup language which is used to describe, store and exchange data. Unlike HTML, XML is designed to focus on what data is instead of how it looks. XML tags are not predefined in XML. One must define ones own tags [15]. XML uses a Document Type Definition (DTD) or an XML Schema to describe the data. In order to display or transform XML, E*x*tensible *S*tylesheet *L*anguage (XSL) is the preferred stylesheet language of XML.

- Relationships among various technologies

The relationships among XML, XSL, CSS, HTML, SGML, DSSSL are a little complicated.

XML is a restricted form of the Standard Generalized Markup Language (SGML) [16], which was first defined to describe document types. The restrictions eliminate some of the more complex and less widely used features of SGML, while preserving its generality.[7]

DSSSL (Document Style Semantics and Specification Language) [17] was defined to supply presentation information for documents conforming to SGML document type definition (DTD). [7]

XSL is to XML as DSSSL is to SGML: it provides presentation information for documents conforming to XML DTDs. XSL is almost a functional subset of DSSSL. [18] With CSS (Cascading Style Sheet) one can add display information to an XML document. CSS is an alternative way to display XML documents. But using XSL as a standard way to formatting XML is more popular in the future. [15].

HTML 4 is an SGML application conforming to International Standard ISO 8879 -- Standard Generalized Markup Language (ISO8879). It describes the set of tags which are legal in HTML, how they work and so on. The main difference between HTML and XML is that they are designed for different goals. XML is about describing information, while HTML is about displaying information.

- IXML

Intensional XML [19] uses XML, XSL, Java, Java Servlet and Cocoon to re-implement IHTML. It stresses that the framework is easily deployable across multiple platforms and Web servers.

- Comparison between IML and IXML

IML and IXML share some similarities with each other. Both IXML and IML target simplifying the tasks of multi-version website development. Moreover, they both use an intensional parametric approach to present different versions of the web page.

However, they are quite different from each other. First, they employ different backbone technologies. As is known, IML is a Groff macro package of ISE. But IXML uses XML and related technologies. Therefore, they have very different syntax and compiling approach. In IML, the author need only program one IML file. Then after it is extended and compiled, the IML file can be viewed with web browsers. But IXML has to program three files: one is a .xml file which specifies the content of the IXML page; the second is name-xsp.xsl which allows the IXML handler to process IXML page; the last one is name-html.xsl which translates the IXML page into HTML presentation. [19].

Second, they use different technologies to handle version parameters. IML depends on CGI to handle version parameters, which is implemented by ISE in Apache web server. So it is platform-dependent. In contrast, IXML relies on the underlying Cocoon web-publishing framework. And its multi-version parameters are based on Java Servlet parameters; the parameters are stored in the session Object. [19] Therefore, IXML is able to deploy cross platforms.

Third, they use different output methods. In IML, there are two ways to output a sentence. One is to output the values of current version dimensions directly; the other is to call the predefined values with macros `.def` and `.val` or `.quote`. In that latter, the output doesn't necessarily have to be the same as the value of the multidimensional

parameter; in the former, it does. This is the difference between the two operations. IXML, on the other hand, uses the iSelect tag to output multi-version content.

Based on the above analysis, we can draw some advantages and disadvantages between IML and IXML. IML has some advantages as follows:

- It only needs to program and compile one source file, therefore it is easy to maintain or update.
- Multiple choice to output the sentences
- Succinct syntax and tags
- More extensible.

However, IXML has its advantage over IML too, namely, its cross platform property, which can be employed in multiple platform and web servers. Moreover, it is good at separating of contents and presentation.

Finally, it must be pointed out that IXML is not currently implemented, except as a prototype which supports only a small fixed set of tags corresponding to IML macros.

4. IML vs. DHTML

Dynamic HTML is the combination of HTML, JavaScript and style sheets (CSS or XSL). It uses a combination of several built-in features of the fourth generation web browser to make the web presentation more dynamic. [20] The glue that holds them together is the Document Object Model (DOM)[21], a standard interface allowing programs to access and modify the content of HTML or XML documents.

One can use DHTML to fulfill some functions that IML presents. Most presentation-related functions can be replaced by DHTML. However, IML has its unique value in the content-related functions. For example, in FFC, each fragment of French

sentence can be presented in version expression. IML can index these version expressions using a simple multidimensional scheme. We can't imagine that DHTML could do the same job without tedious effort.

Compared with DHTML, IML has the following advantages:

- IML is more extensible. It is based on the programming language. It inherits almost all functions of ISE. In addition, expert users are able to define their own macros. Furthermore, it is very easy to embed DHTML, XML, HTML or raw ISE codes.
- IML is better able to handle content-related disciplines.

While DTHML still has its advantages:

1. DHTML has an advantage in presentation over IML.
2. It separates presentation from content. Therefore, it has higher reusability.