A DISTRIBUTED SYSTEM FOR TEACHING SYLLABARY OF A MINORITY LANGUAGE

A Software Framework Development for Learner Classification and Adaptive Testing

BY

GORAN JELICA

A project proposal submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE in INFORMATION SYSTEMS

Athabasca, Alberta

January 2007

©Goran Jelica, 2007
The undersigned certify that they have read and recommended for acceptance the project

A DISTRIBUTED SYSTEM FOR TEACHING SYLLABARY OF A MINORITY LANGUAGE

A Software Framework Development for Learner Classification and Adaptive Testing

submitted by

GORAN JELICA

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE in INFORMATION SYSTEMS

Xiaokun Zhang, Ph.D., Supervisor

Kinshuk, Ph.D., Committee Chair

Charles Van Duren, Committee Examiner

Date: January 2007
For my parents, who departed this world prematurely; thank you for being wonderful parents.
ABSTRACT

This thesis project was an investigation of how to develop a second-language system for learning basic elements of a language with support for multimedia and computer-mediated communications and delivery through a Java enabled Web browser. The implemented system employed some standard Java API-s to realize the desired, rudimentary, probabilistic learner model, which is used to classify the learner in one of available classification (knowledge) states through adaptive testing. The project considered the implementation of the system for a small language group based on syllabic writing system (syllabary). For minority language groups, software development is frequently unaffordable, typically due to financial reasons and the limited number of potential users. Accordingly, the implemented system uses Plains Cree, one of the Canadian Aboriginal languages, as an example of a second language. The implemented system tools are reusable for other Cree dialects and possibly for other languages based on similar writing system.
ACKNOWLEDGMENTS

My gratitude to

Dr. Xiaokun Zhang, my current thesis supervisor and academic advisor, who has provided invaluable guidance through this project.

The committee chair, Dr. Kinshuk, and member, Charles Van Duren, who are also to examine and judge this thesis report.

Dr. Peter Holt, my previous supervisor and sponsor, who obtained all the Cree content used in this application and ensured some support for the prototype from a local Cree teacher. This application is based on the research and prototype implemented during 2004 while working with Peter (primary investigator and originator).

The native Cree speakers, Vivian House (Plains Cree – Y), Stella Spence (Woods Cree – TH), and Garth Noskiye (Northern Plains/Woods Cree), who provided audio files used in this application.

The application testers, Maja & Done (Munich, Germany), Hemnalina & Ilver (Burlington, Ontario), Daco (London, Ontario), and Sinisa (Calgary, Alberta), who generously contributed their effort and time to study and examine this application.
# TABLE OF CONTENTS

## CHAPTER I – INTRODUCTION

- The Opportunity
- Significance of the Opportunity
- Investigated Issues
- Organization of Thesis

## CHAPTER II – REVIEW OF RELATED LITERATURE

- Introduction
- Cree Syllabics
- The Problem Using Syllabics on Computers
- Cree Related Software Projects
  - SIL International
  - The Interactive Cree Language Project
  - The Cree Living Language Encyclopedia Project
  - NEHINAWE-Speak Cree: The Cree Language
  - Universal Syllabic Transcriptor
  - The Ultimate Language Store
- Previous Work on the Problem at Athabasca University

## CHAPTER III – METHODOLOGY

- Development Strategy
- Content Development
Software Design/Development Process........................................14
Solution to the Problem/Opportunity....................................15
  Physical Architecture of the System..................................15
  Client Tier.................................................................15
  Middle Tier..............................................................17
  Data Tier.................................................................18
Logic Model of the System.............................................19
Learner Modeling..........................................................21
  Related Works.........................................................23
  Common Approach....................................................25
  Comparison to Related Works......................................26
Step 1: Domain Knowledge...........................................29
Step 2: Data of the Learner Model....................................31
Conditional Probability of a Correct Answer....................33
Step 3: Joint Probability Distribution.............................35
Step 4: Bayesian Inference: A Naive Bayes Probabilistic Model.36
Initializing the Model..................................................38
Updating the Model.....................................................39
Active Instruction Sequence Logic Model.........................41
Shannon’s Information Entropy........................................43
Step 5: Selecting the Next Symbol...................................44
Learner Classification...................................................47
Step 6: Maximum A Posteriori Classification (MAP)............48
Different Views of the Learner Model

Object-Oriented View

Entity-Relationship View

HCI View

Presentation Control Component

Presentation Views of the Learner Model

Presentation Views of the Learner Model in Syllabics Tutor

Feedback Component

CHAPTER IV - RESULTS

Specification of Software Capabilities

Syllabic Learner

Syllabic Tutor

Simple Text Editor

Syllabic Guessing Game

Syllabic Input Method

Communication Module

Simple Chat System

Simple Conference Board

Video Tool

System Testing

Utility

Reliability
Robustness..................................................................................78
Performances..............................................................................79
  Storage Requirements.............................................................80
  Memory Requirements............................................................81
  System Response Time............................................................82
Correctness..................................................................................87
Success Criteria and System Evaluation......................................88
  Reusability..............................................................................88
  Stability..................................................................................90
Performances..............................................................................91
Platform Independence.............................................................91

CHAPTER V – CONCLUSIONS AND RECOMMENDATIONS..............92
REFERENCES.............................................................................95
APPENDIX A: A Choice of Learner Model Views.............................102
APPENDIX B: Utility Testing Questionnaire....................................105
APPENDIX C: Updating the Probabilistic Model..............................107
APPENDIX D: Adaptive Testing: Selecting the Next Test Item............110
LIST OF TABLES

1. Setup of Conditional Probabilities of a Correct Answer.................................35
2. Description of a Potential HCI User.................................................................53
3. Usability Testing: Participant Profile...............................................................76
4. Utility Testing Results: Syllabic Learner..........................................................76
5. Utility Testing Results: Communications Module............................................76
6. Utility Testing Results: Video Tool.................................................................77
7. Memory Requirements of the System..............................................................81
8. Response Times of the Login Process.............................................................83
9. Response Times of the Conference Board.....................................................85
10. Response Times of Posting a Message on the Conference Board....................85
11. Response Times of the Learner Profile Window............................................86
LIST OF FIGURES

1. Plains Cree Pedagogical Chart.................................................................06
2. Three-Tier Architecture of the System...................................................15
3. Adaptive Instruction Process Control Model.......................................19
4. Physical Distribution of Logic Model...................................................21
5. Elements of Domain Knowledge.........................................................30
6. Example of Fixed Test..........................................................41
7. Example of Adaptive Test..........................................................42
8. Maximum A Posteriori Classification (MAP)..........................................49
9. Object-Oriented View of the Learner Model (UML Diagram)..................51
10. Entity-Relationship View of the Learner Model (ER Diagram)..............52
11. Hierarchy Task Analysis Chart: Practice Session..............................54
12. Hierarchy Task Analysis Chart: Test Session......................................55
13. Presentation Control Interface..........................................................56
14. System Components......................................................................65
15. Session Options.............................................................................66
16. Immediate Feedback.......................................................................67
17. Feedback after a Test Session..........................................................68
18. Simple Text Editor........................................................................69
19. Guessing Game...............................................................................70
20. Visual Input Component..................................................................71
21. Simple Chat System........................................................................72
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22. Simple Conference Board</td>
<td>73</td>
</tr>
<tr>
<td>23. Video Tool</td>
<td>74</td>
</tr>
<tr>
<td>24. Language Options</td>
<td>88</td>
</tr>
<tr>
<td>25. Bar Chart View</td>
<td>102</td>
</tr>
<tr>
<td>26. Pedagogical Layout</td>
<td>102</td>
</tr>
<tr>
<td>27. Table View</td>
<td>103</td>
</tr>
<tr>
<td>28. Additional Information</td>
<td>103</td>
</tr>
<tr>
<td>29. Bar Chart &amp; List of Taken Test Sessions</td>
<td>104</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

The Opportunity

There are many different software systems for teaching second or foreign language in use today. Some are based on centralized architecture, generally with no communication and/or collaboration capabilities, and intended for individual learners in isolation. Some other are based on decentralized or distributed architectures, typically accessible through a standard Web browser, seemingly appropriate for group as well as individual learning, and becoming very prevalent.

Regardless of their internal architecture and target audience, the common characteristic is that many pursue most popular and major spoken languages in the world. This is likely due to market demand since parents are typically teaching their children English or French or Spanish for economic reasons (Cahill, 2003).

Unfortunately, the languages of small cultural groups (also frequently called minority or endangered languages) tend to be neglected. Software for learning some minority languages seems hard to find; for some languages software may not even exist. Apparently, high expenses and limited number of potential users trims down prospects of development of such software for each minority language.
**Significance of the Opportunity**

Many smaller cultural and lingual groups are losing their language. This is especially true for Canadian Aboriginal cultures. Saskatchewan Education, Aboriginal Education Unit (2003) has noticed that “the number of Indian and Métis students who are fluent in their mother tongue has declined to the extent that teachers of Indian or Michif languages now use second language teaching strategies” (para. 1).

While face-to-face immersion is the preferred method for second language learning (Fillerup, 2000), it is not always feasible especially if the quantity of potential learners is small and physically dispersed over a large geographical area. In such a case, a web-based, second-language learning system with communication capabilities can be an important supplementary tool (Hanna & deNoy, 2003) for maintaining and promoting an endangered language.

**Investigated Issues**

This thesis project attempted to develop a second-language learning system for delivery through a Java enabled Web browser, which can be reused with different languages (based on similar writing script) and with the possibility of adding new capabilities.

The project also tried to provide the system with some intelligence. A rudimentary logic process model was developed that monitors the learner responses in order to direct the selection of the next learning instruction with the purpose of responding to the learner needs.
The model is also used to classify the learner in one of several, possible knowledge levels (classification states) after the testing. The approach is based on Bayesian inference.

**Organization of Thesis**

Chapter II reviews writing scripts based on syllables, Cree syllabary, and the problem of using syllabics on computers. Further, this chapter reviews some related projects and the previous work on the same at Athabasca University.

Chapter III describes a possible solution to the investigated problem. The chapter presents a physical architecture and a logical model of the system including the description of their components. The main focus of this chapter is on the probabilistic model and its role in adaptive testing and learner classification. The chapter also presents some other views of the model including object-oriented, entity-relationship, and HCI views.

Chapter IV describes software capabilities and discusses system testing and evaluation.

Chapter V presents conclusions and recommendations.
CHAPTER II

REVIEW OF RELATED LITERATURE

Introduction

According to Ager, there are two kinds of writing scripts that are based on syllables: syllabic alphabets (1998c) and syllabaries (1998b).

Syllabics alphabets, also called alpha syllabaries or abugidas, include symbols for vowels and consonants. In a typical syllabics alphabet, a consonant frequently has an inborn vowel that can transform to another vowel by adding a mark that indicates a special phonetic value. If they are at the beginning or standalone, vowels can be written with separate symbols. A combination of two or more consonants may also lead to special joint symbols. Examples of languages that are based on syllabics alphabets include Tibetan, Bengali, Gurmukhi (Punjabi), Lao, Khmer (Cambodian), and so on (Ager, 1998c).

A syllabary is a phonetic writing system composed of syllabic symbols. A symbol usually consists of a single vowel or a consonant followed by a vowel. A syllabary may also include some special symbols consisting of one or more consonants. Examples of languages that use syllabary writing systems include Japanese Hiragana and Katakana, the Cypriot dialect of Greek, Aboriginal languages such as Cree, Blackfoot, and Cherokee, some ancient languages such as Iberian and Celtiberian, and so on (Ager, 1998b).
The system implemented in this research uses Plains Cree as an example of a language based on a syllabary.

**Cree Syllabics**

"Cree (Nêhiyaw) is a group of closely related Algonquian languages spoken by about 60,000 people in Canada, especially in Ontario, Manitoba, Saskatchewan and Alberta" (Ager, 1998a, Used to Write, para. 1). Cree embraces a number of dialects including Plains Cree, Woods Cree, Moose Cree, Swampy Cree, and so on. As a consequence, there are various different versions of the Cree syllabary applicable to writing different dialects.

Cree legends state that Kisemanito (Creator) presented the syllabic symbols as a gift to the Cree people. The syllabics were given to two Elders, Mistanaskowew (Badger Bull) from Western Canada and Machiminahahtik (Hunting Rod) from Eastern Canada. The legends further state that these Elders introduced the syllabics to James Evans, a Methodist missionary, who then applied the syllabics in the teaching of Christianity to Native People.

Some historians hold a somewhat different point of view, which leans toward Evans as an originator of the Cree syllabic system. Apparently, while working with Ojibwe early in the 19th century, James Evans devised a syllabic system, whose use was disqualified by his superiors. About two decades later, while trying to devise a way for writing Cree language, Evans recalled his Ojibwe syllabic system and adapted it for Cree.
Due to its straightforwardness and simplicity of learning, the Cree syllabic system was adopted by all the Cree dialects. The literacy level among the Cree people was raised to about 80% in the 19th century. Regardless of the viewpoint of the syllabics origin, James Evans became known as "the man who made birch bark talk" (Ager, 1998a, Origin, para. 4).

An example of the Cree syllabic system showing Plains Cree Syllabics and their phonetic representations is given in Figure 1:

![Cree syllabics](image)

**Figure 1:** Plains Cree

Depending on a Cree dialect, the number of symbols varies typically somewhere in between 80 and 100. However, there are consistent rules relating most of the sounds and the corresponding syllabics. This seems to provide a nice constrained domain for an e-learning application (Holt, Zhang, & Jelica, 2004).
The Problem Using Syllabics on Computers

For long time, “the practical use of any non-roman script on a computer remained an exclusive goal” (Jancewicz & MacKenzie, 2002, p. 84). In order to provide input/output support for syllabics (and logographic scripts), early computers required hardware adjustments such as hard coding the entire new script into microprocessors and physically modifying output devices thus replacing the original roman script with syllabics.

During the 1980s, the progress was made with the introduction of personal computer and graphical user interface, which enabled first software solutions to the problem of using syllabics on computers. During this period, SIL International developed the set of tools that allowed definition of customized sets of characters (Reitz, 1988) as support to field linguists working in minority languages such as Cree.

The first software solutions for using syllabics on computers were rather rudimentary, typically based on bit-mapped fonts representing individual syllabics on only one or two pixel grid sizes. These solutions were characterized with low resolution, limited scalability, different font data for different outputs (the screen and printer), and significantly slow printer output.

During the 1990s, the true type font technology was introduced. A noteworthy value was that true type fonts were easily outputted in various point sizes and the same font data was usable for both the screen and printer. This technology brought noticeable publishing means to Cree language. The Apple Corporation introduced an early commercial solution to
the presentation of syllabics on Macintosh computers, which was adopted by the Cree School Board. The drawback was in recurrent changes of font types supporting syllabics causing incompatibilities between works produced by different font types. A different font type also introduced somewhat different form of syllabic input via a keyboard. As a consequence, the users had to re-learn position of individual keyboard keys or to place stickers on the keys for each new font type.

Another setback to the adequate use of syllabics on a computer is due to the fact that Cree communities (and other Native communities that use the same or similar script) are relatively small and geographically dispersed across Canada. While some communities opted to remain tied with commercial vendors and solutions, other communities were working on their own solutions for this problem. Unfortunately, their work appeared rather unsynchronized since the individual communities came up with isolated solutions. As a result, a number of different and incompatible syllabic encoding systems appeared.

The issue of many different and incompatible encoding schemes was common occurrence throughout the entire world. To address this issue, the Unicode Standard was established. The purpose of Unicode is to provide means for encoding (a unique hexadecimal number) every possible symbol used in each of the world’s writing systems that would work on all computer operating systems (The Unicode Consortium, 2005). Unicode has been steadily acknowledged in the software industry since.
Cree Related Software Projects

SIL International

SIL International, also known as the Summer Institute of Linguistics, is an organization that studies, documents, and assists in developing languages of small language groups. It has developed more than 60 pieces of software to support field-linguists working with endangered languages (SIL International, 2005). It should be noted that the computing efforts of SIL are mainly concentrated on providing software tools that help field workers to organize and analyze language data and to prepare the results for publication rather than on those that facilitate language acquisition.

A well-know piece of software developed by SIL is Keyman. Keyman is a software utility for creating and managing keyboard input methods. This utility is suitable for entering text in other languages with no need to alter the current system keyboard. It supports Unicode and some legacy encoding. A possible constraint of this utility is that is limited to MS Windows applications only. Keyman is also used for the inputs of Cree syllabics. However, it requires installation of a keyboard layout designed for a specific Cree dialect (i.e. mappings between Roman keys and Cree syllabics) and fonts supporting the dialect syllabics. Keyman is currently developed and supported by the Tavultesoft software company and is available under the name Tavultesoft Keyman.
**The Interactive Cree Language Project**

The Interactive Cree Language Project was a collaborative effort between Cree Programs of the Cree School Board, and Carleton University linguist, Professor Marie-Odile Junker. Together, they created an interactive web site dedicated to the linguistic documentation and survival of East Cree. This project also included technical research for putting syllabic fonts and sound files on the web, in discussion groups and in relational databases. Their Web sites offer several demonstration lessons containing on-line exercises for learning East Cree syllabics orthography. This project was concluded during 2004. Its continuation is described in the next subsection.

**The Cree Living Language Encyclopedia Project**

This project expands the original ‘Interactive Cree Language Project’. It is a collaborative effort involving James Bay Cree people in the creation of an online Cree Encyclopedic Dictionary and involving other Cree and Innu people in the creation of a Multimedia Linguistic Atlas of the Cree language and dialects. The project is also to create the first ever spell-check for Cree (Junker & MacKenzie, 2004).

**NEHINAWE - Speak Cree: The Cree Language**

This is a small project maintained by an individual, Grant Neufeld. Grant took a Cree language course at Carleton University many years ago. Due to expressed interest in Cree, Grant started the project with the instructor of that course. While Grant still from time to time carries on the work on creating computer-based language training materials, the project
seems to becoming archaic. The project offers several text-based lessons, mainly examples, with no audio support. The project sites offer no support for syllabics; syllabics are represented with its Roman equivalent. The project lessons originate from "A Cree Phrase Book" published by the Department of Native Studies at Brandon University. The project relates to the Cree dialects spoken in Manitoba.

**Universal Syllabic Transcriptor**

Edward Brabec and Ales Polednik, with the co-operation of native people of Saddle Lake Cree First Nation (Alberta), developed an online server based system, which allows users to write a native language just in Standart Roman Orthography, which is instantly transcribed into aboriginal syllabic glyphs as an output. Result of immediate processing output is expression of the native language text in the syllabic glyphs orthography of selected native language or dialect. It is easy to use, with no special programming skills needed by the user. According to the authors, no similar product is known to exist.

**The Ultimate Language Store**

The Ultimate Language Store is an e-commerce site rather than a project. It offers several commercial software products relevant to different Cree dialects including the standalone CD version of First Nations Swampy Cree "N" Dialect Language Lessons at Beginner Level. There is no information about the authors and origins of the software. The software appears intended for the use on a single computer with no support for computer-mediated communications.
Previous Work on the Problem at Athabasca University

During 2004, a prototype of software for teaching Cree syllabics was designed by Dr. Holt and Dr. Zhang, and implemented by myself. This prototype was intended for testing the various approaches to building re-usable software and hypotheses in regard to language learning, student modeling, and human computer interaction (Holt et al., 2004). The prototype features include a syllabic tutor with a simple text editor, video tool for playing media files, and some P2P communication tools.

A possible disadvantage of the original prototype is that it is a standalone, centralized application characterized by large storage requirements, which potentially influenced the application’s performance. Also P2P communications appear limited and obsolete considering that they were reused from an early version of DALE (Distributed Adaptive Learning Environment) (Holt et al., 2001).

The prototype is implemented in this project as a distributed application delivered through the Web browser, by improving the features and eliminating various issues noticeable in the prototype. The significance of this software, regardless of whether it is a standalone prototype or distributed system, is that it is anticipated to be used as a project in a program focusing on Information Technology and Aboriginal Students (Holt, 2003), and in the planned future field-test in the introductory Cree course of Athabasca University (and similar settings if possible) with the full participation of a Cree teacher (Holt et al., 2004).
CHAPTER III

METHODOLOGY

Development Strategy

This project attempted to build a set of tools that can be used for Cree and other languages with similar writing scripts. The system is built with consideration of reusability and expandability in mind.

The logic process model is based on the Bayesian statistical inference, which is to assist the system in the learner classification, and the instruction sequence generation and adaptability to the learner needs. The Bayesian inferential method, and other explored options will be elaborated further later in this discussion.

The system is implemented on the student server “io.acad.athabascau.ca” of the School of Computing and Information Systems at Athabasca University. The implementation is dependent on the middleware and database technologies currently offered and limitations imposed by the student server. The system is built using standard industry tools Java, XML, and MySQL.

Content Development

The proposed system reused the Plains Cree pronunciations and words from the first, standalone system for all its examples. They are supplied by an Aboriginal speaker, Vivian
House (Calling Lake), and are available in reputable literature such as the “Cree: Words” dictionary published by Canadian Plains Research Center, University of Regina. The Cree content also received some guidance from local Cree teachers (Holt et al., 2004).

In order to test the system reusability, two other Cree dialects are added including Woods Cree and Northern Plains/Woods Cree. The pronunciations and words are also reused from the original prototype; they are provided by Stella Spence (Woods Cree) and Garth Noskiye (Northern Plains/Woods Cree) who are also native, Aboriginal speakers.

It should be noted that since the focus of this project is on the system development and associated issues (i.e. reusability, learner model, etc.), the selected language and its content used in the system are of secondary or no importance for this project. Once the system is released to public, modifications and/or corrections of the language content are left to those who may find the system interesting for their own purposes.

**Software Design/Development Process**

The developed, distributed system is built on the experiences and lessons learned while implementing the prototype of the original, standalone system. A rough prototype of the original system was developed and distributed to the Cree teachers to give them some ideas, but not to bias them in their suggestions (Holt et al., 2004). The feedback from the teachers was used to develop the original, standalone system. The current system is based on the functionality and user interface of the original, standalone system. However, changes were necessary to customize the system for the Web delivery and its distributed architecture.
Solution to the Problem/Opportunity

Physical Architecture of the System

The system is implemented using the three-tier architecture as shown in Figure 2, which shows physical distribution of the components of the application.

![Three-Tier Architecture of the System](image)

**Figure 2:** Three-Tier Architecture of the System

**Client Tier**

Two approaches were considered for the implementation of the application client:

1.) A ‘Dumb’ Client

2.) A ‘Semi-intelligent’ Client
The first one considered was the standard thin client consisting of pure HTML. It is called ‘dumb’ since all the intelligence and processing is moved to the middle tier. Any user request is forwarded to the middle tier which processes it and responds with a new HTML based content. The disadvantage of this approach is that it involves a high number of request/response communications with the middle tier. A typical Cree dialect involves between 80 and 100 symbols (Figure 1), the same number of examples (words) and double number of audio files (one for each symbol and each example), which is between 360 and 400 data pieces. In the best case, if we download a symbol, its example and their audio files in a single connection to the server, the application will require at least between 80-100 connections to the server in order for the user to complete a practice or a test session involving the entire symbol set. In addition to this, connections performed by the system to track the learner responses and other updates, download and reload time etc. should also be considered. This approach was considered impractical for this kind of application since it appears that the application would likely operate in a near constant request/respond mode influencing its performance.

The second approach involves more intelligent user interfaces which place some processing on the client side. Consequently, this reduces the number of interactions with the middle tier. It is also appropriate for creating more challenging and interactive GUIs. The implemented client is available to the user in two modes: as a Java applet embedded in HTML and an application delivered via Web-Start through a Web browser. While an applet was initially considered as the sole option for the client, the delivery through Java Web-Start appears as the more current and preferred choice for the delivery of this kind of Java
application. When delivered through Web-Start, the application also shows better performance than the applet. Compared to a ‘dumb’ client, a semi-intelligent client takes a bit more time to start up, but once loaded, the learner has the appearance of working with a local system installed on his/her computer system.

**Middle Tier**

The middle tier consists of a number of Java servlets. They accept the client requests, manage connections with the data layer, process data, and return responses to the client.

The client communicate with the middle tier via serialized Java objects. The data processing and object creation is performed on the server side so that the client receives complete (populated with data) and ready-for-use Java objects. The communications between the client and server could be implemented differently. For example, XML could be used. But in this case, XML processing and object composition would have to be moved to the client side. Consequently, the client size would grow, which did not appear acceptable.

Instead of Java servlets, Java RMI or perhaps Java sockets would probably be better choice for some functions of the implemented system (i.e. Text Chat Module). Considering very limited control the author had over the SCIS student server and AU firewall that protects the server, the use of other access ports, which may be required by RMI or sockets, appeared rather questionable. In order to avoid potential problems, it was decided to handle all the middle layer functions through Java servlets and the already-open standard Web server port 80.
Data Tier

This tier is responsible for managing the application data through the use of a MySQL relational database and files on the student server.

All private data such as learner profiles, test scores, etc. is stored in a MySQL database due to security and privacy considerations and ease of data management.

On the other hand, public data (i.e. language symbols and examples) is stored in XML files. The symbol set of a language is publicly available from many sources including literature and the Internet. It is also considered that a symbol set of a language is constant and highly unlikely to changes in near future. This implies that once created and formatted for the use, updates and modification of the language data are improbable. So storing this data in the database and then converting it to XML appeared rather unnecessary. Holding the public data in XML files appeared sufficient.
Logic Model of the System

The implementation of the system is based on the abstract model depicted in Figure 3.

A learner model can be thought as a set of beliefs that an agent considers about a specific learner (Ayala & Peredes, 2003). The purpose of the learner model is to help the application to adapt to the learner needs. The data for such a model is usually collected and updated through the learner input and the learner’s interaction with the system. The learner modeling is discussed in more details in Section ‘Learner Modeling’.

The Active Instructions Sequence Logic Model (AISLM) represents the generator of instructions that provide learning (practice and test) activities for a learner. AISLM generates instructions personalized for each learner. This is achieved through the application of the
learner model during the instruction sequence synthesis process. AISLM is discussed in Section 'Instruction Sequence Logic Model'.

The presentation control component is an interface between the learner and the rest of the system. It is involved in direct interactions with the user. The presentation component delivers the personalized instructions to the learner and accepts the learner’s input. The presentation control component is depicted in Section ‘Presentation Control Component’.

The feedback loop keeps the learner informed during a (learn or test) session or at the end of a test session. The feedback can also be used to update the learner model. Depending on a session type (a learn or a test), the feedback control may provide information regarding the content to be learned, about the learner’s performance, learning behavior, etc. More details about the system feedback are provided in Sections ‘Feedback’ and ‘Syllabic Tutor’.

A possible way to map the logic model of the system to its physical architecture is shown in Figure 4.
Learner Modeling

The purpose of a learner model, in the context of an adaptive learning system is to provide information about the learner’s knowledge and skills, so that the instruction sequence model can provide instructions tailored to the best advantage of the learner (Ross, Jones, & Millington, 1987). According to Tchetagni and Nkambou (2002), there are two approaches to knowledge representation of a learner through learner modelling, namely state models and procedural models. State models depict the knowledge of a learner at certain stages of the learning process. They are typically used in applications for learning concepts. On the other
hand, procedural models portray the problem solving process applied by the learner. The application of procedural models appears appropriate in new skills acquisition (Anderson & Pelletier, 1992).

Furthermore, there are two common approaches that describe how to represent the knowledge of a learner (Tchetagni & Nkambou, 2002). Overlay models consider the knowledge of a learner as a subset of the domain knowledge. Buggy models go a step further by attempting to capture the learner misconceptions about the subject, which are evaluated according to bug rules included in the system (Tekinerdog, 1995).

According to Shang, Shi, & Chen (2001), some more computationally complex approaches to generating a learner model and representing knowledge/skills include approaches such as the Dempster-Shafer theory of evidence (Bauer, 1996), the fuzzy logic (Hawkes et al., 1990), and Bayesian networks (Murray, 1998; Rudner, 2001; Shang et al., 2001; Bekele & Menzel, 2005; Petrushin & Sinista 1993; Villano, 1992). The Dempster-Shafer theory depicts the learner model through employment of belief functions and plausible reasoning, which are used to combine separate pieces of evidence to compute the probability of an event (Wikipedia, 2006a). Fuzzy logic generates a learner model through reasoning that is approximate rather than precisely deduced from classical predicate logic. It is not a probabilistic approach; fuzzy truth represents membership in vaguely defined sets, not likelihood of some event or condition (Wikipedia, 2006b). Bayesian networks infer the learner model through reasoning with uncertainty, based on probabilistic values assigned to the beliefs (Paredes, 2002).
There are also less computationally demanding approaches such as the model-tracing approach (Anderson et al., 1995). The drawback is that such a model can only trace the learner knowledge, but not behavior and characteristics of a learner (Shang et al., 2001).

The learner model derived for this application is a probabilistic model based on Bayesian inference. This choice was influenced by the nature of the knowledge domain and the demands of the entropy based adaptive testing. The knowledge domain consists of symbols of a syllabary. All symbols are of the same difficulty level. No symbol has a prerequisite and no symbol is a prerequisite for any other symbol. Therefore, all symbols are co-requisites; they should be learned concurrently. Considering the co-requisite organization of the knowledge domain, the application of another kind of model such as overlay may not be so effective. Overlay models are more applicable when domain knowledge is organized in a prerequisite hierarchy so that the progress of a learner can be traced and compared to the domain (Kumar, 1992). Next, the entropy based adaptive testing requires probabilistic variables in order to calculate the current and conditional entropies, and then information gain for each symbol. Without association of probabilities with symbols, adaptive testing may not be feasible due to the lack of any relationship among domain symbols, which would indicate which symbol to administer as next considering the current learner’s answer.

**Related Works**

Murray (1998) presented a probabilistic learner model that is deduced from performance data. The model uses the domain knowledge to determine the performance of
the learner through Bayesian inference. The measure of the learner knowledge is represented
as a joint probability distribution over skill levels. Skill levels are also used to classify the
domain knowledge according to difficulty.

Rudner (2001) described basic steps for creation of a plain probabilistic learner model
based on Bayesian inference. Rudner’s example presents a simple, general case using the
smallest, binary, set of classification states (pass/fail) and a minimal set of only 3 general test
items with no difficulty classification and with focus on specific calculations relevant to
these limited sets of data.

Shang et al. (2001) applied analogous principles to Murray (1998) and Rudner (2001)
to derive their probabilistic model. The difference is that this model considers both, slips and
guesses, in the learner performance data. Shang et al. also represented performance data of a
learner as a joint distribution over skill levels.

Millán et al. (2001) also considered the concept of guesses and slips in their model. In
contrast to Shang et al. (2001), Millán did not consider classification states. As a
consequence, the joint probability functions of these two models are different.

Vomlel (2003) presented a general framework for building strategies using Bayesian
inference. Vomlel discussed the application of the framework to adaptive testing. Vomlel’s
learner model describes relations between the learner skills, abilities, and misconceptions.
The learner model is accompanied by several evidence models. An evidence model describes
skills the learner should have to correctly answer the current test item.

In contrast to others, Bekele and Menzel (2005) tried to use social and personal
attributes instead of the domain knowledge to estimate the learner performance through
Bayesian inference. Their research refers to a case with Ethiopian students.

Common Approach

Taking into consideration the related works, it is possible to reason a common
approach for deriving a probabilistic learner model. The approach may briefly cover the
following principal steps:

Step 1: Specify some domain knowledge; in most cases, the domain is to be used for
determining the performance of a learner.

Step 2: Decide what to model and express it as degrees of belief (probabilities).

Step 3: Based on the modeled data in Step 2, deduce a joint probability function to illustrate
the measure of a learner’s quality (e.g. performances) over some variables of interest (e.g.
classification states).

Step 4: Apply Bayesian inference to measure the probability of an assumption about the
learner.

If the purpose of such a model is to serve adaptive testing and learner classification such as in
the case of this application, additional steps may be added:
Step 5: Select and apply one of available adaptive testing techniques to generate a sequence of test items with regard to the learner’s responses.

Step 6: Select and apply one of available classification techniques to determine the knowledge level of a learner.

Accordingly, this project followed the same steps to arrive to its probabilistic learner model. Briefly described, the presented learner model is deduced from the learner’s performance data. The learner’s knowledge is illustrated as a probability distribution over classification states (knowledge levels). The learner model and learner’s responses are taken into consideration to update the probabilities of possible, alternate learning paths. The realization of individual steps is presented in the later sections of this chapter.

Comparison with Related Works

While this research followed the steps of the general approach outlined above, the uniqueness of this project is that it presents a specific case pertinent to teaching syllabary of a minority language. Since the main references in this project were Shang et al. (2001) and Rudner (2001), other differences will be stated in comparison to their models.

The model of Shang et al. (2001) is tailored for a different kind of application where either the learner or a teaching agent determines curriculum sequencing based on the learner’s ‘learned score’ obtained through the combination of scores on quizzes, and study and review activities. In our application, the model is customized for the sequencing of test
symbols based on Shannon’s entropy and information gain. The test symbol that provides the greatest information gain is administered as next (described in more details under ‘Active Instruction Sequence logic Model’). Next, the model of Shang et al. (2001) is 3-dimensional

\[ M[\text{classification-states}] \times M[\text{topic-categories}] \times N[\text{test-topics}] \]

As well, their model considers the concepts of prerequisites, co-requisites, and related and remedial topics. In contrast, our model is 2-dimensional (M x N)

\[ M[\text{classification-states}] \times N[\text{test-symbols}] \]

The values of M and N differ for each model. The lack of the third dimension (topic categories) is due to the nature of symbols of a syllabary. All test symbols are of the same skill level and there are no prerequisites. Accordingly, all test symbols are co-requisites in the same topic category; they should be learned simultaneously. In addition, to learn a syllabary, the learner must learn all its symbols. Omitting just one or even more symbols will influence the learner’s ability to participate in more advanced syllabary activities (e.g. reading and writing syllabic content). Finally, the probability distribution functions are different. While both models use the concept of ‘guesses’ and ‘slips’, the Shang’s model also considers the counts of correct and incorrect answers.

Compared to the model of Rudner (2001), the first noticeable difference is that our model introduces its own function for distribution of probabilities (4-c). The function allowed inclusion of the concept of ‘guesses’ and ‘slips’. Rudner’s model is based on a direct application of the Bayes Theorem with no consideration for a guess/slip extension. Next, the introduction of the function (4-c) also simplified the initialization of the model. In order to be used, our model requires M + 2 initial values for M classification probabilities, 1
value for the probability of a guess, and 1 value for the probability of a slip. The initial values for the conditional probabilities of N test symbols are calculated through the function (4-c). Rudner’s model requires M x N + M initial values for M classification states and N test items (each test item is associated with M classification states, so we need M x N values). For example, let us assume that there are M = 6 classification states and N = 20 test items. If so, then our model will require 8 initial values, while Rudner’s model will need 126 initial values (which need to be manually set). Finally, it seems that Rudner’s model has an imperfection. According to Rudner (2001), “in the absence of information, equal priors can be assumed” (p. 4) for the initialization of the model. But if equal values are assumed for conditional probabilities of a correct answer (e.g. P(A|K) = n) and different equal values are assumed for the probabilities of classification states (e.g. P(K)=1/M), then Rudner’s model seems to get blocked. Regardless of what the learner’s responses are (e.g. all correct, all incorrect, or mix of correct/incorrect), the model will always produce the same results equal to the initial and equal probabilities of the classification states. Thus, the learner classification and adaptive testing will not be possible. Quite the opposite, our model avoids this trap since the distribution function prevents initial equalization of conditional probabilities across the columns of the model.
**Step 1: Domain Knowledge**

There are two key places for intelligence in an adaptive learning system. One is in the knowledge the system has of its subject domain. The second is in the principles by which it tutors and in the methods by which it applies these principles (Anderson, 1988). Insufficiency in any of these two key areas may lead to instructional ineffectiveness. Also according to Anderson, a powerful learning system must have a significant body of domain knowledge, otherwise the instruction may be poor and incomplete.

The domain knowledge of this application consists of elements, which represents the knowledge of writing systems of a Cree dialect, to be learned by the learner. Each domain element is either:

- **Language Element**: the Cree dialect or other language whose writing system is to be learned. It controls the selection of symbol elements (i.e. only the symbols that belongs to this dialect/language will be selected).

- **Symbol Element**: the main domain elements that represent the actual knowledge to be learned. They represent the knowledge relevant to the writing system of the selected dialect/language.

- **Word (Example) Element**: suplemental domain elements that are used to facilitate understanding of individual symbols; they represent examples of the use of individual symbols and are displayed during the learning session.

- **Audio Element**: also suplemental domain elements that explain how to pronounce the current symbol or its word example; they are also presented during the learning session.
- **Relation**: determine associations between individual languages, symbols, examples and audio files, symbols and presentation components such as the pedagogical keyboard.

Figure 5 depicts domain elements including relations among them. According to the figure, a symbol is the function of a language; the selected language determines the symbols set to be used in the learning process. A symbol element is described by its unicode representation, roman equivalent, type (primary or final), rotation (symmetrical or asymmetrical), and row location in the pedagogical layout table.

Furthermore, a word is the function of a symbol. The selection of a symbol determines which word will be selected as an example of the symbol application. A word is described with its Cree representation (unicode), Roman equivalent, and English translation.
An audio element is the function of a string. The Roman equivalent of the selected symbol (and consequently its word example) determines the audio objects to be presented to the learner.

For each symbol element of the domain knowledge, the learner model maintains some data about the learner’s competence and previous experience with this element. This will be further elaborated in the section below.

**Step 2: Data of the Learner Model**

To provide support for adaptive testing and learner classification, the learner model needs to sustain the following data:

A set of $M$ classification states:

$$ K = \{K_1, K_2, \cdots, K_M\} \quad (1) $$

Each classification state corresponds to a knowledge level. For this project, the following set of knowledge levels is considered

$$ K = \{\text{Newcomer}, \text{Beginner}, \text{Junior}, \text{Intermediate}, \text{Senior}, \text{Expert}\} \quad (1-a) $$

The system maintains a probability for each classification state. The probability of knowledge level $K_j$ will be denoted as follows

$$ P(K_j) \quad (1-b) $$

A set of $N$ test symbols selected from the domain knowledge:
As mentioned before, all test symbols are considered of equal difficulty and there is no separation of test symbols according to the skill/knowledge level. No symbol has a prerequisite and no symbol is a prerequisite for any other symbol. Knowledge of one symbol does not presume or influence knowledge of any other symbol. So they are co-requisites and should be learned concurrently. In contrast to this, most authors (Bekele & Menzel, 2005; Shang et al., 2001; Vomlel, 2003) categorized test questions according to a difficulty or skill level (where a skill level is usually one of the classification states) and specified some prerequisites and/or co-requisites for each test item.

A vector of the learner’s answers to the set of test symbols defined in (2):
\[ \tilde{\mathbf{A}} = A_1, A_2, \cdots, A_N \]
(3)

There will be \( N \) answers, one for each test symbol; each learner’s answer is either true or false.

\[ A_i \in \{ \text{true, false} \} \quad 1 \leq i \leq N \]  
(3-a)

Taking in consideration the learner’s answers (3) and classification states (1), the system maintains the learner’s performance data for each test symbol (2), which is expressed as a probability and called the conditional probability of a correct answer. Thus, the conditional probability of the correct answer \( A_i \) for a symbol \( S_i \) and knowledge level \( K_j \), will be written as follows

\[ P(A_i \mid K_j) \]  
(3-b)
The related works use similar approaches to represent the learner’s responses. The difference appears to be in the level of complexity and notation. For example, Rudner (2001) referred to it as observed data and represents correct/incorrect answers with numerical values 1 and 0. Shang et al. (2001) called it the evidence vector, in which each element contains the number of correct and the number of incorrect answers to test items at a certain difficulty level.

**Conditional Probability of a Correct Answer**

Before defining the joint probability distribution, it is necessary to deduce a way for calculating conditional probabilities of a correct answer (3-b).

Let us assume that a novice learner has no significant knowledge of test symbols and that the learner’s answers are rather guesses. Assuming a sufficiently large (finite) number of test symbols in a test session, there is a chance that the learner will provide one or even more correct answers by guessing. If so, then the probability of a correct answer of a newcomer, regardless how small, is still larger than 0. This probability of the guesswork of a learner represents the minimum probability of a correct answer and is to be denoted as

\[
P(g) \quad (4-a)
\]

Accordingly, let us assume that an expert learner knows all the test symbols, but it is considered that such a learner may perhaps make one or even more mistakes (slips) due to various factors (i.e. distraction, fatigue, a lack of concentration or attention, forgetfulness, etc.). If so, then the probability of a correct answer of an expert, regardless how close to 1, is still smaller than 1. This probability of a mistake or slip is to be denoted as
The maximum probability of a correct answer of a learner is to be calculated as follows:

\[ 1 - P(s) \]

The classification states (1-a) represent an ordinal scale of measurement where each next classification state represents a higher ranking than a previous one. Accordingly, the probability of a correct answer for a ‘Beginner’ will be higher than the same probability for ‘Newcommer’, the probability for ‘Junior’ will be higher than the probability for ‘Beginner’, and so on. Therefore, the probability of a correct answer will be distributed in increasing order from the lowest to the highest classification state.

Considering the classification states (1) and (1-a), if we associate the minimum probability of a correct answer (the probability of a guess \( P(g) \)) with the lowest classification state and the maximum probability of a correct answer (the probability \( 1 - P(s) \)) with the highest classification state and assume that the probabilities of a correct answer are increasingly distributed across remaining classification states, then a possible way to represent remaining conditional probabilities may look as shown in Table 1 (shaded area).
Table 1: Set up of conditional probabilities of a correct answer $P(A_i \mid K_j)$

<table>
<thead>
<tr>
<th>Test Symbol $S_i$</th>
<th>Classification State $K_j$</th>
<th>Conditional Probability $P(A_i \mid K_j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_1 = Newcomer$</td>
<td></td>
<td>$P(g)$</td>
</tr>
<tr>
<td>$K_2 = Beginner$</td>
<td></td>
<td>$P(g)+1\times[1-P(s)-P(g)]/5$</td>
</tr>
<tr>
<td>$K_3 = Junior$</td>
<td></td>
<td>$P(g)+2\times[1-P(s)-P(g)]/5$</td>
</tr>
<tr>
<td>$K_4 = Intermediate$</td>
<td></td>
<td>$P(g)+3\times[1-P(s)-P(g)]/5$</td>
</tr>
<tr>
<td>$K_5 = Senior$</td>
<td></td>
<td>$P(g)+4\times[1-P(s)-P(g)]/5$</td>
</tr>
<tr>
<td>$K_6 = Expert$</td>
<td></td>
<td>$1-P(s)$</td>
</tr>
</tbody>
</table>

Considering Table 1, the general equation that determines conditional probabilities (3-b) of a correct answer across classification states will look as follows:

$$P(A_i \mid K_j) = P(g) + \frac{j-1}{M-1}[1 - P(s) - P(g)]$$

where $1 \leq j \leq M$ (4-c)

Step 3: Joint Probability Distribution

If we symbolize conditional probabilities of a correct answer (3-b) and the probability of the answer vector (3) for a knowledge level $K_j$ as follows

$$P(Y_j) = P(Ai \mid K_j) \text{ and } P(\bar{Y}) = P(\bar{A} \mid K_j)$$

then according to the probability chain rule, the joint probability of variables $Y$ can be expressed as follows

$$P(\bar{Y}) = P(Y_{N} \mid Y_{N-1},...,Y_{1})P(Y_{N-1} \mid Y_{N-2},...,Y_{1})...P(Y_{2} \mid Y_{1})P(Y_{1})$$

which is
\[ P(\vec{Y}) = \prod_{i=1}^{N} P(Y_i \mid Y_{i-1}, \ldots, Y_1) \]

Assuming the conditional independence of the variables \( Y \), we have

\[ P(Y_i) = P(Y_i \mid Y_{i-1}, \ldots, Y_1) \]

and the joint probability will be simplified as follows

\[ P(\vec{Y}) = \prod_{i=1}^{N} P(Y_i) \]

Now, if we apply substitution, the previous can be written as follows

\[ P(\vec{A} \mid K_j) = \prod_{i=1}^{N} P(A_i \mid K_j) \quad \text{where} \quad 1 \leq j \leq M \quad (5-a) \]

Therefore, the probability of the answer vector can be represented as the product of conditional probabilities of a correct answer to individual test symbols (3-b). In view of (4-c), the probability of the answer vector can be rewritten as follows:

\[ P(\vec{A} \mid K_j) = \prod_{i=1}^{N} \left\{ P(g) + \frac{j-1}{M-1} \left[1 - P(s) - P(g) \right] \right\} \quad 1 \leq j \leq M \quad (5-b) \]

**Step 4: Bayesian Inference - A Naive Bayes Probabilistic Model**

Given the vector of the learner’s answers \( \vec{A} \) (3), the posterior probability that the learner has knowledge level \( K_j \) can be obtained by using the Bayes Theorem. In its simplest form, the equation will look as follows

\[ P(K_j \mid \vec{A}) = \frac{P(\vec{A} \cap K_j)}{P(\vec{A})} \quad \text{where} \quad 1 \leq j \leq M \]

Applying the general probability multiplication rule
\[ P(\tilde{A} \cap K_j) = P(\tilde{A} \mid K_j) P(K_j) \]

we have

\[ P(K_j \mid \tilde{A}) = \frac{P(\tilde{A} \mid K_j) P(K_j)}{P(\tilde{A})} \quad \text{where } 1 \leq j \leq M \quad (6-a) \]

Considering the probability of the answer vector (5-b), the posterior probability of the learner’s knowledge level gets the form

\[ P(K_j \mid \tilde{A}) = \frac{P(K_j)}{P(\tilde{A})} \prod_{i=1}^{N} \left\{ P(g) + \frac{j-1}{M-1} \left[ 1 - P(s) - P(g) \right] \right\} \quad 1 \leq j \leq M \quad (6-b) \]

The equation (6-b) represents the naïve Bayes probabilistic model (Wikipedia, 2006e).

In contrast, Rudner’s (2001) model remains with the general form of a probability distribution (5-a) and Bayes Theorem (6-a); a function such as (4-c) is not considered. Shang et al. (2001) model derives a form of naïve Bayes model such as (6-b), but their form is different and more complex since it considers other factors such as counts of correct and incorrect learner’s responses at a specified difficulty level, which are not considered in this project.
**Initializing the Model**

In order to use the probabilistic model, it is necessary to specify the initial prior probabilities of classification states (1-b), and probabilities of a guess (4-a) and a slip (4-b). The initial values for conditional probabilities (3-b) are obtained through (4-c).

The initial values for the classification prior probabilities (1-b) are assumed equal. Considering that

\[ P(K_1) + P(K_2) + \cdots + P(K_M) = 1 \]

we have

\[ P(K_j) = \frac{1}{M} \]

In view of the assumed classification states (1-a), we obtain

\[ P(K_1 = \text{Newcomer}) = 0.167 \]
\[ P(K_2 = \text{Beginner}) = 0.167 \]
\[ P(K_3 = \text{Junior}) = 0.167 \]
\[ P(K_4 = \text{Intermediate}) = 0.167 \]
\[ P(K_5 = \text{Advanced}) = 0.167 \]
\[ P(K_6 = \text{Expert}) = 0.167 \]

These are just initial values; they are recalculated after each answer provided by the learner.

Generally, the probabilities (4-a) and (4-b) are best estimated by a domain expert (i.e. a teacher) based on her/his experience and considering the difficulty level of the domain knowledge (i.e. less difficult test items are easier to guess, while more difficult test items are easier to slip). However, regardless of who estimates these values, they represent subjective beliefs. More objective values are likely if obtained through pilot testing, old examination
records, questionnaires, etc. The following are sample estimates for the probabilities (4-a) and (4-b) used in this project:

\[ P(g) = 0.15 \]
\[ P(s) = 0.10 \]

It should be noted that these values are not permanent; they can be changed. A potential user (i.e. a teacher or an expert) may reset these probabilities to new values, which the user may consider more realistic given specific classroom or other settings.

Different authors use different ways for obtaining the initial data for their models. Rudner (2001) acquired prior data through pilot testing. Bekele & Menzel (2005) got initial data from the school records and questionnaires. The model of Millan et al. (2001) requires that a teacher estimates the required conditional probabilities and probabilities of slips and guesses.

**Updating the Model**

To keep the model current, the conditional probabilities of a correct answer (3-b) and the probabilities of classification states (1-b) should be updated after each learner’s response. Appendix C shows an example of updating the model following the approach described in this subsection. The formula (6-a) is used to calculate the posterior probability of a correct answer for the symbol \( S_j \). Considering the law of total probability

\[ P(A_j) = \sum_{k=1}^{M} P(K_k) P(A_j | K_k) \]

we have
\[ P(K_j \mid A_i) = \frac{P(K_j)P(A_i \mid K_j)}{\sum_{k=1}^{M} P(K_k)P(A_i \mid K_k)} \quad 1 \leq j \leq M \quad (7-a) \]

By applying (4-c), we can rewrite the above equation as follows

\[ P(K_j \mid A_i = \text{true}) = \frac{P(K_j)\left\{P(g) + \frac{j-1}{M-1}\left[1 - P(s) - P(g)\right]\right\}}{\sum_{k=1}^{M} P(K_k)\left\{P(g) + \frac{k-1}{M-1}\left[1 - P(s) - P(g)\right]\right\}} \quad 1 \leq j \leq M \quad (7-b) \]

So if the answer \( A_i \) to the test symbol \( S_i \) is correct (true), the formula (7-b) is used to update the conditional probability of a correct answer. Otherwise if the answer is incorrect (false), since

\[ P(A_i = \text{false} \mid K_j) = 1 - P(A_i = \text{true} \mid K_j) \quad (7-c) \]

formula (7-d) is applicable

\[ P(K_j \mid A_i = \text{false}) = \frac{P(K_j)\left\{1 - \left[P(g) + \frac{j-1}{M-1}\left[1 - P(s) - P(g)\right]\right]\right\}}{\sum_{k=1}^{M} P(K_k)\left\{1 - \left[P(g) + \frac{k-1}{M-1}\left[1 - P(s) - P(g)\right]\right]\right\}} \quad 1 \leq j \leq M \quad (7-d) \]

The meaning of the calculated posterior probabilities \( P(K_j \mid A_i) \) is twofold. First, they represent the new (updated) prior probabilities \( P(A_i \mid K_j) \) of a correct answer for the symbol \( S_i \) (these values are preserved for future needs for calculation of conditional entropies, see the subsection “Step 5: Selecting the Next Symbol”). Second, they also represent current (updated) classification probabilities \( P(K_j) \), which will be used to calculate the posterior probability of the next test symbol. Thus we have \( P(K_j) \equiv P(K_j \mid A_i) \).
**Active Instruction Sequence Logic Model**

The Active Instructions Sequence Logic Model (AISLM) represents the generator of instructions that provide learning activities for a learner. AISLM generates instructions personalized for each learner. This is achieved through the application of the learner model during the instruction sequence control process.

The original prototype, which was developed for a previous project (Holt et al., 2004), used fixed testing to verify the knowledge of a learner. Figure 6 shows an example of a fixed test. Test symbols are randomly mixed and then presented in a sequence one by one until the end of the test sequence is reached. The learner’s answer to the current test symbol has no influence on the selection of the next symbol.

![Figure 6: Example of Fixed Test](image)

The current application applies adaptive or tailored testing to determine the level of the learner’s knowledge. An example of adaptive testing is shown in Figure 7. Adaptive tests
are automatically tailored to the needs of individual learners. The system considers the answer to the current test symbol before deciding which one to display as the next test symbol.

Adaptive testing allows us to maximize information about the learner, update classification probabilities and then determine whether there is sufficient information to end the testing (Rudner, 2001). While it can be implemented to terminate when sufficient information about the learner is achieved, an adaptive test typically terminates after a given number of questions is answered (Vomlel, 2003).

Two approaches for selecting the next test item include Minimum Expected Cost (e.g. Vos, 1999; Rudner, 2001) and the application of Shannon’s entropy to calculate Information Gain (e.g. Rudner, 2001; Vomlel, 2003).

Figure 7: Example of Adaptive Test
Minimum Expected Cost, which is the concept from the Bayesian decision theory, calculates
the cost for each remaining (unused) test item and then selects the one with the lowest cost.
Information Gain, the concept from the information theory, applies Shannon’s information
entropy to obtain information gain for each remaining test item, and then selects the one that
provides the maximum information gain. This approach is selected in this project and
explained in more details in the next subsection.

Shannon’s Information Entropy

Suppose the variable K, which represent classification states (1), can have any of the
following M values

\[ K = \{ K_1, K_2, \ldots, K_M \} \]

If we denote probabilities of the variable K for each state as follows

\[ P(K = K_j) = P(K_j) \quad 1 \leq j \leq M \]

then, according to the definition of entropy (Wikipedia, 2006c), the entropy of distribution
P(K) can be obtained as follows

\[ H(K) = H(K_1, K_2, \ldots, K_M) = - \sum_{j=1}^{M} P(K_j) \log_2 P(K_j) \quad (8) \]

If we want to obtain the entropy of K considering the learner’s input (the vector of answers
(3)), then according to the definition of conditional entropy (Moore, 2003), we have

\[ H(K \mid A_i) = \sum_{k=1}^{2} P(A_i) H(K \mid A_i = value_k) \quad (9) \]

where \( A_i \in \{ true, false \} \) and \( 1 \leq i \leq N \)
In order to calculate the information gain, we subtract the conditional entropy from the entropy of $K$ (Moore, 2003)

$$IG(K \mid A_i) = H(K) - H(K \mid A_i) \tag{10}$$

**Step 5: Selecting the Next Symbol**

Considering the preset number $N$ of test symbols (2),

$$S = \{S_1, S_2, \cdots, S_N\}$$

let us assume that the learner provided answer for the current symbol

$$S_n \quad \text{where } n \geq 1 \text{ and } n < N$$

The next test symbol

$$S_i \quad \text{where } i > n \text{ and } i \leq N$$

to be administered should be selected from the remaining

$$N - n$$

test symbols. To determine which one to present next, we calculate information gain for all the remaining symbols using (10)

$$IG(K \mid A_i) = H(K) - H(K \mid A_i)$$

In (10), $H(K)$ is the entropy of the most recent test symbol $S_n$ for which the learner provided an answer. It is obtained using (8). $P(K_j)$ are values $P(K_j \mid A_n)$ obtained with (7-b) if the answer to $S_n$ was correct, or with (7-d) if the answer was incorrect. If so, then (8) can be rewritten as
\[
H(K)_{s_n} = -\sum_{j=1}^{M} P(K_j \mid A_n) \log_2 P(K_j \mid A_n)
\]

Also, in (10), \(H(K \mid A_i)\) represents the conditional entropy of the next symbol \(S_i\) to be administered. Since there are two possible answers correct (true) or incorrect (false) to a given test symbol, the conditional entropy for \(S_i\) calculated with (9) will be

\[
H(K \mid A_i) = P(A_i = \text{true})H(K \mid A_i = \text{true}) + P(A_i = \text{false})H(K \mid A_i = \text{false})
\]

\(P(A_i)\) values are obtained as follows. According to the law of total probability

\[
P(A_i = \text{true}) = \sum_{j=1}^{M} P(K_j)P(A_i = \text{true} \mid K_j)
\]

and then

\[
P(A_i = \text{false}) = 1 - P(A_i = \text{true})
\]

The conditional entropies of the correct and incorrect answers are obtained using (8)

\[
H(K \mid A_i = \text{true}) = -\sum_{j=1}^{M} P(K_j \mid A_i = \text{true}) \log_2 P(K_j \mid A_i = \text{true})
\]

\[
H(K \mid A_i = \text{false}) = -\sum_{j=1}^{M} P(K_j \mid A_i = \text{false}) \log_2 P(K_j \mid A_i = \text{false})
\]

where the probabilities \(P(K_j \mid A_i = \text{true})\) with (7-a), while \(P(K_j \mid A_i = \text{false})\) are obtained with (7-a) and (7-c). The prior probabilities \(P(A_i \mid K_j)\) are taken from the model (database) where they are maintained from the previous test session. \(P(K_j)\) represents current values of classification states considering the answer to \(S_n\).

45
The symbol $S_i$ that provides the greatest information gain larger than 0 (negative result leads to a decrease in information gain) is the one to be administered next.

$$\max [IG(K \mid A_i)] \geq 0$$

(see Appendix D for an example).

Optionally, we may make a decision considering only conditional entropies. The symbol that has the conditional entropy with the smallest value, is to be selected next

$$\min H(K \mid A_i)$$

The lower the expected entropy, the more we know about the learner (Vomlel, 2005).

The application implemented for this project is to support learning of a syllabary through repetition and personalized testing. The user (i.e. a teacher or learner) presets a number of test symbols to be administered during a test session. Accordingly, a test is terminated after a preset number of symbols is answered.

Another approach where a test is ended when sufficient information about the learner is collected, after a smallest possible number of test items, is not considered. For more information about this approach, see Rudner (2001). Rudner “presents procedures for deciding when one has enough information to hazard a classification guess” (p. 15).

Shang et al. (2001) used a different approach for selecting the next topic. Their application calculates ‘learned score’ from ‘quiz performance, study performance, and reviewed topics’, and then based on the score sufficiency, allows learners the option of either letting the teaching agent to choose the next topic or selecting the next topic by themselves.
Learner Classification

One of the fundamental problems in machine learning, data analysis, and pattern recognition is the classification of observed instances into predetermined categories of classes (Bekele & Menzel, 2005). The fields of statistics and artificial intelligence have come up with a number of approaches including some trendy ones such as neural nets, decision trees, genetic algorithms, and those based on Bayesian inference.

Some standard statistical classification approaches of more interest for this project, as described by Rudner (2001), include Maximum-Likelihood Classification, Minimum Probability of Error Classification, Bayes Risk Criterion Classification, and Maximum A Posteriori Classification (Bayes Rule). In order to briefly define each in the context of this project, classification hypotheses will be declared. Since the goal is to classify the learner in one of the classification states (knowledge levels defined in (1)), the number of hypotheses will be equal to the number of classification states, which is $M$:

$$h_1, h_2, \ldots, h_M$$

The Maximum-Likelihood Classifier estimates the most likely knowledge level $K_j$ that generated the vector of answers $\tilde{A}$ in order to select hypothesis $h_j$. This approach may not always result in a good classification since it does not consider the prior data. The Minimum Probability of Error Classifier considers the vector $\tilde{A}$ to select a hypothesis ($h_j$) region that minimizes the total probability of error.
The *Bayes Risk Criterion Classifier* considers the vector $\tilde{A}$ and associates costs with each hypothesis in order to select a hypothesis $h_j$ to minimize the total expected cost.

Finally, the *Maximum A Posteriori Classifier (MAP)*, also called the Bayes Rule, selects hypothesis $h_j$ if $K_j$ is the most probable classification state considering the vector of answers $\tilde{A}$.

According to Rudner (2001), the Minimum Probability of Error, Bayes Risk Criterion, and Maximum A Posteriori (MAP) lead to identical classification results. Considering this, the Maximum A Posteriori Classification (MAP) is selected for the purpose of this project.

**Step 6: Maximum A Posteriori Classification (MAP)**

According to MAP, given the vector of observations (answers) $\tilde{A}$, hypothesis $h_j$ is selected if the $K_j$ is the most probable classification state. Considering formula (6-a)

$$P(K_j \mid \tilde{A}) = P(K_j)P(\tilde{A} \mid K_j) / P(\tilde{A})$$

MAP can be expressed as follows (Figure 8):

$$h_j = \max P(K_j \mid \tilde{A})$$

Accordingly, the learner has the knowledge level (classification state) that has the largest posterior probability considering the vector of the learner’s answers.
Different Views of the Learner Model

Some other views of the learner model are presented below including object-oriented, entity-relationship, and HCI views. While each view offers its own specific advantages, they all together provide a more complete picture of the learner model.

The object-oriented view shows a possible representation of the learner model in terms of its basic building blocks (classes) and their relationships. This view presents behavioral and data management responsibilities of each class, and thus how this responsibility is delegated across the class model (Bennett et al., 2001). The benefit of this view is that it offers three different perspectives when designing a system: conceptual, specification, and implementation (Fowler & Kendall, 2000).

The entity-relationship view represents a conceptual data model of the learner model. It depicts the learner model as a set of data entities and their relationships. The benefit of this
view is that it describes information needs of the learner model (the information that is to be stored in the database).

The HCI view shows interactions between the learner and the system. It describes low level actions that the learner is required to perform in order to accomplish a task higher in the hierarchy. This view facilitate understanding of the system including the way information circulates within it.

**Object-Oriented View**

Figure 9 shows the object-oriented depiction of the learner model. The profile of a learner consists of the learner’s personal data, a list of test sessions taken by the learner, and a set of probabilistic symbol statistics (one instance of statistics for each symbol).

The Learner class represents a learner with her/his personal data. Each learner is assigned a unique ID. Each instance of the Session class provides information about a test session taken by the learner including the session date and type (show or hide symbol), success rate, errors (symbols answered incorrectly), and symbol set used (full set, primary symbols, endings, symmetric or asymmetric symbols). The Statistics class describe statistics associated with a symbol class for a given learner. For each symbol, the statistics include the total number of correct guesses and the total number of attempts for all test sessions taken by the learner, and probabilities of a correct answer associated with each classification (knowledge) state. The Language, Element, Symbol, and Word classes represent the domain knowledge.
Figure 9: UML diagram of the learner model

**Entity-Relationship View**

The entity-relationship view of the learner model is shown in Figure 10. Please note that the ERD (Entity-Relationship Diagram) does not show Language and its elements (Symbols and Words). The language data is held in XML files as explained above under ‘Data Tier/Layer’. The relationships among language elements are depicted in Figure 5.
A learner can have only one profile. The profile includes the data of all test sessions taken by the learner. Also, it includes statistics for all symbols of the writing system of a language. Statistics of a symbol contain classification probabilities of a correct answer.

**HCI View**

To design an interactive system, it is necessary to determine potentials users of the system, their goals, and the actions they need to take to accomplish their goals. For the purpose of this application, it is assumed that a potential user (learner) is a non-expert willing to use the application as a tool for learning symbols of a writing system of a foreign

---

**Figure 10:** ERD of the learner model.
language. The learner knowledge is assumed to be functional, structured around a set of actions necessary to complete a task, and derived from previous experiences with similar tasks.

The considered learner’s knowledge, skills, and background are shown in Table 2:

**Table 2: A Potential User**

<table>
<thead>
<tr>
<th>Knowledge</th>
<th>Skills</th>
<th>Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum knowledge of personal computers; knows how to use input devices; familiar with OS functions; know how to run an application</td>
<td>Some typing skills; typing speed irrelevant</td>
<td>Some education; can read, write, learn, and reason</td>
</tr>
<tr>
<td>Familiar with most popular PC applications such as a text editor, media player, chat system, message board, etc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The user goals can be defined as follows:

- To practice the symbols of the writing system of the selected language (dialect).
- To test the current knowledge of the symbols of the writing system of the selected language (dialect).

The above model was used in the analysis of the user goals, corresponding actions for achieving the goals, and needs and wants (i.e. relevant information, references, etc.) while
working with the system. The intent was to establish possible ways for assisting the users while they are working with the system.

Hierarchical Task Analysis (HTA) chart in Figure 11 shows the actions that should be taken to practice.

![HTA Chart: Practice Session](image)

**Figure 11:** HTA Chart: Practice Session

A practice session starts when the learner displays a test symbol (option 1 in Figure 11). At this stage, the learner has multiple options 1.1 to 1.6 in Figure 11. The next step is to provide an answer (option 2 in Figure 11) to the currently displayed test symbol. This step is optional since the user may reveal the answer (option 1.3), skip the current test symbol and move to the next one (option 1.4), or reset the practice session before providing an answer.
(option 1.5). In a practice session, the learner is allowed to reenter her/his answer (option 2.1) until the correct one is provided (or the learner selects one of the remaining options 2.2 to 2.7 in Figure 11.

Figure 12 shows HTA chart for a test session. A test session is similar to a practice session, but the allowed options (shown in Figure 12) are significantly reduced (compare Figures 11 and 12). Providing an answer to the displayed test symbol is mandatory. The number of allowed attempts for a given symbol can be set to 1 or more. The learner cannot reveal the correct answer or reset the session.

**Figure 12: HTA Chart: Test Session**
**Presentation Control Component**

The presentation control (or dialog) component enables communications between the learner and the system. It denotes allowed actions, action sequences, and interface responses during a session. The presentation controls for practice and adaptive testing are based on the learner-task model introduced in ‘HCI View’; Figure 11 and Figure 12 illustrate the learner actions and their sequences. Figure 13 shows the interface through which the personalized instruction is delivered and through which the learner issues the actions and the system delivers responses. The presentation (response) panels of this component display the elements of the domain knowledge. The left side displays a Symbol, while on the right side is the corresponding example (Word) that uses the symbol.

**Figure 13: Presentation Control Interface**

The action ‘Next’ displays a new symbol and its example, while ‘Show’ discloses the answer. ‘Reset’ resets the session parameters to their initial values and starts from the
beginning. ‘Record’ invokes the audio recorder so that the learner can record (and listen) her/his pronunciation.

**Presentation Views of the Learner Model**

A prospective approach for responding to the needs of a learner consists of opening the contents of the model to the learner. The learner is typically provided with a choice of different views of her/his knowledge described with the model. The model data may be conveyed in the form of a chart, plain text, graphics, or some other form based on uniqueness of the learner and the purpose for the modeling (Hansen & McCalla, 2003). Different model views allow the learner to reflect on the properties of her/his model. As a result, the learner achieves better understanding of the knowledge domain, her/his present beliefs, and performances (Bull & Nghiem, 2002). If the views also allow interactions, the learner may supply further and/or perhaps update existing information and so positively influence the accuracy and utility of the model.

According to Hansen & McCalla (2003), the learner model can also be opened up to peers and teachers. In this case, a learner can judge her/his model against models of other learners. If so, the learner may determine issues encountered by other learners, compare her/his progress with others, locate the most suitable peer who may offer help, and so on. On the other hand, a peer that offered help may determine how to help by examining the model of the learner seeking help.
Mabbot & Bull (2004) conducted a study to determine benefits of a choice of views showing information in the learner model. The following are some study conclusions:

- While learners found it useful, they showed different preferences to different views or representations of their knowledge. The solution is to offer multiple views so that a learner can select the preferred one.

- No link is established between the learner preferences and her/his learning style; “so intelligent adaptation of presentation to learning style does not seem beneficial” (Summary, para. 2).

- Color-based representation of the learner knowledge levels of study topics proved effective.

- Students liked the idea of comparing models with others.

Some example applications that demonstrate a variety of interaction and presentation styles include:

- STyLE-OLM (Scientific Terminology Learning Environment - Open Learner Model) uses conceptual graphs and dialogue games to involve the learner in interactive open learner modeling (Dimitrova, 2003).

- Flexi-OLM offers the user a choice of seven views of their learner model contents: alphabetical index, list ranked according to knowledge, concept map, hierarchical structure grouping related concepts, pre-requisites structure, lecture structure, and textual summary (Mabbott & Bull, 2004).
- SQL –Tutor opens the student models as skill-meters. Evaluation studies show that even such simple open student models support students in self-assessment and reflection (Mitrovic & Martin, 2002).

- ViSMod (Visualization of Bayesian Student Model) helps students and their teachers to interact with Bayesian student models. Bayesian student models are maps containing information about cognitive and social aspects of the student. These maps can be explored and annotated by students and teachers during the learning process (Zapata-Rivera & Greer, 2004).

A possible weakness of current applications is that while they offer a choice of presentation styles, each tends to remain with a constant interaction method (Mabbot & Bull, 2004). Furthermore, some applications showed tendency of providing model views with overwhelming information (Zapata-Rivera & Greer, 2000). To overcome this, Hansen & McCalla (2003) suggested partial views based on specific purpose rather than having a single large view.

**Presentation Views of the Learner Model in Syllabic Tutor**

The application implemented for this project offers two groups of different representations of a learner’s knowledge: symbol view and test session view. These two view groups provide two different perspectives to the contents of a learner model.
The symbol view group describes the learner knowledge related to individual test symbols of a syllabary. This view is available in three flavors (Figures 25, 26, and 27 in Appendix A):

- Pedagogical View shows syllabics (accompanied with related learner knowledge statistics) in its natural layout (Figure 1).
- Table View is a kind of a spreadsheet table with a list of symbols and corresponding statistics.
- Bar Chart View is a type of graph in which the learner knowledge of each test symbol is represented by rectangular bars.

While the view choices have different look and feel, they basically convey the same information. For a specific test symbol, each view shows the learner success rate in percentages, and the correct/incorrect answer proportion. The correct component of the proportion is presented in blue, while incorrect is in red. For each test symbol, each view also provides a link to additional information (Figure 28 in Appendix A), which shows a pie chart with correct (blue)/incorrect (red) ratio with the counts of total attempts and numbers of correct and incorrect answers, and the probability of the correct answer.

The test session view group describes the learner performances in adaptive testing. This group comes in two forms, as a table view and a bar chart view (Figures 29 in Appendix A). In appearance, they are identical to the table and bar chart views of a symbol, but they embody different information. Each view shows the date of the test and the learner score as a percentage. Each test session provides a link to more information, including a pie chart with
the session details such as total symbols used in the test, the count of correct and incorrect
to answers, the symbol set used, and the list of test symbols that were incorrectly answered.

Feedback Component

It is a common belief that individuals exhibit diverse preferences and approaches
toward learning. While some individuals may favor learning through visual imagery, others
may prefer learning through auditory activities or by doing practical tasks etc. This individual
inclination toward a specific learning approach can be phrased as the individual’s learning
style.

A huge number of learning style models has been proposed. According to Wikipedia
(2006d), there are over 70 learning style models. Some major models include Allinson &
Hayes’ Cognitive Style Index (CSI) (1996), Apter’s Motivational Style Profile (MSP)
(1998), Kolb’s Learning Style Inventory (LSI) (1976), Dunn & Dunn model and instruments
of learning styles (1984), Gregorc’s Mind Styles Delineator (MSD) (1982), and so on. The
current research in the area of learning styles seems to have limited agreement on structure or
taxonomy of learning styles. Thus, different theories tend to appear in different terminology.
Some of the simplest and perhaps the most common taxonomies are Lockitt’s (1997) Visual-
Auditory-Kinesthetic (VAK), and Fleming’s (1987) Visual-Aural-Read/Write-Kinesthetic
(VARK). Compared to VARK, VAK appears to omit the third classification state of VARK.
Otherwise they appear to belong to the same taxonomy family articulated as follows:

1. Visual Learning refers to learning by seeing (visual imagery i.e. images, videos,
diagrams, charts, etc.).
2. **Auditory or Aural Learning** refers to learning by hearing (using words and sounds for learning i.e. lectures, storytelling, music, etc.).

3. **Reading/Writing Learning** refers to learning by processing text.

4. **Kinesthetic Learning** refers to learning by doing practical things (i.e. role playing, clay modeling, and acting).

To determine a preferred learning style of an individual, learning style instruments or tools such as questionnaires, surveys, tests, and inventories are used. Such instruments are typically based on one of the established models such as those mentioned above. Many tools are freely available online, while those offered by their originators such as VARK (Fleming, 1987) and Gregorc’s Mind Styles Delineator (MSD) (1982) require licensing fee.

Learning Style theories and instruments have received considerable criticism from other researchers. In general, researchers such as Curry (1990) raised issues concerning the lack of evidence and application of questionable scientific background and theories for learning style models. A recent investigation (Coffield et al., 2004) expressed reservations towards a number of most widely used learning style models due to analogous reasons. Considering pedagogic implications and a lack of clear methodology for dependably deriving the appropriate pedagogic strategies, Melis (2004) also displayed skepticism regarding the viability and validity of using learning style to adapt or personalize a learning environment to suit the needs of the learner. Similar to Melis, Robotham (1999) also questioned the validity of such a ‘flawed’ approach due to a narrow range of classification states, containing a limited number of learning activities.
Without further empirical evidence or validation of learning style theories, tailoring learning strategies and providing instructions to suit certain learning style should be re-evaluated (Melis, 2004; Robotham, 1999). Instead, Robotham proposed a move toward self-directed learning. Palloff & Pratt (2001) suggested production of a course with an assortment of approaches. Likewise, Melis (2004) proposed delivery of instruction in different formats such as visual, audio, textual, symbolic, etc.

Therefore, in order to respond to the learner’s needs, an adaptive learning environment is to deliver learning material in various presentations, while the learner is the one who chooses which direction to follow (i.e. a particular or a combination of learning styles). Considering this, the application of this project delivers instructions in diverse formats, including:

- **Symbolic Presentation** in which syllabic symbols are individually delivered to the learner, one at a time (see Figure 13). Each symbol is also accompanied with an example.

- **Aural Presentation only** delivers the audio pronunciation of a symbol to the learner, one audio representation at a time, accompanied by the audio of an example.

- **Symbolic-Aural Presentation** represents the combination of the above two.

- **Graphical Presentation** delivers syllabics organized in a pedagogical chart (see Figure 1). A number of charts are available, including a complete chart, a chart emphasizing only primary symbols, endings, symmetric or asymmetric symbols.
- *HTML Presentation* represents a tutorial in HTML format consisting of text with colours to distinguish relevant details and a number of images showing syllabic symbols in different layouts giving different properties of syllabics.
CHAPTER IV

RESULTS

Specification of Software Capabilities

Figure 14 shows the components of the system developed in this research and their relationships.

Figure 14: System Components
**Syllabic Learner**

Syllabic Learner consists of three components: Syllabic Tutor, Simple Text Editor, and Guessing Game.

**Syllabic Tutor**

This is the main component of this system and is for learning the syllabics. The interface of this component is shown in Figure 13. Before starting a session, the syllabic tutor allows the learner to choose the session type (learn or test), the presentation (display) option (Symbol or Sound), and the set of syllabics (the entire set, primary syllabics or finals, symmetric or asymmetric symbols) (Figure 15). If a test session is selected, the learner should also specify the number of test symbols to be presented during the session and the maximum number of allowed attempts for each symbol (1, 2 or 3).

![Figure 15: Session Options](image-url)
When the session starts, the tutor displays a syllabic (visual or audio, depending on the selected presentation type), accepts the learner response, and provides immediate feedback (i.e. correct/incorrect) (Figure 16).

![Figure 16: Immediate Feedback](image)

This process continues until all test symbols have been displayed and responded. After the session (test mode only), the tutor provides more comprehensive feedback informing the learner about the session type, the symbol set used, the number of correct and incorrect answers, the session mark, and the list of wrongly guessed symbols (Figure 17).
Figure 17: Feedback after a test session
**Simple Text Editor**

The editor (Figure 18) allows learners to enter individual syllabics, words, or more complex syllabic content depending on the second-language skills of the learner. The editor allows creating and saving files so that the learners can exchange syllabic files among themselves. Files created with the simple editor can also be edited with MS Word and Notepad. To do so, it is necessary to apply fonts capable of displaying Cree syllabics to the file content within MS Word (or Notepad).

![Simple Text Editor](image)

**Figure 18:** Simple Text Editor

**Syllabic Guessing Game**

It is a variation on a common game (Figure 19). It is a board consisting of squares under which randomly distributed syllabics are concealed. Each syllabic occurs twice under the squares at different positions. The learner reveals a syllabic at a time by clicking on a
square. The learner must remember the revealed syllabic and the location of its square. The learner tries other squares until a match is found. The learner must then click on the original square to confirm the match. The matched syllabics remain revealed until the end of the game.

![Figure 19: Guessing Game](image)

**Syllabic Input Method**

The syllabic input is provided through the visual keyboard (Figure 20). The position of syllabics on the keyboard follows the pedagogical layout shown in Figure 1. Each keyboard key shows one of the syllabics.
The message in red (i.e. SERVING: Syllabics Tutor) at the bottom of the keyboard informs the user which window is currently accepting the input from the keyboard. To perform input to a different window such as Simple Text Editor, user should click on the respective keyboard button on the toolbar of the editor window (Figure 18). The message at the bottom of the keyboard will change to SERVING: Simple Text Editor.

The syllabic input method can also be provided through the system keyboard. The system syllabic keyboard is not part of this project, but was implemented for a Comp648 project as part of author’s studies at Athabasca University. Basically, the learner enters a combination of keys from the standard Latin keyboard to produce a single Cree symbol. The
system keyboard is an independent component that is installed as an extension to Java Runtime Environment. It supports Cree input for any Java software with text input components.

Communication Module

The communication module consists of two components, a simple chat system and a conference board.

Simple Chat System

![Simple Chat System](image)

**Figure 21**: Simple Chat System

The simple chat system is shown in Figure 21. The chat system allows learners to communicate using syllabics. In addition, learners can also communicate in English or any
other language based on Latin alphabet. The way the chat system works is rather straightforward. The learner first enters a chat session by clicking the ‘Connect’ button. The list of current users is displayed on the right side of the chat window. Next, the learner simply enters some text in the input field and presses the Enter key.

**Simple Conference Board**

The conference board (Figure 22) allows learners to post message written in a syllabic writing system. Similarly to the chat system, the conference board also allows postings written in English or other language based on similar alphabet.

![Simple Conference Board](image)

**Figure 22:** Simple Conference Board

The functionality of the conference board is also simple and follows the functional approach of standard conference boards and forums. ‘Open’ displays the currently selected message content in a separate window; ‘Reply’ posts a reply message to the currently selected
message; ‘Post Message’ posts a new message to the current topic thread (folder); and ‘Start New Topic’ creates a new topic thread.

**Video Tool**

It (Figure 23) is a tool for playing video (and audio) materials. It can be used for playing various syllabic, language, and culture related media materials.

![Simple Media Player](image)

**Figure 23: Simple Media Player**

The box ‘Select Files’ provides a list of available media files posted on the server by the administrator (or teacher). The button ‘Open’ provides means for playing local media files. The video tool is based on Java Media Framework (JMF), which provides support for limited number of video formats, mainly those uncompressed and inappropriate for the Web use due
to their file sizes. Most popular (compressed) video formats such as MP4 and MOV are not supported since they require licensing.

**System Testing**

According to Schach (2002), behavioral properties of the software system that must be tested include utility, reliability, robustness, performance, and correctness.

**Utility**

Utility refers to the ability of the system or its module to perform a task or tasks and/or the level to which the system functions correspond to the learner needs when the system is used as intended. This may involve testing the ease of the system use, whether it provides the intended functionality, whether a specific function is practical, and so on.

Due to the limited project resources, the participants were selected among friends and acquaintances. Of those contacted six responded and provided answers to the questionnaire. The copy of the questionnaire is provided in Appendix B. The educational background and computer experience of participants are presented in Table 3. Tables 4, 5 and 6 summarize the participant responses (SA - strongly agree; A - agree; EU – extremely useful; VU – very useful; NA – undecided/cannot say).
### Table 3: Utility Testing: Participant Profile

<table>
<thead>
<tr>
<th>Age Range</th>
<th>30-40</th>
</tr>
</thead>
</table>
| **Education** | High School: 1 participant  
College Diploma: 2 participants  
Applied Degree: 2 participants  
University Degree: 1 participant |
| **Computer and Internet Experience:** | All participants feel confident about using a computer. All participants consider themselves skilled with tasks such as Web browsing, file downloading, multimedia playing, electronic communications, game playing, MS Word document processing, and so on. |
| **Information System Experience:** | Advanced experience – 1 participant  
Some experience – 1 participants  
Little or No experience – 4 participants |

### Table 4: Utility Testing Results - Syllabic Learner

<table>
<thead>
<tr>
<th>Module</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q1</td>
</tr>
<tr>
<td>Syllabic Learner</td>
<td>SA = 100%</td>
</tr>
<tr>
<td><strong>Question Q5</strong></td>
<td>I strongly recommend this software for the future use.</td>
</tr>
<tr>
<td><strong>Comments:</strong></td>
<td>Overall Syllabics Tutor with its modules (components) is simple, practical, and functional tool for learning Cree syllabics.</td>
</tr>
</tbody>
</table>

### Table 5: Utility Testing Results - Communication Module

<table>
<thead>
<tr>
<th>Module</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q6</td>
</tr>
<tr>
<td>Communication Module</td>
<td>SA = 83%</td>
</tr>
<tr>
<td></td>
<td>A = 17%</td>
</tr>
<tr>
<td><strong>Question Q10</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td></td>
</tr>
</tbody>
</table>
**Table 6:** Utility Testing Results - Video Tool (Media Player)

<table>
<thead>
<tr>
<th>Module</th>
<th>Question</th>
<th>Q11</th>
<th>Q12</th>
<th>Q13</th>
<th>Q14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media Player</td>
<td>Q11</td>
<td>SA = 50%</td>
<td>EU = 33%</td>
<td>SA = 67%</td>
<td>SA = 67%</td>
</tr>
<tr>
<td></td>
<td>Q12</td>
<td>A = 50%</td>
<td>VU = 50%</td>
<td>A = 33%</td>
<td>A = 33%</td>
</tr>
<tr>
<td></td>
<td>Q13</td>
<td>NA = 17%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Question Q15</td>
<td>Comments</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Considering the received feedback, all the respondents either ‘Agreed’ or ‘Strongly Agreed’ with the questions whether the system modules provide the intended functionality and with their usefulness and practicality. One respondent remained undecided regarding the usefulness of the communication module. Further contacts with the participant revealed that the participant somehow missed the question and did not provide a response so that the Web form applied its default answer NA (undecided; cannot say). Another respondent remained undecided regarding the usefulness of the multimedia player concerning the Cree content. While the participant thinks highly about the module appearance and ease of use, the participant seems unsure about usefulness of the module due to the limited number of supported media formats. The supported formats are described in the subsection ‘Video Tool’.

**Reliability**

The purpose of reliability is to determine the occurrence rate of and consequences caused by the system failure. In other words, it determines how often the system fails and how long it takes to recover it. The system achieved its final deployment for the public use.
during January 2006. Considering the time of writing this, the system has been in operation for about 6 months without any maintenance. The testing participants did not report any problems with the system. Other users who accessed the system just to see/examine it did not report any failure problems.

During December 2005, before the final deployment, there were some problems with the Java Servlet Engine (Tomcat) on the AU student server ‘io.acad.athabascau.ca’. According to the SCIS programmer/analyst, the engine failure occurred about once every week or two during December 2005. The failure of the servlet engine caused the system to stop working since it could no longer access its servlets and data. The resolution of the problem was quick and consisted of restarting the servlet engine. It appears that the SCIS programmer/analyst resolved this problem given that no Tomcat failure has been reported since.

Robustness

According to Schach (2002), testing robustness means determining whether the system behavior is satisfactory under different valid and invalid operational conditions. For example, a robust system should not produce improper results if the valid input has been provided and should not produce undesirable consequences if the invalid input has been entered.

The current system provides almost no opportunity for the learner to enter a wrong input. Most interaction components are implemented as buttons or selection lists or boxes. So
the learner can only click on or select one of available options. In cases where the user’s
direct input is required (i.e. input text fields), the error handling is applied. If invalid input is
provided, an error or warning dialog is displayed informing the user about the problem
and/or a possible subsequent action. In such a case, the system continues with no
consequences.

The system modules such as simple text chat system and conference board simply
echo the user input. The user is allowed to enter any input. Similarly, the simple text editor
also allows any input. These modules operate on the principle ‘What you enter is what you
get.’ In other words, they do not validate or process the user input (i.e. grammar, spelling,
etc.).

The syllabic tutor requires the user input. It compares the user input with the system
data. Although any entry is allowed, no malfunction will happen, since the system will
simply respond ‘Correct’ or ‘Incorrect’.

Performance

Performance testing determines whether the system storage and memory
requirements and response times are acceptable.
Storage Requirements

The client application is compressed into a single JAR file for delivery via Java Web-Start. The file size is 293 kb. The same JAR file can be used to embed the application as an applet in a HTML page.

The server size consists of a number of Java servlets and supporting classes. The total size of all the server classes is around 96 kb.

The application data consists of audio files, XML data, and database records. Each language element (a symbol or an example word) is accompanied with an audio file. The average total size of all audio files for a language is around 660 kb.

XML data includes Unicode representation of language elements, their Latin equivalents, and other attributes relevant for the application. The typical size of all XML files for a language is around 21 kb.

A database maintains various records for each learner. The MySQL Control Center is used to determine the size of records. Typical size of the learner model data is around 18 kb. This type of data usually does not grow; the data is only updated so that its size is maintained. The data that grows include test sessions and conference board postings. Considering the current example test sessions and example postings, a typical test session data consumes around 9 kb and an average posting takes around 5 kb. But the actual size of a posting will depend on the length of the entered text.
Considering the above presented values, it is worth to note that the system storage requirements are expressed in kb, while current computer configurations express their storage capacities in GB. So the storage requirements of the system appear minimal.

The above does not include storage requirements necessary for Java Virtual Machine, Java Media Framework, and fonts.

**Memory Requirements**

To determine the memory requirements, a simple Windows process monitor, Windows Task Manager, is used. Table 7 shows estimated values:

<table>
<thead>
<tr>
<th></th>
<th>Memory Usage [MB]</th>
<th>Peak Memory Usage [MB]</th>
<th>Virtual Memory Size [MB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Load</td>
<td>26</td>
<td>28</td>
<td>24</td>
</tr>
<tr>
<td>After Some Use</td>
<td>41.5</td>
<td>42</td>
<td>38.5</td>
</tr>
<tr>
<td>Required Minimum</td>
<td>7</td>
<td>42</td>
<td>32</td>
</tr>
</tbody>
</table>

*Memory Usage*, also called the working set or resident set of the system, represents the set of equal-sized memory units (page frames) currently in main memory. This value may not be exact since it may include memory parts shared with other processes running on the same computer system.

*Peak Memory Usage* shows the highest value of *Memory Usage* by the system.

*Virtual Memory Size* represents the total private virtual memory allocated by the system.
The first row of the table shows the values when the application is first loaded. The second row shows the values after some extensive work with the system. The third row shows the minimum amount of memory required by the system. The minimum amount of the required memory is determined by applying the Microsoft utility ClearMem as suggested by Shepherd et al. (2004).

The configurations of modern, personal computers include RAM measured in hundreds of MB. More recent computers express RAM in GB. The memory requirements of the implemented system appear satisfactory.

**System Response Time**

The following system was used to determine response times of some tasks:

Internet Connection: ADSL
Processor: AMD Athlon 64 3000+ (1.8 GHz)
RAM: 512 MB

The response time of a task is determined by measuring the start and end times of the task, and then subtracting the start time from the end time. The code that notes start and end times was inserted around the code of a task as follows:

```java
long startTime = System.currentTimeMillis();
-Begin the task
-End the task
long endTime = System.currentTimeMillis();
long responseTime = endTime - startTime;
System.out.println( "Response Time: " + responseTime );
```
Only remote tasks (those that make connections to the remote server) were tested. Local tasks (no connections to the remote server) are much faster and comparable to those of any other application that executes locally. Hence local task were not tested.

The response times of the following remote tasks were tested:

1. The login process
2. The conference board (downloading and displaying all the conference posts)
3. Posting a message on the conference board
4. The learner profile (downloading and displaying the data)
5. The simple text chat system

1. The response time of the login process

The most time-consuming task of the system is the login process. This task involves:

- validating the user ID and password
- retrieving the user data from the database
- parsing language data from XML files
- converting audio files to byte arrays
- generating serialized objects for the User, and Language and its elements
- returning the objects to the client application

<table>
<thead>
<tr>
<th>Task: Log In</th>
<th>Data Received [kb]</th>
<th>Response Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trial #</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>610</td>
<td>4797</td>
</tr>
<tr>
<td>2</td>
<td>610</td>
<td>4766</td>
</tr>
<tr>
<td>3</td>
<td>610</td>
<td>4688</td>
</tr>
</tbody>
</table>

According to the obtained data (Table 8), this task takes close to 5 seconds to complete. While on the first thought the task appears rather long, it is the satisfactory
alternative to the option of close to hundred remote connections, each lasting under or around 1 second.

The login task downloads all the language elements (symbols and examples) and their audio files in a single connection to the server. The advantage is that the work with the syllabic tutor modules appears local since no additional connections to the servers are made. It is also considered that the learner logs in only once per session, while the learner may wish to use other modules multiple times within the same session. So the small sacrifice in the login process is rather a huge gain for the remaining parts of the system.

The other, considered option (during the design process) was to access language elements and their audio files on as needed basis. In other words, when a symbol and its example are to be displayed to the learner, the system would make a connection to the remote server and download the required data (a symbol, an example, and two audio files). If we consider Plains Cree, which has 84 symbols, the system would have to make at least 84 connections (one for each symbol) to the remote server and download around 7.5 kb of data per connection. This option would reduce the response time per connection, but would increase the overall system connection time to the server. Also, tasks of the syllabic learner would no longer be local; they would involve connections to the server and so become slower. Consequently, the entire performance of the system would be affected.
2. The response time of the conference board

When the conference board window is opened, the system makes a connection to the server, and retrieves and downloads all the conference postings. The response times are shown in Table 9.

<table>
<thead>
<tr>
<th>Trial</th>
<th>Board Data [kb]</th>
<th>Response Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>157</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>266</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>250</td>
</tr>
</tbody>
</table>

All response times are around 0.25 seconds, which appear satisfactory. However, as the number of postings increases, the actual response time is expected to increase.

3. The response time of posting a message to the conference board

Adding a post to the conference board involves inserting the message into the database and then reloading the conference message tree. The following are some response times given the message size:

<table>
<thead>
<tr>
<th>Trial</th>
<th>Message Size [kb]</th>
<th>Response Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>218</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>296</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>297</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>250</td>
</tr>
</tbody>
</table>

The response times (Table 10) appear in between 0.2 and 0.3 seconds.
The response time of a posting may also increase as the conference message tree increases in its size since the updating process may take a bit longer.

4. The response time of the learner profile window

The learner profile window connects to the server and downloads all the data related to test session taken by the learner, the learner model, and individual symbols.

**Table 11: Response Times of the Learner Profile Window**

<table>
<thead>
<tr>
<th>Trial#</th>
<th>Response Times [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>94</td>
</tr>
<tr>
<td>2</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>78</td>
</tr>
</tbody>
</table>

At the time of testing this, there were around 10 completed test sessions and 84 symbols and related data. The response times are near to 0.1 second (Table 11). The response time may increase as the number of test sessions taken by the learner increases.

5. The response time of the simple chat system

The learner types a short message and presses the Enter key. The response of the text chat system appears instant.
Correctness

The system will be correct, if input satisfies the input specifications, the product is given all the resources it needs, and the resulting output satisfies the output specifications (Schach, 2002).

The correctness of the system is tested through a number of test sessions. During a test session, the pedagogical chart showing all the syllables (symbols) of the language is used as a reference. For each displayed symbol, the reference chart is consulted and the correct answer is entered. The system always responded correct. Next, during another test session, for each displayed symbol an incorrect answer is entered. The system responded ‘Incorrect’ as anticipated. The remaining sessions were used to randomly provide correct and incorrect answers for displayed symbols; each time the system responded as expected.

The simple text editor is tested by entering some text, saving the entered data in a file, and reloading the file. The process was repeated several times with positive results.

The conference board is tested by creating new discussion topics, posting new messages, and replaying to the existing messages. The results were as expected.

To test the simple chat system, a chat session was established with multiple users. No unanticipated results were reported.

Finally, the guessing game was tested. The game always performed correctly.
Success Criteria and System Evaluation

The system was evaluated according to the following success criteria: reusability, stability, performances, and platform independence.

Reusability

The system is reusable for other Cree dialects and even languages with similar syllabics writing scripts. To test this, two additional Cree dialects (Woods Cree and Northern Plains/Woods Cree) were added in addition to the original one (Plains Cree). Before the learner logs in the system, the learner selects a language of interest.

![Language Selection](image)

**Figure 24**: Language Selection

The following steps explain how to add a new dialect/language to the system:
Step 1:

Add a new entry to the language XML file. The entry includes the dialect/language name, the name of fonts capable of displaying the language symbols, and the name of a folder where the language data is to be stored. An example of the entry looks as follows:

```xml
<LANGUAGE name='Plains Cree' font='Aboriginal Serif Unicode' folder='plainscree'/>
```

Step 2:

Create a new XML file that will include all the symbols of the language specified above in Step 1. A sample XML file consisting of two symbols may look as follows:

```xml
<?xml version='1.0' encoding='ISO-8859-1' ?>
<SYMBOLS language='Plains Cree'>
  <SYMBOL row='1' char='140a' roman='a' audio='a' type='primary' rotation='symmetric'/>
  <SYMBOL row='1' char='1404' roman='ii' audio='ii' type='primary' rotation='na'/>
</SYMBOLS>
```

This file should be saved in the folder specified in the language entry in Step 1. The attribute ‘row’ refers to the row in which the symbol appears on the pedagogical chart or table. If the new language does not provide a chart, the user may order symbols in any desired way. The column of a symbol is determined by the order of symbols in the XML file. The first symbol is in column 1 and the second symbol is in column 2. The attribute ‘char’ represents the Unicode of the symbol, which is to be used to generate the symbol. The attribute ‘roman’ represents the Latin (Roman) equivalent of the symbol. The attribute ‘audio’ specifies the name of the audio file associated with the symbol. It may be equal to ‘roman’. The attribute ‘type’ refers to the symbol type. For example, Cree symbols can be classified as primary and special symbols (or endings). The attribute ‘rotation’ refers to the rotation of a symbol. For
Step 3:

Create a new XML file that will include an example (word) for each symbol in the XML file created in Step 2. A sample file consisting of two examples may look as follows:

```
<?xml version="1.0" ?>
<LANGUAGE name="Plains Cree">
  <EXAMPLE char='>' word='oβ' roman='neevo' english='four' audio='neevo'/>
  <EXAMPLE char='Δ' word='Δ<Δ' roman='wiiwa' english='wife' audio='wiiwa'/>
</LANGUAGE>
```

The file should also be saved in the folder specified in Step 1. The attribute ‘char’ represents the symbol from XML file created in Step 2 to which this example belongs. The attribute ‘word’ represents the actual word written in its native language. The attribute ‘english’ represent the translation of the word in English. The remaining attributes, ‘roman’ and ‘audio’, have the same meaning as explained in Step 2.

### Stability

This refers to the reliability of the system discussed above under the System Testing. The stability of the system is confirmed through its reliability testing (see System Testing, Reliability).
**Performance**

It is anticipated that the system will have reasonable response times. This was verified through the performances testing (See System Testing, Performances).

**Platform Independency**

The system is implemented in Java, a platform independent programming language. It does not have any ties to any specific platform. The system is successfully executed on two platforms, MS Windows XP and Linux Red Hat 8.0.
CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

The original prototype was developed considering the feedback received from Cree teachers (Holt et al., 2004). The current adaptive learning system is based on the functionality and user interface of the original, standalone prototype. The development of an adaptive learning system is a complex, time-consuming and expensive undertaking. As a result, the implemented system is quite elementary. However, it is highly anticipated that the system will prove useful at least as a base for further research and development.

At the technical level, the system appears relatively successful. It achieves stability, reasonable performances, and reusability for languages and dialects based on syllabary. The system also received quite positive feedback from those who have had the opportunity to see and experiment with it.

The implemented system appears to have some prospects in the project of Dr. Peter Holt (Holt & Jelica, 2006). Dr. Holt’s project is to conduct a usability study of the learning system implemented for this thesis project. Dr. Holt is presently on a leave of absence so that the field-testing is delayed. Positive results of the prospective field test may perhaps initiate further research and development.
Subsequent versions of the system can improve existing features and implement some new ones. Some of these improvements and additions may include:

- The installation process on the client side appears rather complex. Alternative technologies may be explored for the implementation of the client. For example, Macromedia Flash appears a suitable alternative to Java clients. Compared to Java Virtual Machine and Java Media Framework, Flash plug-in is more commonly included with current Web browsers. This option may perhaps completely eliminate the installation process on the client side. A possible drawback is that Macromedia Flash development environment requires licensing.

- The current system teaches only syllables. The next step would be to teach some simple syllabic words. For example, a word may be supplemented with a picture of object that it describes and an audio file that demonstrates the word pronunciation. In addition, a simple dictionary with search capabilities may also be added in order to provide quick access to multimedia-supplemented words.

- Further, the system may be enabled to teach more complex language constructs such as sentences. This involves the knowledge of the language grammar. So an external language expert would likely be required.

- Also, the communications module can be extended by adding audio and video capabilities for the chat system. The conference board can be extended to support HTML and images.

Addition of new learning features (i.e. words, sentences and grammar) imposes new challenges in the learner modeling and modeling of the instructions sequence generation. It is
likely that the models would have to be reevaluated and adapted to handle new kinds of instructions. The current models consider all questions (symbols) at the same difficulty level. Adding words and sentences would likely necessitate the categorization of different instructions according to different difficulty levels. For instance, instructions related to syllabics may be considered at ‘Newcomer’ or ‘Beginner’ instruction difficulty level, syllabic words at ‘Intermediate’ level, and syllabic sentences and grammar at ‘Senior’ and ‘Expert’ level. In view of that, the opportunities for further research, improvements and new features appear quite unlimited.
REFERENCES


97


APPENDIX A:

A Choice of Learner Model Views

Figure 25: Bar Chart View

Figure 26: Pedagogical Layout
Figure 27: Table View

Figure 28: Additional Information:
Figure 29: Bar Chart and List of Taken Test Sessions
APPENDIX B:

Utility Testing Questionnaire

1.) Syllabics Tutor & Utilities Module

Q1: This software module achieves its primary task (facilitate learning of Cree syllabics).

Strongly Agree Somewhat Agree Somewhat Disagree Extremely Disagree Undecided / Cannot Say

Q2: The functions of this module are useful for learning Cree syllabics.

Extremely Very Quite Somewhat Not at all Undecided / Cannot Say

Q3: This software module is easy and/or practical to use.

Strongly Agree Somewhat Agree Somewhat Disagree Extremely Disagree Undecided / Cannot Say

Q4: Overall, I am satisfied with this software module. Its functions correspond to my learning needs.

Strongly Agree Somewhat Agree Somewhat Disagree Extremely Disagree Undecided / Cannot Say

Q5: Comments:

2.) Communications Module

Q6: This software module achieves its primary task (facilitate communications among learners using syllabics or Roman alphabet).

Strongly Agree Somewhat Agree Somewhat Disagree Extremely Disagree Undecided / Cannot Say

Q7: The functions of this module are useful for exchanging and/or posting messages composed of Cree syllabics (or letters of Roman alphabet).

Extremely Very Quite Somewhat Not at all Undecided / Cannot Say

Q8: This software module is easy and/or practical to use.
Q9: Overall, I am satisfied with this software module. Its functions correspond to my communication needs with other learners of Cree syllabics.

Q10: Comments:

3.) Multimedia Module (Video Tool)

Q11: This software module achieves its primary task (playback of video and audio files).

Q12: The functions of this module are useful for viewing multimedia content relevant for learning syllabics.

Q13: This software module is easy and/or practical to use.

Q14: Overall, I am satisfied with this software module as a visual supplement to the Syllabics Tutor.

Q15: Comments:
APPENDIX C:

Updating the Probabilistic Model

Initial Data
Considering the initial data assumed in the subsection “Initializing the Model”,

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P(g)$</td>
<td>$0.150$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$P(s)$</td>
<td>$0.100$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$P(K_j)$ where $j=1,2,…,6$</td>
<td>$0.167$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

the initial model will look as follows:

| Classification | $P(A_i|K_1)$ | $P(A_i|K_2)$ | $P(A_i|K_3)$ | $P(A_i|K_4)$ | $P(A_i|K_5)$ | $P(A_i|K_6)$ |
|----------------|--------------|--------------|--------------|--------------|--------------|--------------|
| $A_i = \text{true}$ | $0.15$ | $0.3$ | $0.45$ | $0.6$ | $0.75$ | $0.9$ |

Test Session 1:
The first test assumes 6 test items. After each answer, the model is updated (see details in the subsection “Updating the Model”). After the first answer, the model data will look as follows:

| Classification | $P(A_i|K_1)$ | $P(A_i|K_2)$ | $P(A_i|K_3)$ | $P(A_i|K_4)$ | $P(A_i|K_5)$ | $P(A_i|K_6)$ |
|----------------|--------------|--------------|--------------|--------------|--------------|--------------|
| $A_1 = \text{False}$ | $0.298246$ | $0.245614$ | $0.192982$ | $0.140351$ | $0.087719$ | $0.035088$ |
| $A_2 = ?$ | $0.15$ | $0.3$ | $0.45$ | $0.6$ | $0.75$ | $0.9$ |
| $A_3 = ?$ | $0.15$ | $0.3$ | $0.45$ | $0.6$ | $0.75$ | $0.9$ |
| $A_4 = ?$ | $0.15$ | $0.3$ | $0.45$ | $0.6$ | $0.75$ | $0.9$ |
| $A_5 = ?$ | $0.15$ | $0.3$ | $0.45$ | $0.6$ | $0.75$ | $0.9$ |
| $A_6 = ?$ | $0.15$ | $0.3$ | $0.45$ | $0.6$ | $0.75$ | $0.9$ |

After the second, and then the third answer, the updated model will look as follows:
Finally, after the sixth answer, we have:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 = False</td>
<td>0.298246</td>
<td>0.245614</td>
<td>0.192982</td>
<td>0.140351</td>
<td>0.087719</td>
<td>0.035088</td>
</tr>
<tr>
<td>A2 = False</td>
<td>0.413448</td>
<td>0.280401</td>
<td>0.173104</td>
<td>0.091559</td>
<td>0.035765</td>
<td>0.005722</td>
</tr>
<tr>
<td>A3 = True</td>
<td>0.199448</td>
<td>0.270531</td>
<td>0.250518</td>
<td>0.176674</td>
<td>0.086266</td>
<td>0.016563</td>
</tr>
<tr>
<td>A4 = ?</td>
<td>0.15</td>
<td>0.3</td>
<td>0.45</td>
<td>0.6</td>
<td>0.75</td>
<td>0.9</td>
</tr>
<tr>
<td>A5 = ?</td>
<td>0.15</td>
<td>0.3</td>
<td>0.45</td>
<td>0.6</td>
<td>0.75</td>
<td>0.9</td>
</tr>
<tr>
<td>A6 = ?</td>
<td>0.15</td>
<td>0.3</td>
<td>0.45</td>
<td>0.6</td>
<td>0.75</td>
<td>0.9</td>
</tr>
</tbody>
</table>

After the first test, the classification state of the learner is **Newcomer (0.454275)**.

**Test Session 2:**

The second test also assumes 6 test items. The initial values for the probabilities of the classifications are taken from the last row of the model data after the first test. The probabilities of a guess and slip remain the same.

<table>
<thead>
<tr>
<th>P(K1)</th>
<th>P(k2)</th>
<th>P(K3)</th>
<th>P(K4)</th>
<th>P(K5)</th>
<th>P(K6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.454275</td>
<td>0.344147</td>
<td>0.154582</td>
<td>0.041936</td>
<td>0.004999</td>
<td>0.000061</td>
</tr>
</tbody>
</table>

After the six answers, the updated model looks as follows:
Following the second test, the classification state of the learner is **Junior (0.433978)**.

**Test Session 3:**

The initial values for the probabilities of the classifications are taken from the last row of the model data after the second test. The probabilities of a guess and slip remain the same.

After the third test, the updated model looks as follows:

And the classification state of the learner is **Intermediate (0.538798)**.
APPENDIX D:

Adaptive Testing: Selecting the Next Test Item

The way the test symbols are administered is explained in the subsection “Step 5: Selecting the Next Symbol”.

Let us assume that the learner provided answers to three test symbols X, Y, and Z. The last answered test symbol was Z. The portion of the model that was updated is shaded in gray with bold letters. There are three more test symbols (W, V, and U) to be presented to the learner. One of them is to be selected and administered next. The data of W, V, and U represent the prior data or the data from the previous test session (this data is to be updated after the learner provides an answer for each).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>True</td>
<td>0.013161</td>
<td>0.216374</td>
<td>0.455624</td>
<td>0.275500</td>
<td>0.039150</td>
<td>0.000191</td>
</tr>
<tr>
<td>Y</td>
<td>True</td>
<td>0.004229</td>
<td>0.139072</td>
<td>0.439271</td>
<td>0.354150</td>
<td>0.062908</td>
<td>0.000368</td>
</tr>
<tr>
<td>Z</td>
<td>True</td>
<td>0.001269</td>
<td>0.083438</td>
<td>0.395320</td>
<td>0.424954</td>
<td>0.094356</td>
<td>0.000663</td>
</tr>
<tr>
<td>W</td>
<td>True</td>
<td>0.066119</td>
<td>0.330004</td>
<td>0.393073</td>
<td>0.183829</td>
<td>0.026750</td>
<td>0.000226</td>
</tr>
<tr>
<td>V</td>
<td>True</td>
<td>0.023820</td>
<td>0.237775</td>
<td>0.424826</td>
<td>0.264906</td>
<td>0.048185</td>
<td>0.000488</td>
</tr>
<tr>
<td>U</td>
<td>False</td>
<td>0.037606</td>
<td>0.309142</td>
<td>0.433978</td>
<td>0.196809</td>
<td>0.022374</td>
<td>0.000091</td>
</tr>
</tbody>
</table>

The current entropy $H(K)$ (the entropy of the last test symbol Z to which the learner provided an answer) is

$$H(K) = 1.6934881271992215$$
The conditional entropy of the remaining three symbols:

\[ H(K \mid A_w) = 1.6295558182300378 \]

\[ H(K \mid A_v) = 1.642065945132396 \]

\[ H(K \mid A_u) = 1.6174919721805243 \]

Considering the above, information gain for each remaining symbol is

\[ IG(K \mid A_w) = H(K) - H(K \mid A_w) = 0.06393230896918367 \]

\[ IG(K \mid A_v) = 0.05142218206682547 \]

\[ IG(K \mid A_u) = 0.07599615501869716 \]

Since the symbol U provides the greatest information gain larger than 0, it is the one to be administered next.