Java™ Intelligent Tutoring System Model and Architecture

Edward R. Sykes

School of Computing and Information Management, Sheridan College
1430 Trafalgar Road, Oakville, Ont., Canada, L6H 2L1
ed.sykes@sheridanc.on.ca

Abstract

Accessibility of computers and computer resources are increasing in our society at a staggering rate. Computer technology is changing more rapidly now than at any other time in history and the price of computers are continually decreasing inversely proportional to the power they deliver. Nearly 50% of households in Canada and the United States have computers [15]. Internet connections and capabilities are growing at an amazing rate due to the number of people who want to be connected to the world of information [15]. Internet Service Providers are upgrading their infrastructure to support real-time video and audio to their clients. Personal Digital Assistants such as cellular telephones and palm-pilots are Internet-ready and becoming commonplace in our society. In spite of the advances in computer technology and accessibility, educators have been relatively slow in seizing technology to enhance student learning. There are significant problems in the context of personalized student instruction in current educational systems that can be remedied through the use of appropriate technologies.

Online teaching tools such as WebCT and Blackboard are becoming extremely popular for distance and in-class education. In fact, entire universities have implemented online teaching tools as the central mechanism for delivering all of their courses [16]. The strength of these tools is their ability to provide the teacher and student with a great deal of versatility within the learning environment [16]. Unfortunately, they do not provide any means by which a student may receive ongoing personalized instruction. Teaching students on a one-on-one basis significantly influences the degree of knowledge and skill retained by the student; Bloom [3] showed that an individual human tutor can improve student learning by two standard deviations over classroom instruction. In other words, the average individually tutored student performs better than 98 percent of students in a classroom instructional environment [10].

This raises the following crisis in the educational community. In order for students to reach their potential they need individual tutoring. However, due to numerous limitations such as access to online teaching tools, financial considerations, and sheer logistics each student cannot be granted access to a personalized human tutor for a consistent duration of time. After all, traditionally there is only one teacher in a classroom of students. So, what can be done to solve this problem? One solution lies in the implementation of Intelligent Tutoring Systems.

Introduction

A technological counterpart to a human tutor called an Intelligent Tutoring System (ITS) may be an answer to the current crisis in education. ITSs are computer based systems designed using cognitive science and Artificial Intelligence (AI) techniques to provide students with individualized instruction in a similar way to human tutors.

The "Java™ Intelligent Tutoring System" (JITS) model focuses on designing an accelerated learning system using Intelligent Tutoring Systems (ITS) coupled with advanced artificial intelligence components. The framework of JITS will be sufficiently flexible to provide minimal configuration to enable other subject disciplines to be plugged-in (e.g., calculus, geometry, biology, etc.). An additional design aspect of JITS is its augmentation to popular web-based instructional tools such as WebCT and Blackboard.

The proposed model consists of two distinct components. First, an overview of the characteristics of existing Intelligent Tutoring Systems is described. Second, the design strategies of the Java™ ITS is presented. The JITS draws from the research advancements of ITS, cognitive science, and AI.

This project is not yet complete. However, it is hypothesized that the Java™ Intelligent Tutoring System will provide an interactively-rich learning environment for students that will result in increased achievement. Based on the success of similar Intelligent Tutoring Systems, it is also hypothesized that these students will be able to learn the course material more quickly than students in traditional classroom environments. The purpose is to demonstrate that learning may be accelerated and cognitive development deepened using this tutor.

At the core of an ITS is an AI module. The AI module is responsible for many tasks including capturing the student's knowledge state, delivering an appropriate lesson, assessing and evaluating student performance, and providing valuable feedback to the student. Thus, an ITS is explicitly designed to assist in resolving the crisis in education regarding personalized student education by providing a means through which students may improve their academic performance. The purpose of this study is to design and construct an ITS using advanced AI components for students and to perform a quantitative study to determine ITS' effectiveness.

This paper is divided into two parts. The first section presents an evaluation of the current state of Intelligent Tutoring Systems. The second section presents the Java™ Intelligent Tutoring System Model and Architecture.

Intelligent Tutoring Systems

The infrastructure supporting the development of Intelligent Tutoring System relies on two main activities: cognitive and information processing.

Cognitive Activities

These activities include such things as: perceiving, thinking, learning, remembering, and problem solving. In order for Intelligent Tutoring Systems to be effective for the student the learning process must support two distinct purposes. First, the ITS must exhibit cognitive activities so that the learner will interact and respond appropriately to the tutoring process. Second, the ITS must be able to identify these types of activities in the learner and plan the next set of steps in the student's learning process.

Information Processing Activities

The details behind cognitive activities include a range of information processing activities that take sensory data and engage in processes to create meaningful information for some specific purpose. For example, some of these activities involve the following:

- ***Acquiring:*** paying attention to what is happening and perceiving the relevant;
- ***Encoding:*** transforming sensory data into mental propositions and constructs for processing;
- ***Associating:*** relating new mental propositions and constructs to existing knowledge;
- ***Storing:*** keeping information and knowledge for future use. This involves short-term, intermediary, and long-term memory stores;
- ***Retrieving:*** timely access to stored information and knowledge;
- ***Communicating:*** producing results and desired outcomes; sharing knowledge with others.

The philosophical and pedagogical framework of the proposed Intelligent Tutoring System based on cognitive science and Artificial Intelligence. The framework is based on the foundational cognitive and information processing activities. This architecture is represented by six components: Curriculum Knowledge, Student Modeling, Expert Modeling, Teacher/Tutor Modeling, Mixed-initiative, and Self-Learning. Figure 1 depicts the framework of an Intelligent Tutoring System.

Curriculum Knowledge Modeling (CKM)

The Curriculum Knowledge Modeling component generates appropriate instructional material based on the context of the student's performance. Components in this module include problems, solutions, exercises, hints, and help. The CK module presents the student with the appropriate instructional information to stimulate student learning while minimizing discourse and frustration. The ITS needs to be able to distinguish from pre-defined responses and genuine responses that would most help the student.

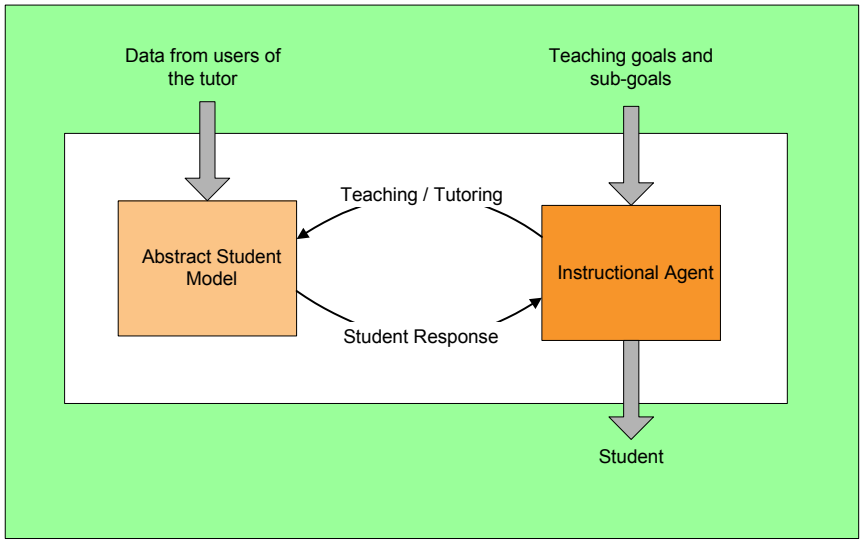

Figure 1. Java™ Intelligent Tutoring System Framework

Abstract Student Modeling (ASM)

This component assesses the student's knowledge and performs instructionally useful actions based on the assessment. The assessment need not be formalized as a quiz or test, but may include informal approaches such as coaxing or subtle probing questions too.

Expert Modeling (EM)

This component of the ITS models expert performance and does something instructionally useful based on the knowledge of the domain. Current systems vary in the type of knowledge they teach. Some ITS teach formal logic and formal knowledge while others teach overall processes in the context of the domain. For example, AnimalWatch, MathTutor, Algebra tutor all associate with mathematics teach formal logic and knowledge [2, 1]. On the other hand, Intelligent Tutoring Systems such as MeTutor focuses on processes involved in firefighting [13].

Constructing the expert model requires specifying the relative difficulty of the topics, knowledge of the strategies that can be used, and a large amount of analogies, examples and error diagnosis abilities to effectively tutor the student[18].

Instructional Modeling (IM)

This module is designed to change instructional strategies based on changes in state of the student model. Depending on the domain various strategies may be used, such as, explanation, guided discovery learning, coaching, and/or probing. Human teachers and tutors do this regularly within the delivery of a lesson [3]. How and why human teachers alter their teaching style is an interesting question and is currently being researched [10, 18]. Research in cognitive science will lead to more effective modules that will better adapt to the needs of the student [18].

Collaborative Modeling (CM)

This component of the ITS is based on the development of human-computer interaction [18]. The dynamics involved are based on human-human interaction that have been studied and analyzed from a communication perspective [5]. As a result, the mixed-initiative module determines the most effective way for the ITS to communicate with the student while ensuring rich interactivity and productivity. Natural language dialog is used more frequently in the design of recent Intelligent Tutoring Systems [5]. One goal is to establish a balance between the student controlling the conversation and the ITS. Another responsibility of the Mixed-Initiative module is its ability to recognize mistakes. ***Error diagnosis*** is the term used for a system's ability to diagnose mistakes, reasonable misconceptions, and recognize missing information.

Self-Learning (SL)

The ITS needs to have the capacity to monitor, evaluate, and improve its own teaching performance as a function of experience. In other words, the Abstract Student Model reflects upon what was successful, what was not, and refines the process for the current student and future students (see figure 1).

Java ITS Model

This section presents the model and architecture for the Java™ Intelligent Tutoring System (JITS). Three distinct components are presented that support the JITS: the infrastructure architecture, the user interface (student view) architecture, and the curriculum architecture.

Java ITS Infrastructure Architecture

The JITS infrastructure will support the client via a browser accessing information from the tutor via the HTTP request/response process model. The processing will be accomplished by a set of Javabeans on a web server housing the business logic for the tutor. The presentation layer will be provided by JavaServer Pages™ technology which will communicate with Javabeans for processing and return a simple html page back to the student's browser. During processing the Javabean will access an appropriate AI module for seeking information from the JITS. This may entail selecting a suitable skill-level problem for the student, or providing feedback in the form of an appropriate hint based on the current level of performance. The infrastructure architecture also provides a JDBC connection from Javabeans to an external database to store and retrieve specific information about the student including student history and performance statistics.

The proposed architecture has numerous benefits. For instance, the student will not need to install any software on their machine and will not need a high-speed network connection to the JITS. Other benefits include comparably expected faster execution as all processing will be done on a middle-tier server which typically has significantly faster and more efficient hardware. These benefits have a net result of increasing the accessibility for the JITS to many students – a vital requirement for an equitable and successful educational product in today's Internet-ready community. The Java™ ITS Infrastructure Architecture is depicted in Figure 2.

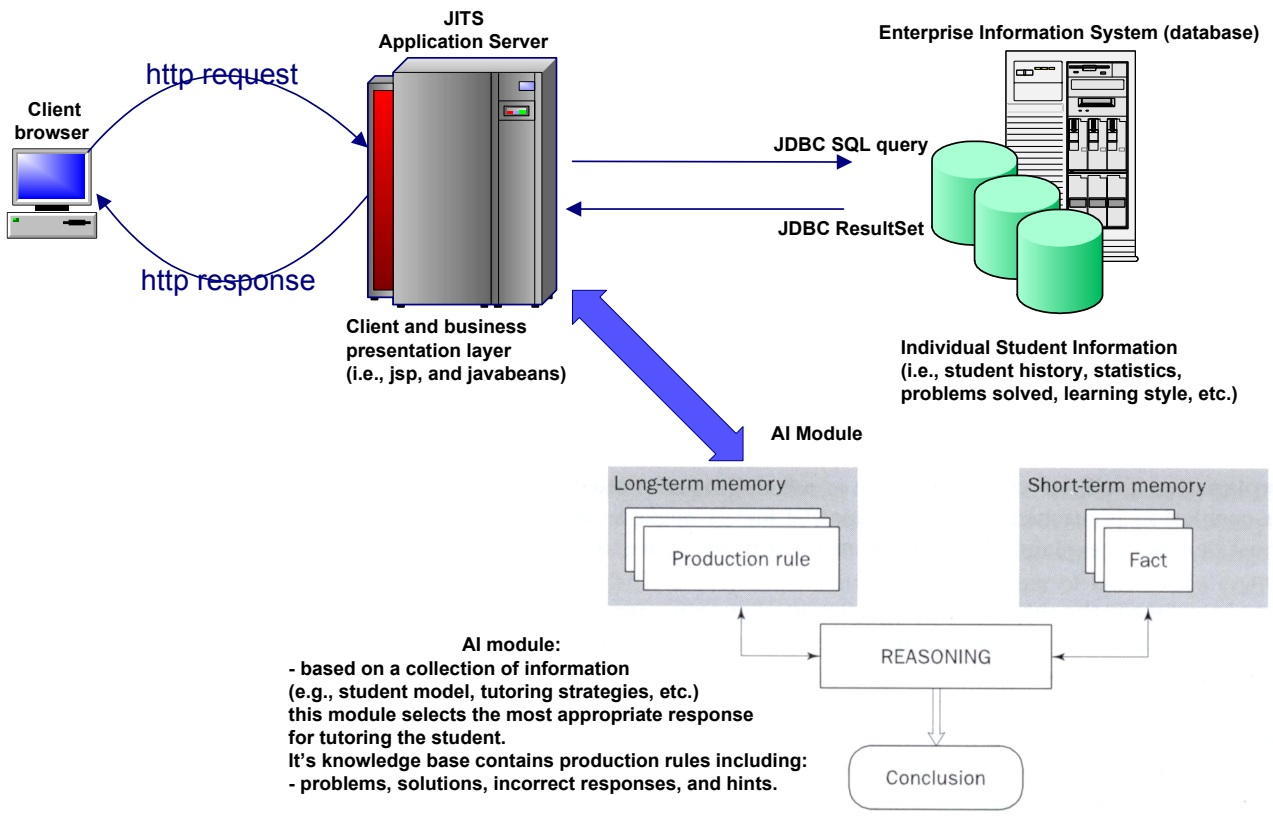

Figure 2. Java™ ITS Infrastructure Architecture

Java ITS User Interface

This section describes the user interface (student view) of the JITS. The following provides a description of the issues involved during a session.

Upon connecting to the JITS web site, the student's browser displays the working environment for the JITS. An appropriate skill-level problem is selected or the problem that last attempted is presented to the student.
The student can type in a solution in the Source Code area and then press 'Parse'. This invokes a refined java parser which checks that the student's code is grammatically correct. If the parser is satisfied, then the student may then press the 'Compile' button. This makes a platform specific invocation to 'javac' which currently is implemented for Microsoft Windows 95, 98, NT, 2000, and XP. If the java compiler is satisfied, then the 'Run' button becomes enabled which invokes the platform specific 'java' binary. The output of the student's running program is displayed in the student's browser.

Appropriate feedback is provided to the student in the form of hints, and solutions (if requested). The students, at any time, may opt to quit a problem and select another from a bank of problem sets in the database. Additionally, the student can view a performance summary based on various statistics gathered including problems attempted, problems solved, number of attempts on a problem, and problem difficulty. A depiction of the interface is found in Figure 3.

Java™ Intelligent Tutoring System (JITS) User Interface

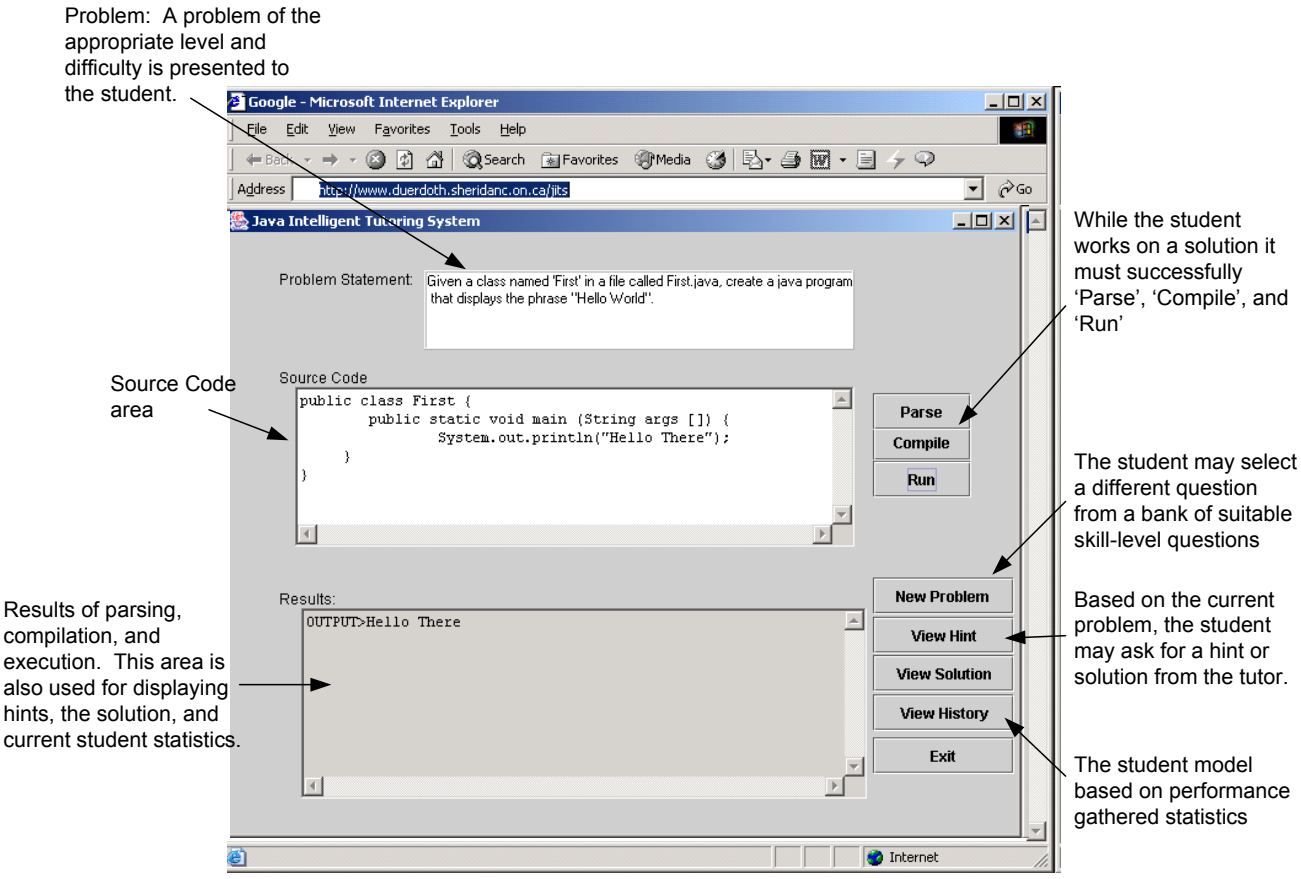

Figure 3. Java™ ITS User Interface Architecture

Java ITS Curriculum Architecture

This section describes the curriculum architectural model for the JITS. Due to the complexity involved with semantic parsing it is necessary to restrict the JITS to tutor a small subset of the Java programming language. Currently, the topical areas of focus involve the following list of Java™ language basics:

a. variables (declaration, use, local vs. global),

b. operators, and

c. looping structures.

A database of records with the following structure will be constructed:

Given a Problem (P), there are n number of Solutions (S_1, S_2, …, S_n), with a finite set (m) incorrect responses (R_1, R_1,…, R_m). For each incorrect response there is a finite set (t) of hints (H_1, H_2,,…,H_t). Table 1 illustrates an example of a problem with solutions, incorrect responses, and hints.

Table 1. Java™ ITS Curriculum Architecture.

Problem:
Write a program called "Summer" which adds all the integer numbers from 1 to a specified number (N). For example, if N were assigned the value 10, then the sum of the numbers from 1 to 10 is 55.

Program specifications:
This program requires the use of a for-loop structure. A skeleton structure of the solution is given. Fill in the code to complete this program.

```
OUTPUT>Sum = 55
```

Skeleton Program (given to student in Source Code area):

```
public class Summer {
 public static void main(String[] args)
 {
   int sum = 0;
   int i = 0;
   int n = 10;

    <<Student types code here>>

   System.out.println("Sum = " + sum);
 }
}
```

Solution (one of n):

```
public class Summer {
   public static void main(String[] args) {
     int sum = 0;
     int i = 0;
     int n = 0;
     for (i = 1; i <= 10; i++) {
         sum += i;
     }
     System.out.println("Sum = " + sum);
  }
}
```

Incorrect response #1 (student response area):
(redeclaration of variable 'i')

```
     for (int i = 1; i <= 10; i++) {
         sum += i;
     }
```

Incorrect response #2:
(sum does not include last integer (i.e., ' 10')

```
     for (i = 1; i < 10; i++) {
         sum += i;
     }
```

Incorrect response #3:
(sum is 0, as the body of the loop is never executed)

```
     for (i = 1; i > 10; i++) {
         sum += i;
     }
```

Incorrect response #4:
(adding 1 instead of variable 'i': results in sum being lower than expected)

```
     for (i = 1; i <= 10; i++) {
         sum += 1;
     }
```

Incorrect response #5:
(incorrect formula)

```
     for (i = 1; i <= 10; i++) {
         sum = i + i;
     }
```

Incorrect response #6:
(correct formula, but incorrect incrementing of i)

```
     for (i = 1; i <= 10; i=i+i) {
         sum = sum + i;
     }
```

Hints:

The JITS will gather information regarding the problem specification (e.g., specific variables or specific constructs to be used in the solution), the output expected, the parse tree representation of the student's solution, the results of parsing, compilation and executing the student's program. All these elements will be used to perform a very specific form of semantic analysis on the student's solution. In this fashion, appropriate hints can be generated for the student.

Conclusions

In summary, the Java™ Intelligent Tutoring System model is designed to leverage on the recent advancements in cognitive science and artificial intelligence, to extend the science of building Intelligent Tutoring Systems, and to assist in helping students learn better and faster.

Although the project is in progress, the model is based on sound theories and practices used in successful Intelligent Tutoring Systems and draws from the achievements artificial intelligence has provided in other disciplines.

Acknowledgements

I wish to thank Dr. Rosemary Young and Dr. Franya Franek for the years of support and direction they have given me.

References

[1] Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive Tutors: Lessons learned. *The Journal of the Learning Sciences, 4,* 167-207.

[2] Beal, C. R., Beck, J. E., Woolf, B. P., & Rae-Ramirez, M. A., (1998). WhaleWatch: An Intelligent Model-Based Mathematics Tutoring System. *Proceedings of the Fifteenth IFIP World Computer Congress, Ed., J. Cuenal,* (pp. 472-483). Austria: Austrian Computer Society

[3] Bloom, B. S. (1984). The 2-sigma problem: The search for methods of group instruction as effective as one-to one- tutoring. *Educational Researcher, 13*, 4-16.

[4] Gallant, S., I. (1993). *Neural Network Learning and Expert Systems*. MIT Press, Cambridge, MA.

[5] Graesser A. C., Person, N. K., & Harter, D. (2001). Teaching tactics and dialog in autotutor. *International Journal of Artificial Intelligence in Education, 12,* 12-23.

[7] Haykin, S. (1994). Neural Networks: A Comprehensive Foundation. Macmillan College Publishing Company, New York.

[8] Jang, J.-S., R., Sun, C.-T., and Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence.* Prentice Hall, Englewood Cliffs, NJ.

[9] Jacobs, R., A. (1988). Increased rates of convergence through learning rate adaptation. Neural Networks, 1, pp. 295-307.

[10] Koedinger, K. R. (2001). Cognitive tutors. In K. D. Forbus & P. J. Feltovich (Eds.), *Smart machines in education* (pp. 145-167). Cambridge, MA: MIT Press.

[11] Minsky, M. (2001). An introduction to artificial intelligence. *Generation 5: At the forefront of artificial intelligence*. Retrieved July 10, 2002, from http://www.generation5.org/aihistory.shtml

[12] Negnevitsky, Michael (2002). Artificial Intelligence: A Guide to Intelligent Systems, Pearson Education Limited, England.

[13] Rowe, N. C., & Galvin T. P. (1998). An authoring system for intelligent procedural-skill tutors. *IEEE: Intelligent Systems, 14,* 61-69.

[14] Russell, S., J., and Norvig, P. (1995). Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs, NJ.

[15] Sciadas G., (2001). The Digital Divide in Canada. Statistics Canada. Retrieved November 12, 2002, from http://www.statcan.ca/Daily/English/021001/d021001e.htm

[16] Sikora, A. (2002). A Profile of Participation in Distance Education: 1999-2000. Postsecondary Education Descriptive Analysis Report. Retrieved November 12, 2002, from http://nces.ed.gov/pubs2003/2003154.pdf

[17] Watrous, R., L. (1987). Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization, Proceedings of the First IEEE International Conference on Neural Networks, San Diego, CA., vol. 2, pp. 619-627.

[18] Woolf, B., P., Beck, J., Eliot, C., & Stern, M. (2001). Growth and maturity of intelligent tutoring systems: A status report, In K. D. Forbus & P. J. Feltovich (Eds.), *Smart machines in education* (pp. 100-144). Cambridge, MA: MIT Press.