7-2007

Developmental Process Model for the Java Intelligent Tutoring System.

Edward R. Sykes
Sheridan College, ed.sykes@sheridancollege.ca

Jl. of Interactive Learning Research (2007) **18***(3)*, 399-410

Developmental Process Model for the Java Intelligent Tutoring System

EDWARD SYKES
Sheridan College, Canada
ed.sykes@sheridanc.on.ca

The Java Intelligent Tutoring System (JITS) was designed and developed to support the growing trend of Java programming around the world. JITS is an advanced web-based personalized tutoring system that is unique in several ways. Most programming Intelligent Tutoring Systems require the teacher to author problems with corresponding solutions. JITS, on the other hand, requires the teacher to supply only the problem and problem specification. JITS rigorously analyzes the student's submitted code, determines the intent of the student, and intelligently guides the student towards a potentially unique solution to the programming problem. JITS is intended to be used by beginner programming students in their first year of College or University. This article discusses the process by which the design and development of JITS took place. JITS has been and is currently being field-tested at the Sheridan Institute of Technology and Advanced Learning.

This article presents the development model used in the design and construction of the Java Intelligent Tutoring System (JITS; Sykes & Franek, 2004d). In this article, many anecdotal comments are included to illustrate the issues, problems, and solutions that were encountered in the development of JITS. Input from students and instructors proved to be invaluable in this process. JITS was field-tested from April, 2004 to December, 2004 at the Sheridan Institute of Technology and Advanced Learning. Additional field-tests were scheduled for fall 2005. A brief overview of the JITS is presented including the initial User Interface and the "intent" recognition module. The remainder of this article discusses student and instructor input on the developmental process of JITS.

JITS DEVELOPMENTAL PROCESS MODEL

The design of the JITS heavily relied on an embedded intelligence module entitled the Java Error Correction Algorithm (JECA) to provide the necessary information to offer suitable feedback to the student programmer (Sykes & Franek, 2004c). There were several additional factors, aside from JECA, that were integral to the development of JITS: students and instructors. In order for an ITS to be successful in today's e-learning society, JITS was designed with these perspectives in mind.

Student Perspective

The following qualities were deemed important in the design of JITS to satisfy students:

1. provide an easily understood student-friendly user interface that provides all the necessary features for effective ITS tutoring;
2. provide access through an ordinary browser;
3. will not need a high-speed internet connection (i.e., dial-up connection will work fine, thus, students in remote locations have full access to this resource);
4. process student's code submission and respond quickly to the student;
5. support many students concurrently working with the ITS;
6. engage the student by communicating in a clear and concise personalized fashion (e.g., unique hints and error messages);
7. track student performance in a database (e.g., ORACLE); and
8. model the user as s/he works through a problem;

Instructor Perspective

The following factors were important in meeting the needs of teachers using this ITS:

1. requires the author of the problem to provide minimal information (e.g., problem statement, program requirements and required output);
2. the author of the problem does ***not*** specify any solutions (this is based on the premise that for a given programming problem there may in fact be numerous solutions);
3. JITS must be able to recognize a very large number of possible solutions for a particular programming problem;
4. student performance information should be easily accessible; and
5. an instructor-friendly web-based user interface to author problems (i.e., Authoring Tool);

The following section presents the initial JITS User Interface and a description of the initial web-based infrastructure architecture.

INITIAL JITS USER INTERFACE

The initial User Interface of JITS provided a basic window in which the student could interact with the tutor (Sykes, 2003). Many features in JITS' User Interface are representative of professional programming Integrated Development Environments (IDE). The underlying design aspects of the interface provide students with a problem, the problem specification, the skeleton code, the code editor, and a number of buttons with which to interact with the tutor. Figure 1 presents the initial design of the JITS User Interface.

During the design stages, the JITS User Interface underwent some changes. The completed JITS User Interface prototype consolidated three buttons: "Parse," "Compile," and "Run" into a single button entitled "Submit." It was felt this was much clearer for students since the operation of these three buttons could be executed behind the scenes with JITS' interpretation of the results displayed in the "Output" region. The first prototype JITS User Interface is presented in Figure 2.

INFRASTRUCTURE DESIGN

The infrastructure design for JITS draws from leading-edge techniques and technologies for multi-threaded distributed concurrent e-learning appli-

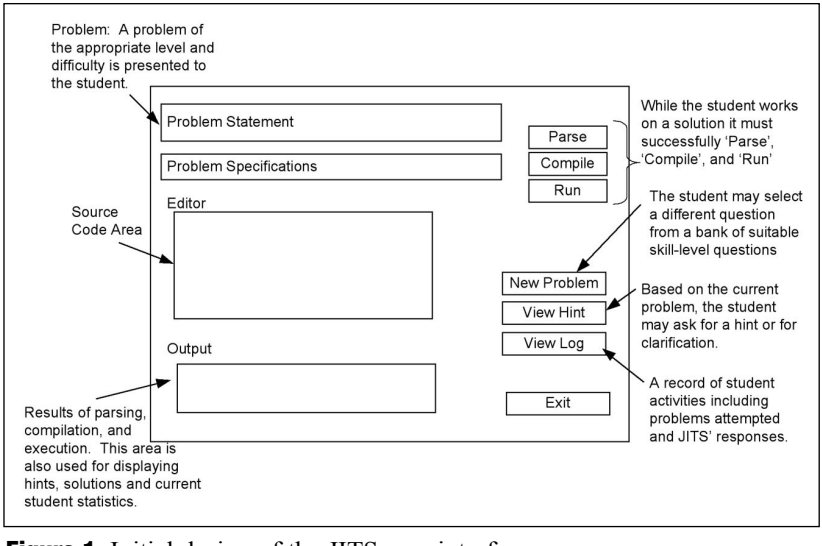

Figure 1. Initial design of the JITS user interface

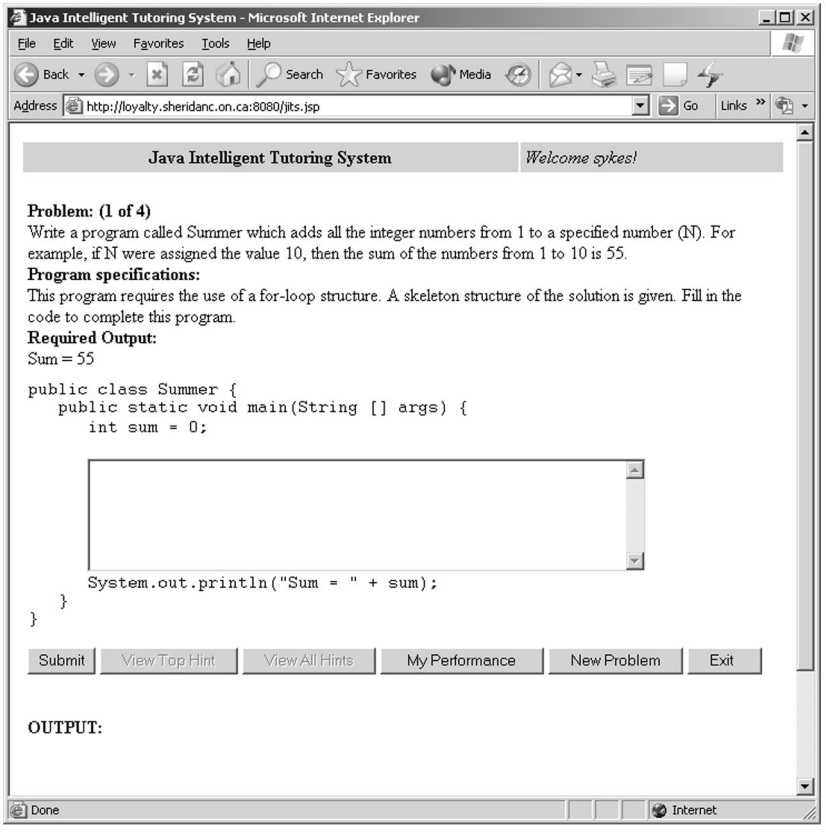

Figure 2. First prototype of the JITS user interface

cation designs. The Model-View Controller (MVC) design pattern was used to ensure that concurrency and robustness would be provided by JITS. The MVC contains three main tiers: (a) the client's browser, (b) the middle-tier, and (c) the database tier. By design, there were no restrictions placed on the browser. In other words, JITS was designed to work with any browser and no custom installed client software of any sort was required. The middle-tier is a server running a TomCat web server, currently equipped with 4GB RAM, and 2 Pentium-IV processors. The database-tier is a separate server running ORACLE. The initial JITS database schema was designed to support the core functionality of JITS consisting of three tables: student, problems, and student-problems. The student table contains information regarding each student in the system such as student name, password, current problem, and so forth. The problems table contain details regarding programming problems used by JITS such as problem description, specifications,

templates, and so forth. The student-problems table is an intersection relation representing details regarding each student's attempt at a problem. The Model-View-Controller design pattern was a core component to the design of JITS. Figure 3 depicts the MVC design pattern. First the student makes a request (through HTTP in the browser). The Controller module receives the request and performs operations that including instantiating JavaBeans. These beans are used to model the student as s/he works with JITS. The collection of these beans represents the modeling of each student in JITS. During specific operations, beans may need to retrieve information from the JITS database schema (e.g., to select a new problem, or retrieve solutions to a problem, etc.). This data is stored in the ORACLE JITS database schema represented in the figure as the Enterprise Information System (EIS). The information is gathered up and processed by the bean which then forwards it to the View component (i.e., the Java ServerPage (JSP)) which then formats it appropriately for the student in the JITS user interface and returns it to the student's browser.

JITS INTENT RECOGNITION MODULE

The initial design of the Java Intelligent Tutoring System involved the core intelligence unit called JECA (Java Error Correction Algorithm). This module performs a number of operations behind the scenes. It implements a sophisticated scanner and parser that autocorrects the student's code when appropriate and constructs a number of parse trees that have small variations. This module then attempts to parse and compile the best trees to ascertain the

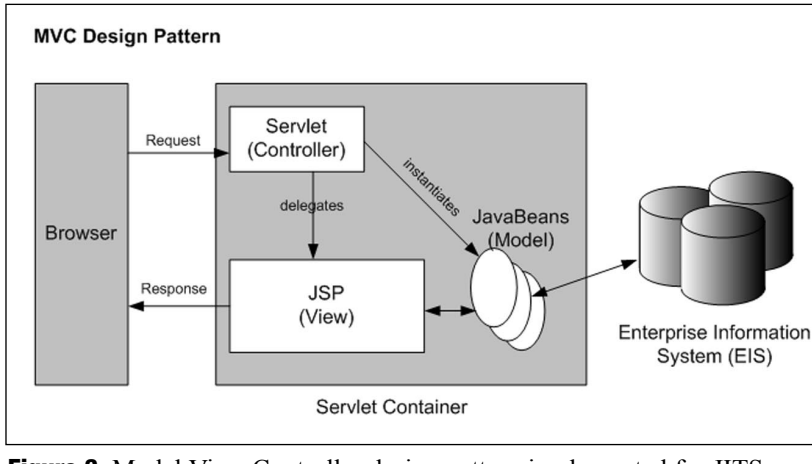

Figure 3. Model View Controller design pattern implemented for JITS

most likely path the student "intended" to follow. With this knowledge, JITS can efficiently and effectively tutor the student. The goals JECA are to:

1. intelligently recognize the "intent" of the student;
2. analyze the student's code submission;
3. "auto-correct" where appropriate (e.g., converting "While" into the keyword "while," "forr" into "for," etc.);
4. learn individual student's misconceptions, and categorizes the types of errors s/he makes;
5. produce a "modified code" that will compile (or bring the code closer to a state of successful compilation); and
6. prompt the student programmer for information when necessary via well-defined hint support structures.

METHODOLOGY

The methodology employed in this project was cyclical: it related to the manner in which JITS was designed, constructed, tested, redesigned, and so forth. In field-tests, students and professors using the prototype JITS offered suggestions and comments for the improvement of JITS. The new knowledge was fed back into the redesign and construction of JITS. This research methodology involved qualitative instrumentation including observation, surveys, and personal interviews. Table 1 presents the qualitative survey sheet.

Participants

The population in this study was students in their first year of college taking a beginner Java programming course at the Sheridan Institute of Technology and Advanced Learning. During the summer of June to August 2004, there were two such classes taking this course. The class at the Davis campus was the experimental group (i.e., JITSC). The other class, at the Trafalgar Road campus, was the control group (i.e., C). One professor taught both classes for first seven weeks. After a mid-term break in week eight, another professor took over and taught both classes for the remainder of the term (i.e., for the last seven weeks). Fourteen students consented to use the JITCS. Approximately every week _ to 1 hour long sessions were conducted by the researcher to elicit specific information about students' experiences with the Java Intelligent Tutoring System. Professors were also asked to participate in this study. The selection of professors was based on a number of factors including their knowledge of the Java programming language, level of course offerings, and interest in offering critical opinions on the JITS. A total of four professors were involved in this study.

Table 1
Interview Sheet

Project Interview

I am conducting a survey of those participants who were taught using the Java Intelligent Tutoring System at Sheridan. The information gathered from our interview will be used for my research. This involves determining the effectiveness of learning in this environment. For each question, select the most appropriate response based on the following scale:1 = strongly favourable to the concept, 2 = somewhat favourable to the concept, 3 = undecided, 4 = somewhat unfavourable to the concept, 5 = strongly unfavourable to the concept. The following questions will be asked during the interview.

1. How do you rate the Java Intelligent Tutoring System's usefulness?

Very Useful				Not Useful
1	2	3	4	5

Comments: _____

2. Do you feel the Java Intelligent Tutoring System is beneficial to your studies? List and explain the advantages/disadvantages of this learning environment.

Very Beneficial				No Benefits
1	2	3	4	5

Comments:_____

3. Compare JITS with a traditional classroom. Do you feel JITS is better or worse than an ordinary classroom teaching environment? Identify any similarities and differences between a traditional classroom experience and the JITS learning experience.

JITS is much better than traditional classroom				JITS is much worse than traditional classroom
1	2	3	4	5

Comments:_____

4. How do you rate the ease with which you use and understand the tutoring style of the JITS?

Very easy to use & understand				Very difficult to use & understand
1	2	3	4	5

Comments:_____

5. Have you enjoyed JITS? Explain why or why not.

Very Enjoyable				Not enjoyable
1	2	3	4	5

Comments:_____

6. Do you feel you learn more detailed information or about the same as a regular classroom when using JITS? Explain why or why not.

Learn Better				Learn the same
1	2	3	4	5

Comments:_____

FINDINGS

Students and teachers alike provided a number of suggestions that were made to improve the design of JITS. They were based on the improvement of the screen navigation keys, the user interface, and the design of new buttons. A number of students stated that they wanted to be able to see the solution after a certain number of attempts at a problem or if they get frustrated. Essentially, they said, "It would be nice for solution button to be available." As a result, the researcher designed and developed a "View Solution" button with supporting infrastructure. In the JITS teachers are not required to submit solutions during problem authoring. This is based on the premise that given virtually all programming problems there are potentially limitless solutions. Supplying only one solution for a given programming problem is not an acceptable approach. As a result, a Collective-Student-Model representing the sum knowledge of all students was designed and developed. This Collective-Student-Model analyses all students' submissions and extracts those which are solutions to the particular problem the student is currently working on. The AI-module uses the information in Collective-Student-Model to determine appropriate feedback. The revised JITS user interface is shown in Figure 4 with a small example illustrating this additional functionality due to student suggestions.

Several students suggested a list of different programming topics should be available on the side of JITS User Interface page. The students suggested that it would make JITS more useful and attractive for many students. It

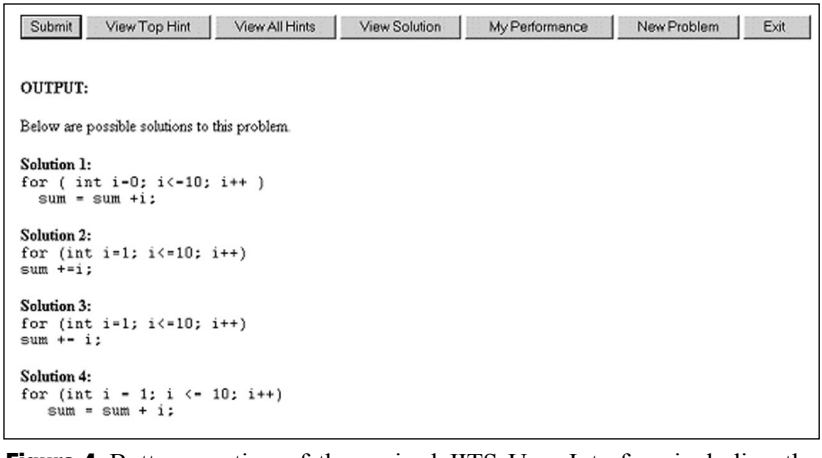

Figure 4. Bottom section of the revised JITS User Interface including the "View Solution" button displaying all unique solutions for the current problem

would also add a more professional appearance to JITS. These students suggested that, "Students want to have a choice over their own learning to be able to select the area of study (e.g., Java basics, arrays, loops, etc.)." Addressing this problem required quite a bit of work. In order to fulfill the request, a complete redesign of the following JITS components was necessary: the ORACLE database schema, the User Interface, and JITS' internal infrastructure. In total, the researcher spent several hundred hours to redesign JITS to provide the following programming topics: Java Basics, Java Statements, if statements, for loops, do while loops, while loops, and Arrays.

During this experiment a number of interesting issues were raised. One student suggested that she would like to be able to navigate both forwards and backwards through the problem sets. The current JITS system allowed only forward movement through the problems to encourage incremental skill development by presenting more difficult problems. However, the researcher decided that allowing the student full control over the movement through the problem sets has merits. As a result, the researcher designed the infrastructure for the "Previous Problem" and tested it out.

Furthermore, during the field-test, students mentioned they would be interested in seeing a small tutorial section be incorporated into JITS. This tutorial section would explain each of the different programming topics designed and developed earlier in JITS. The purpose of the JITS tutorial would be to remind students about basic problem solving strategies and basic syntax associated with a specific Java constructs (e.g., if statements, for-loops, etc.). The Tutorial window provides detailed mini-lessons on programming topics that are cross-referenced with the programming problems within JITS. The revised JITS User Interface is presented in Figure 5 including the "Previous Problem" and "Next Problem" buttons, Programming Topics, and "View the Tutorial" button.

FINAL VERSION OF THE JITS USER INTERFACE

Some students suggested that actual equations be presented in problems that refer to mathematical expressions. For example, in one of the problems the information was presented only as text: "… the volume of a cone is 1/3 times PI times the radius squared times the height. …" Students suggested it would be clearer to present this information as: $\frac{1}{3} \pi r^2 h$. For this problem the researcher first added another column to the Problems table of the JITS database schema. The researcher then developed a Windows Popup which streams a binary image from the ORACLE database. The researcher then updated the main JITS webpage (i.e., jits.jsp JavaServer Page) to include a link "View the image for this problem." If the student clicks on the link, a new window is created and the image for the problem is streamed inside. This approach has the certain benefit including students being able to see the

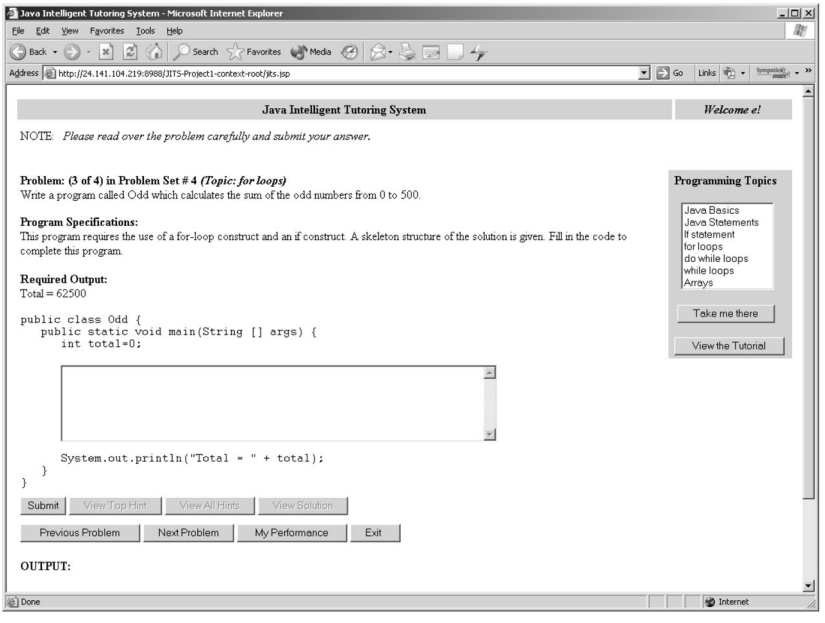

Figure 5. Redesigned JITS User Interface depicting "Previous Problem" and "Next Problem" buttons, the list of Programming topics and a "View the Tutorial" button

image viewer contents while working on a solution to a programming problem in the code editor of JITS. (This information may be seen in Figure 6 – label 4 is the link which enables the Image Viewer Popup.)

After several months of redesign and retesting to incorporate the requests of students and instructors the final version of the JITS User Interface was completed. This latest version of JITS' User Interface has the following features. When a student is ready to submit the code to JITS, the "Submit" button may be pressed. The student can then view the hints by pressing "View Top Hint" or "View All Hints." Every hint is dynamically constructed and is context sensitive based on JITS' understanding of the student's submission. At any time, the student can see their performance by pressing the "My Performance" button, which provides detailed information regarding the number of questions attempted, questions solved, and relative ranking based on other students in the course. Additional feature of the User Interface include the "Help Me" and "Tutorial" buttons. The "Help Me" provides information regarding the organizational layout of the JITS user interface. Figure 6 depicts the final User Interface for JITS.

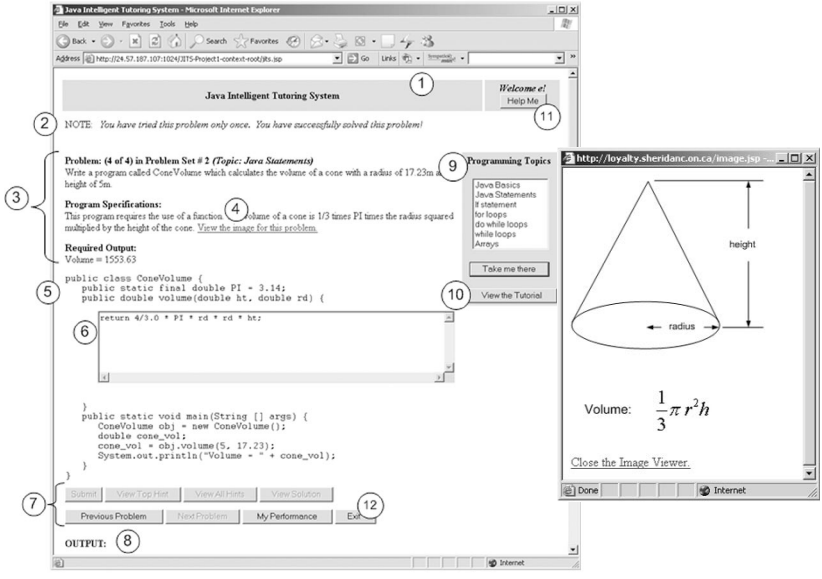

Figure 6. Final version of the JITS User Interface with Image Viewer Popup

CONCLUSIONS

The JITS prototype has been a success (Sykes & Franek, 2004b). JITS was met with interest by students and professors alike. After attempting several problem sets, both groups were happy with the performance of JITS. JECA enables JITS to significantly improve the performance of beginner Java programmers (Sykes & Franek, 2004a). In fact, during the summer term of 2004, a two-way ANOVA with repeated measures was conducted confirming these results, $F(1,35) = 4.162$, $p = .049$, indicating there was a significant statistical difference in performance scores between the control group (i.e., students that did not use JITS) and experimental groups (i.e., students who used JITS on a regular basis; Sykes & Franek, 2004a).

JITS was field-tested from April, 2004 to December, 2004 at the Sheridan Institute of Technology and Advanced Learning. Additional field-tests have been scheduled for future academic terms. The development and refinement of JITS is by no means over. There are a number of other exciting suggestions raised by students and professors that are being reviewed. For instance, some students would like to see some gaming elements in JITS, others said video streaming. Research into these areas is scheduled for early 2006. Integration of these requested features will be available in the future releases of the JITS.

References

Sykes, E. R. (2003, March). Java intelligent tutoring system model and architecture. *Proceedings of American Association of Artificial Intelligence Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments* (pp. 187-193), Palo Alto, CA.

Sykes, E. R., & Franek, F. (2004a). *Field-report of the java intelligent tutoring system.* Retrieved October, 2004, from http://lttf.ieee.org/learn_tech/

Sykes, E. R., & Franek, F. (2004b, June). *Preliminary assessment of the java intelligent tutoring system*. Paper presented at the International Conference on Education and Information Systems, Technologies and Applications, Orlando, FL.

Sykes, E. R., & Franek, F. (2004c, November). *Presenting JECA: A java error correcting algorithm for the java intelligent tutoring system*. Paper presented at the IASTED International Conference on Advances in Computer Science and Technology, St. Thomas, Virgin Islands, USA.

Sykes, E. R., & Franek, F. (2004d). A prototype for an intelligent tutoring system for students learning to program in Java. *International Journal of Computers and Applications, 1*(1), 35-44.

Acknowledgements

This article is a republication of a paper presented at ED-MEDIA 2005.