A Web-based ITS for OO Design

Glenn Blank, Shahida Parvez, Fang Wei and Sally Moritz

Computer Science and Engineering Department
Lehigh University
Bethlehem, PA 18015 USA
001-610-758-4085, 001-610-758-4867
{gdb0, smp9, faw2, sgh2}@lehigh.edu

Abstract. Learning object-oriented design and programming is a challenging task for many beginning students. CIMEL ITS coordinates student learning in two client programs: web-based multimedia courseware (CIMEL) and the Eclipse IDE, each of which post student interactions to a server-based CIMEL ITS. The Expert Evaluator analyzes student work in Eclipse, comparing novice with expert solutions. The Student Model combines knowledge from the expert evaluator and the multimedia. Finally, the Pedagogical Agent, guided by updates from the student model as well as a learning styles inventory, interacts with the learner by selecting from several tutorial strategies. All three components share knowledge from a Curriculum Information Network, which represents a new "design-first" introduction to software development in Java.

1. Introduction

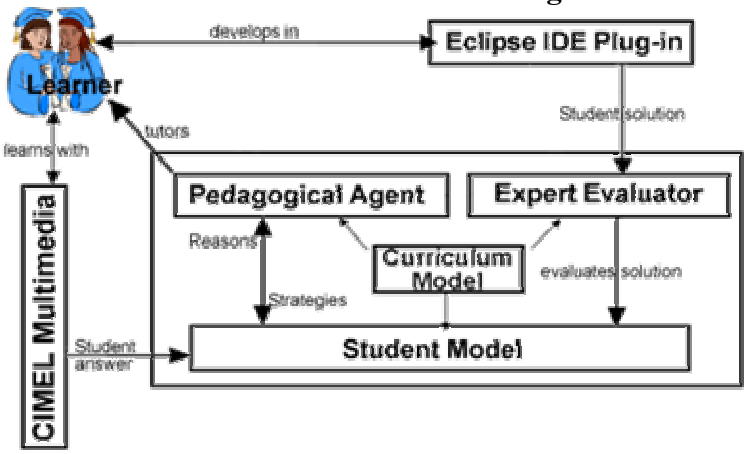
The application of ITS to programming has been limited to specific procedural aspects of programming languages such as LISP [2, 4], Pascal [20], C++ [11] and Java [17], not keeping up with "objects-first" pedagogy let alone recent trends in object-oriented design. Object-oriented design and programming are challenging for beginners, with large percentages of students struggling to understand basic concepts [12, 16]. We created web-based interactive multimedia courseware supporting an "objects-first" approach using BlueJ [10]. Experimental evaluation demonstrates that CIMEL multimedia improves conceptual understanding, but does not necessarily improve performance on a programming task [14]. These results motivate the development of an adaptive tutoring system, CIMEL ITS, observing student work with both the multimedia and the Eclipse programming environment.

CIMEL ITS is designed to interact with students through two client programs. First, multimedia courseware called CIMEL (Collaborative, constructive, Inquiry-based Multimedia E-Learning) introduces concepts in object-oriented design and Java [3]. CIMEL includes its own interactive activities, introduces small design/programming exercises, and gives quizzes based both on its own content and on programming exercises. CIMEL posts all student interactions to a database on a server, which the Student Model of CIMEL ITS observes. Experimental evaluation demonstrates that this multimedia improves conceptual understanding, but does not necessarily improve performance on a programming task [14]. These results motivate the development of an adaptive tutoring system, CIMEL ITS, which observes student work with both the multimedia and the Eclipse programming environment.

Second, students solve problems in the Eclipse integrated development environment (IDE), configured with plug-ins for UML design and interactive programming. Our UML plug-in also sends student work to an Expert Evaluator, which compares it with expert solutions. The CIMEL ITS, running on a server, will thus integrate observation of each student's progress from two different client-based perspectives. It will then offer assistance based on adaptive pedagogical strategies, within either the multimedia (such as reviewing prerequisite knowledge) or the IDE (such as improving flawed design).

The rest of this paper presents the design of CIMEL ITS as follows. Section 2 describes the flow of control from the multimedia through the ITS, then section 3 describes the flow of control from Eclipse through the ITS, using concrete examples. Section 4 summarizes innovations of our architecture and planned future work.

2. Flow of control from multimedia through the ITS



The figure on the left shows the architecture of CIMEL ITS. In its heart is the Curriculum Model, which represents knowledge about a novel curriculum. Before getting bogged down in the details of coding, we believe students should get a glimpse of the big picture of software development. We therefore propose a "design first" approach to learning software development, in which students learn object-oriented analysis and design as problem-solving skills [13, 14]. In our curriculum, students learn how to develop use cases to analyze problems, then design solutions using UML, before implementing much code in Java.

The Curriculum Model [19, 21] represents the knowledge that students must learn, design-first, in a Curriculum Information Network (CIN). We use the CIN to provide core domain knowledge for all three active components of the ITS, simplifying their design. The CIN links concepts together to show relationships between them. A concept may be identified as having one or more prerequisite concepts, and it may also be a component of another, higher-level concept. For example, in order to understand the concept of a method, one must know the concept of class, parameter, return value, etc. On the other hand, a method could be considered a component of an object. A difficulty measure is also assigned to each concept within the CIN. The three active parts of the ITS refer to the CIN to tie the student's learning activities to concepts in the curriculum.

The CIMEL ITS can interact with students either through CIMEL multimedia (on the left) or the Eclipse IDE (upper right), each of which initiate different flows of control through the ITS architecture. One possibility is that a student begins viewing the multimedia chapter "Objects and Classes". This chapter introduces basic object concepts, including classes, attributes, methods and instances. Interspersed throughout the content are interactive exercises and quizzes, which reinforce concepts and measure the student's understanding.

Suppose a student is learning how to identify the attributes for a class. The multimedia shows several examples (such as a "Tree" class which has attributes height and color). In a drag-and-drop exercise, the student is presented with things that represent a class, an object and an attribute. The student must classify these things by dragging and dropping them in containers labelled class, object and attribute. Later she completes a quiz with 4 multiple-choice questions relating to understanding attributes. For example: "Characteristics of an object (such as size and color of a dress) are called _____?" [learner selects from: attributes, methods, instances or classes]."

The CIMEL client records each screen the student visits, each interaction in exercises, and each quiz result to a server-side database, which the Student Model (SM) monitors. The SM represents student understanding of concepts as Bayesian networks corresponding to the concept structure in the CIN. The SM has three layers. First, the problem-domain layer is a set of graphs, each of which models the probability that a student understands a target concept and

its prerequisites, given their performance on a target exercise. Suppose an exercise asks a student about the concept on a UML attribute. There is a link from the node for the attribute concept to a node for the student's answer. There are also links from the prerequisite concepts, the concepts object and class to the attribute concept. The attribute concept also appears as root nodes in other networks for which it is a prerequisite. Once the SM determines the probability of the attribute concept, all other concept networks for which attribute is a prerequisite are updated.

Second, the historical knowledge layer updates a model of the student's knowledge state for student solutions to the same problem or multiple problems. If there are many wrong answers, this layer identifies possible reasons, such as that the student doesn't understand the target concept (attribute), or the prerequisite concepts (object and class), or the student may have forgotten these concepts over time (say, more than three days).

Finally, the cognitive layer [8] infers whether a student exhibits general problem solving patterns (such as decomposition or analogy) or antipatterns (such as blind hacking). For instance, if the student skipped the drag-and-drop question, the cognitive layer infers that the student may not be paying attention. The SM packages its diagnosis, including all possible reasons with probability values, as a packet, which it sends to the PA.

The Pedagogical Agent (PA) provides feedback and tutoring to the student. It consists of a feedback network and tutoring strategies. The feedback network is similar to the CIN, adding feedback constructs for each concept in the domain knowledge. Feedback is assigned a numerical level indicating if the feedback is basic or advanced. For example, the feedback consisting of concept definitions will be assigned level 1. The tutoring strategies, which may be represented by distinct agents, include a traditional tutoring strategy in which an agent plays the role of a tutor, and variations of cooperative learning strategies [5]. The "learning by disturbing" strategy employs a traditional tutor agent and a companion agent that attempts to test the student's knowledge by misleading him [1]. The "learning by teaching" strategy also has a traditional tutor agent and a companion agent that learns along with the student [15, 16]. In addition to the diagnosis from the SM, the PA considers the student's preferred learning style of the student. When the student uses CIMEL ITS for the very first time, she fills out a learning styles questionnaire [7], which categorizes individual learning styles based on the Felder-Silverman learning style model [6]. This model categorizes individual learning styles on a sliding scale of five dimensions: sensing-intuitive, visual-verbal, active-reflective, sequential-global, and inductive-deductive. The PA selects its tutorial strategy from the feedback network by combining the SM's diagnosis and the student's learning style.

For example, if a student has already reviewed a concept in CIMEL but the SM reports several mistakes and inattention, the PA will consult the student's preferred learning style to provide guidance about how the student could get more out of the multimedia. If the student's learning style indicates that she learns better with concrete facts, the PA will recommend that the student review a section of the HTML-based "Just The Facts" material, then follow the links to the quizzes. If the PA judges that the student cannot learn more from the multimedia and is still performing weakly, it can send a diagnostic report to a human instructor via e-mail.

3. Flow of Control from Eclipse through CIMEL ITS

The multimedia also directs the student to begin work on an assigned problem in Eclipse. The student then starts building a simple movie ticket vending machine in a student-oriented Eclipse plug-in supporting the design of classes in Unified Modeling Language (UML). As the student enters each piece of her solution (class name, each attribute, each method), the plug-in sends data to the Expert Evaluator (EE) for analysis.

The EE tries to match the student's entry with a corresponding part of an acceptable solution, which it generates from an instructor's textual description of a problem. Whenever the student completes an action, the EE will evaluate the student solution and generate a

data packet with the evaluation results. When the EE is able to match the component entered by the student, it will generate a packet containing the student's action, a packet count (how many packets were generated for this step) and the concept in the Curriculum Information Network to which the action relates. If the student correctly creates a class called TicketMachine and adds two attributes, movieTitle and ticketPrice, three packets will be sent to the SM.

Like quiz results for the multimedia, each Eclipse action is a leaf node in a problem-domain model. For the target concept class-name, class and object are prerequisite concept nodes, and the Eclipse action naming the class TicketMachine is the leaf node. Successfully performing this action updates the probability of the class-name concept. This probability will in turn be passed to the historical knowledge layer, which updates the student profile with the student's solution. The SM appends the probability that the student understands the target concept to the packet it received from the EE, then passes it to the PA.

The PA will play the role of an experienced tutor who encourages the student when he does well and offers help when it is needed. Determining the timing and frequency of positive feedback is tricky. The agent should try not to distract or annoy the student; on the other hand it does not want the student to feel abandoned. This is particularly important for female students who may lack confidence about their coding ability or by peers who have a head start [9]. The agent will use each student's preferred learning style and gender to determine the frequency of positive feedback. If the student prefers active learning, then the pedagogical agent will provide encouraging phrases more often than if the student prefers reflective behavior.

Suppose the student enters an attribute called "ticketsRequested." The EE will attempt to match the attribute on possible acceptable attributes for the TicketMachine class. When it doesn't find a match, it will search through other parts of its solutions: other class names, attributes in other classes, methods, and method parameters. A common student error is misidentifying a valid element of the solution: in this case, the student has identified a data item as an attribute that is more appropriate as a parameter to a method (the "printTickets" method). The EE infers valuable information about the student's reasoning.

Student errors are tied to CIN concepts. The CIN concept "attribute" is tied to the action of adding an attribute, but the definition of an attribute has more than one component: an attribute represents a characteristic, or data item describing or used by the class, and it is a value that must persist through the life of the object. These two parts of the definition are separate nodes in the CIN, and represent components of the CIN node for attribute. This particular error is tied to the "persistence" portion of the attribute definition.

The EE now generates an error packet which contains the student's action, the error (misidentified a parameter as an attribute), the CIN concept tied to the error, the correct action for the step (if any), and the actual text from the problem description that applies to this step. It then sends this error packet to the SM.

Since this example leads to multiple target concepts in the CIN, the SM will generate a corresponding structure of multiple target nodes in the problem-domain layer. The problem-domain layer calculates the probabilities for each target concept. After the historical knowledge layer updates itself, it may infer possible reasons for the error, such as the student doesn't understand an attribute is a property or an object-life value, or she does not understand the prerequisite concepts. Depending on the student's solution history, the cognitive layer may discern that the student is using preconceived notions to put the "ticketsRequested" as an attribute, because she did a similar action in another design exercise in a different context. The SM appends its diagnosis of possible reasons and probability values to the EE packet, sending it along to the PA.

When the PA receives an error packet, the first thing it will do is to check if the student had difficulty with the same concept before by looking at its feedback history.

Suppose there is no record that the student had any problem with this concept before. Next, the PA chooses the reason with the highest probability and uses it to get the associated concept from the CIN. Then, the PA uses the reason, concept and preferred learning style to retrieve tutoring content from the Feedback Information Network. In this case, the concept that the student is having a problem with is "Attribute". The feedback information for this concept is multileveled; first it is the problem specific feedback which is tied to the current problem. For example, it could say "Remember, you don't need to keep track of tickets requested." The next level feedback is domain specific level 1 feedback which could say "The attributes are values that need to be kept track of for the life of the object." Feedback for each concept will have many different components, such as verbal, visual, global, sequential, intuitive, and sensing components. Each of these components maps to a type of learning style. Each component will contain feedback that matches with that learning style. For example the visual component could have a picture of an object that specifies its attributes while the intuitive component could describe what an attribute actually means.

4. Conclusions and Future Work

The CIMEL ITS architecture offers several innovations. First, it integrates knowledge about what the student is learning from two sources: web-based multimedia and the Eclipse IDE. Both tools are designed to support a novel "design-first" approach to learning object-oriented software development. Second, the EE performs structured matches of student UML class designs with corresponding snippets of expert solutions, which it generates from instructor problem descriptions. Its step-by-step analysis reduces the complexity of the match and improves responsiveness. Third, the SM makes inferences about reasons for a student's correct or erroneous behaviours at three levels: the current student work, the student's history, and general problem-solving patterns and antipatterns. Fourth, the PA selects pedagogical strategies by combining the SM diagnoses and a learning styles inventory. Finally, it abstracts conceptual knowledge about the domain in a common Curriculum Information Network, facilitating the synergy of the three active components (each the topic of a Ph.D. dissertation). Separating much of the domain knowledge into a common repository simplifies the design of the three active components: the EE accounts for student behaviours in terms of CIN concepts; the SM generates estimates of student knowledge about these concepts; and the PA indexes a Feedback Information Network that parallels the structure of the CIN. While our work is still in the design and early implementation stage, we believe that our architecture makes useful improvements on the standard tri-partite model in terms of reusability.

We plan to demonstrate and discuss the first release of CIMEL ITS at the workshop. After further development and testing, we plan to evaluate CIMEL ITS in both university and high school settings.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants No. EIA-0087977 and 0231768 and the Pennsylvania Infrastructure Technology Association. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or PITA. The first author also gives thanks to God for inspiration and blessing.

References

[1] Aïmeur, E. and Frasson, C. Analyzing a New Learning Strategy According to Different Knowledge Levels. *Computers in Education*, vol. 27, no. 2, 1996, pp. 115-127.

[2] Anderson, J.R. and Skwarecki, E. (1986) The automated tutoring of Introductory Computer Programming. *Communications of the ACM*, vol. 29, September 1986, pp 842-849, ACM Press.

[3] Blank, G. D., Barnes, R. F. and Kay, E. J.. *The Universal Computer: Introducing Computer Science with Multimedia* (McGraw-Hill/Primis, 2003/2004). Sample material at www.cse.lehigh.edu/~glennb/um/ and cimel.cse.lehigh.edu.

[4] Brusilovsky, P., Schwarz, E., & Weber, G. (1996). ELM-ART: An Intelligent Tutoring System on World Wide Web. In Frasson, C., Gauthier, G., & Lesgold, A. (Ed.), *Intelligent Tutoring Systems* (Lecture Notes in Computer Science, Vol. 1086). Berlin: Springer Verlag. 261-269.

[5] Chan T. W. and Baskin A. B. Learning Companion Systems. In *Intelligent Tutoring Systems: At the crossroads of Artificial Intelligence and Education* (Edited by Frasson, C. and Gauthier, G.), Chap 1. Ablex, N.J., 1990.

[6] Felder, R. M. and Silverman, L. K. *Learning Styles and Teaching Styles in Engineering Education*, Engr. Education, 78, 7, 674-681 (1988).

[7] Felder, R. M. and Soloman, B. A. *Index of Learning Styles Questionnaire*, Available online at June (2003) in:http://www.ncsu.edu/felder-public/ILSdir/ilsweb.html

[8] Gürer, W. D. A Bi-level Physics Student Diagnostic Utilizing Cognitive Models for an Intelligent Tutoring System, PhD Dissertation, Lehigh University, 1993.

[9] Irani, L. Understanding Gender and Confidence in CS Course Culture, *Proceedings of the Thirty-Fifth SIGCSE Technical Symposium on Computer Science Education*, Norfolk, VA, March 2004, 195-199.

[10] Kölling, M., Quig, B., Patterson, A. and Rosenberg, J., The BlueJ System and its Pedagogy, *Journal of Computer Science Education,* vol. 13, no. 4, Dec 2003.

[11] Kumar, A. Model-Based Reasoning for Domain Modeling in a Web-Based Intelligent Tutoring System to Help Students Learn to Debug C++ Programs, *6th International ITS Conference*, Biarritz, France and San Sebastian, Spain, June 2002.

[12] McCracken M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Yifat Kolikant, Y., Laxer, C., Thomas, L., Utting, I., Wilusz, T., A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-Year CS Students. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, December 01, 2001, Canterbury, UK

[13] Moritz, S. and Blank, G. A Design-First Curriculum for Teaching Java in a CS1 Course, to appear in *SIGCSE Bulletin (inroads)***,** June, 2005.

[14] Moritz, S., Wei, F., Parvez, S., and Blank, G. D. (2005), From Objects-First to Design-First with Multimedia and Intelligent Tutoring, *The Tenth Annual Conference on Innovation and Technology in Computer Science Education*, Monte da Caparica, Portugal, June, 2005.

[15] Palthepu, S., Greer, J., and McCalla, G. Learning by Teaching. *The Proceedings of the International Conference on the Learning Sciences*, AACE, 1991.

[16] Ratcliffe, M. B.. Improving the Teaching of Introductory Programming by Assisting the Strugglers. *The 33rd ACM Technical Symposium on Computer Science Education*, Cincinnati, USA, March, 2002.

[17] Sykes, E.R. and Franek, F.. A Prototype for an Intelligent Tutoring System for Students Learning to Program in Java. In *Advanced Technology for Learning,* vol. 1, no. 1, 2004.

[18] Wei, F., Moritz, S., Parvez, S., and Blank, G. D. (2005), A Student Model for Object-Oriented Design and Programming, *The Tenth Annual Consortium for Computing Sciences in Colleges Northeastern Conference*, Providence, RI, April 2005.

[19] Wescourt, K., Beard, M. & Gould, L. Knowledge-based adaptive curriculum sequencing for CAI: Application of a network representation, *Proceedings of the National ACM Conference*, Seattle, Washington, 1977.

[20] Woolf, B. and McDonald, D., Human-Computer Discourse in the Design of a PASCAL Tutor, *Proceedings of Conference on Human Factors in Computing Systems,* Boston, MA, 1983.

[21] Zhou, G. Tsong-Li, J. Wang, P., and Ng, A., Curriculum Knowledge Representation and Manipulation in Knowledge-Based Tutoring Systems. *IEEE Transactions in Knowledge Data Engineering* 8(5): 679-689, 1996.