STUDENT MODELING FOR LEARNING OF OBJECT-ORIENTED PROGRAMMING IN E-LEARNING ENVIRONMENT

Abdullah Mohd Zin; Sufian Idris
Universiti Kebangsaan Malaysia, Selangor, Malaysia

Nantha Kumar Subramaniam
Open University Malaysia, Kuala Lumpur, Malaysia

ABSTRACT

E-Learning environment offers some interesting benefits over traditional learning environments. Students' learning in e-learning environment can be independent of distance, time, computing platform and classroom size. E-Learning also provides diversity and completeness of knowledge. In an e-learning environment, students learn by going through learning materials provided through a web-based learning management system. However, most of the learning management systems are nothing more than a network of statically linked hypertext pages. To better support the learning activities of a large number of students with different learning styles, and diverse background and learning goals, there is a need for a more adaptive and intelligence learning management system. To achieve this objective, this system must be able to provide a more personalised approach for learners. This is particularly important in the learning of programming, since different learner would normally face with different problem. The most important component for developing a more adaptive and intelligent learning management system is the students model. The purpose of this paper is to investigate and then to propose a students model for learning of object-oriented programming in an e-learning environment. The development of this model is based on our experience in handling courses on object-oriented programming at Open University Malaysia (OUM) over more than four years.

INTRODUCTION

E-Learning environment offers some interesting benefits over traditional learning environments. Students' learning in e-learning environment can be independent of distance, time, computing platform and classroom size. E-Learning also provides diversity and completeness of knowledge. In an e-learning environment, students learn by going through learning materials provided through a web-based learning management system. However, most of the learning management systems are nothing more than a network of statically linked hypertext pages. To better support the learning activities of a large number of students with different learning styles, and diverse background and learning goals, there is a need for a more adaptive and intelligence learning management system (ILMS) or Intelligent Tutoring System (ITS). To achieve this objective, these systems must be able to provide a more personalised approach for learners. The most important component for developing a more adaptive and intelligent learning management system is the students' model (or user model). Student model is the essential component in enabling individualised learning in ITS. It is the student model that builds and maintains the system's understanding of the student (Stauffer, 1996). The knowledge base of the student model use both subject matter and pedagogical knowledge (Stauffer, 1996). The user model is the stored information that the system has learned or captured from the user. This could be as simple as the user's name or a complex analysis of the abilities of the user (Stauffer, 1996). Most ITS or ILMS have incorporated user models to increase their ability in giving personalised learning support to the learners. There have been many system architectures that has been proposed in implementing intelligent learning environment especially for ITS. Two examples of the ITS architectures are shown in Figure 1 and Figure 2.

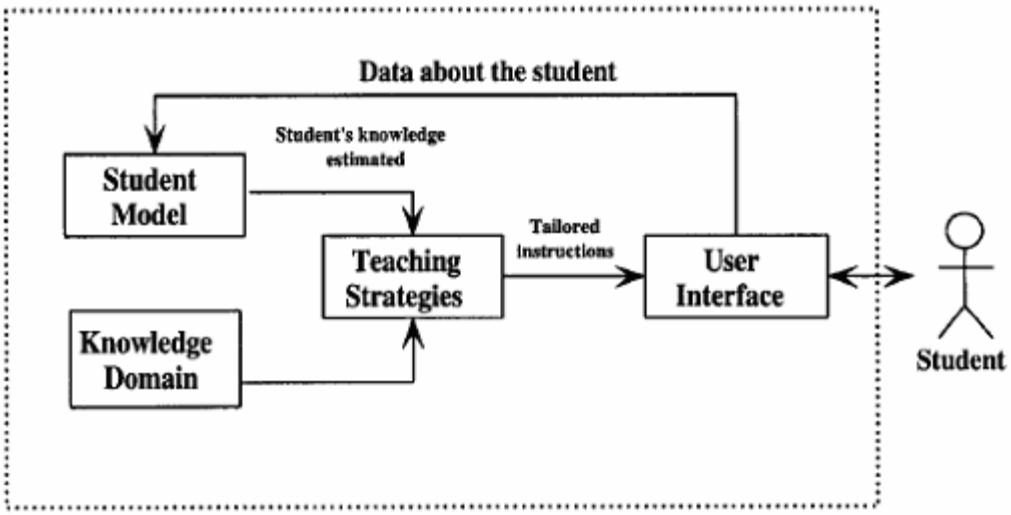

***Figure 1**: ITS architecture as proposed by Hua S (2004)*

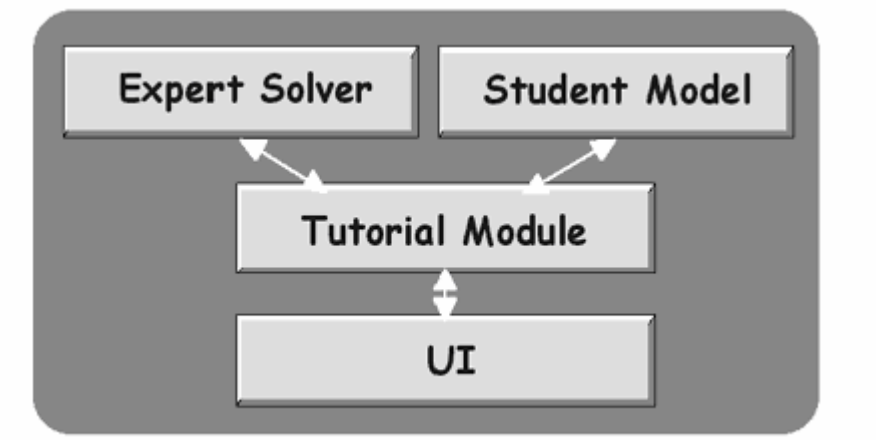

***Figure 2**: ITS architecture as proposed by Alpert S.R., Singley M.K., Fairweather P.G. (1999)*

Most of these architectures for ITS are different to each other in the terms of the components that have been considered (as could be seen in Figure 1 and Figure 2). However, one thing common among all the ITS is they have a student model component as their important component. This reflects the importance of student model in ITS. Student modelling, as the model of a learner, represents the computer system's belief about the learner's knowledge (Stauffer, 1996). It is first necessary to capture the student's understanding of the subject in order to allow instruction to be individually presented. With this information, the difficulty of material, and any necessary remediation can be controlled within the system (Stauffer, 1996).

Building a student model involves defining the following (Stauffer, 1996):

- the degree of specialisation of the learner;
- the history of the learner;
- the goals, plans, attitudes, capabilities, knowledge, and beliefs of the learner;
- how the model is to be acquired and maintained;
- interpreting the learner behavior and giving feedback.

Few factors need to be considered in maintaining the student model. This includes the fact that students do not perform consistently, forget information randomly and display large leaps in understanding (Stauffer, 1996).

STUDENTS MODEL FOR OBJECT-ORIENTED PROGRAMMING (OOP)

Most of the student models that were incorporated in ITS or ILMS are not domain-specific but rather a generic student model that could be applied to all fields. This approach may be acceptable for general courses and conventional learners but may not suitable for highly technical subjects and for those students who have enrolled in e-learning or open distance learning environment. ITS that need to be developed for difficult domains such programming needs a refined student model that will consider many issues related to the subject matter knowledge and other dimensions. Furthermore, this technical subject has their own learning abstraction which is unique to their field. In addition, students studying programming in an e-learning environment belongs to a diverse population with different learning styles, motivations and entry level. Different programming learner would normally faced with different problem The idea of a dedicated student models according to the domains is very much consistent to the research issues that need to be considered when developing a student model as outlined by VanLehn (1998). Some of the research issues raised by VanLehn (1998) are:

- Which diagnostic techniques work well with which student models on which subject domains?
- Are ITS based on fine-grained student models more effective than those based on coarse-grained student models?
- Should student models incorporate models of learning in order to reduce the diagnostic complexity?
- Research on mental models as a knowledge type represents an important contribution to student modelling.

Object-oriented programming (OOP) has become the most influential programming paradigm (Kolling, 1999). It is widely used in education and industry and universities teach object-orientation as part of their curriculum. Learning object-oriented programming is a good thing due to the following reasons as envisioned by Kolling (1999):

- It supports the concepts such as well structured programming, modularisation and program design;
- It supports programming in teams, maintenance of large systems and software reuse.

Object-oriented programming seems to be a good tool for teaching those programming methodologies that could be considered important (Kolling, 1999). However, learning object-oriented programming remains difficult due to the many factors. Learning OOP is difficult because it requires a new way of thinking about computing and more depth to grasp (Hadjerrouit, 1999).

A number of studies have attempted to identify factors contributing to success or failure in OOP. Factors that was considered include: demographic profiles (Rountree, Rountree and Robins, 2002), mathematics capability (Boyle, Carter and Clark, 2002), entry qualifications (Alexander et al., 2003), prior experience of programming (Morrison and Newman, 2001 and Ventura, 2005) and self-efficacy and expectations of success (Ramalingam and Wiedenbeck, 1998). Since 1970's, many innovative approaches have been proposed by educators to overcome problems in teaching and learning of programming. Most of these approaches can be classified under two main categories, namely: pedagogical and tools. Some examples of new pedagogical approaches in teaching of programming are "Tutorial-based teaching of introductory Programming" (Zachary, 1994), "Methodology First and Language Second" (Zhu and Meng, 2003) and "Process Model" approach (Gantenbein, 1989). Under tools, intelligent learning environment such as ITS has been developed to help students to learn OOP. The need for a ITS for the object-oriented programming domain is becomes important as the number of students enrolled in this course in the e-learning and open & distance learning (ODL) environment is increasing.

However, there is no complete student model that available for OOP that could be "ported" into ITS.

While many ITS have been developed to stimulate student in learning OOP but it fails to deliver its objective. One of the main reasons for this is the badly designed student model that has been integrated into ITS. Our objective is to develop a student model for OOP based on our experience teaching this course at Open University Malaysia (OUM) for more than four years. OUM is higher level institution that specialises in e-learning and ODL environment. We have also have interviewed 20 students to obtain their feedback on learning of OOP during the semester of January 2006 at OUM. The following are the common problems reported by the students:

- Students do not know how to install Java compiler. Thus, early frustration is discovered among the students;
- Students confuse between object oriented programming and normal programming. Students wrongly apply the knowledge that they gained from normal programming in OOP;
- Students cannot differentiate object and class which are the important concepts in OOP;
- Students' learning style does not match the content provided;
- Students cannot understand the basics of OOP. Hence, they cannot follow the advance concepts of OOP;
- Students are frustrated due to Internet connection which was very slow;
- Type of learning method preferred by the students was not matched with the given learning material.

In general, students feedback could be classified into three categories namely: demographic profile, subject matter knowledge and learning style. We have also notice from our observation that female students tend to perform better than the male students in OOP. As such, ITS may need to have different mechanism to support female and male students in OOP. It also seems that blended students and online students who enrolled in OOP course need a different kind of support which need to considered in developing a ITS . Based on the above common problems faced by the students and our observations towards the students who are enrolled in OOP, we are proposing the following student model to be captured in a ITS for optimum result for learning of OOP among the students in e-learning or ODL environment:

***Table 1**: Proposed Student Model for Learning of OOP in E-Learning or ODL Environment*

STUDENT MODEL ELEMENTS
Demographic Profile
Age
Type of Internet connection
Gender
Subject Matter Issues
Object first vs Syntax First
Program generation vs program comprehension
Skill in installing compiler
Understanding level of object
Understanding level of class
Understanding level of object interface
Understanding level of message passing
Understanding level of methods
Understanding level of encapsulation
Understanding level of inheritance
Understanding level of polymorphism
Understanding level of Applet
Understanding level of GUI components
Understanding level of layout managers
Understanding level of event handling
Learning Profile
Learning Style
Student type: Blended or Online
Type of learning method (case-based study, problem-based study, inquiry-based study)

The student model proposed in Table 1 has included three main areas namely: demographic profile, OOP domain and the learning style. This is very much consistent to Ragnemalm (1996) who considers two kinds of knowledge, namely subject matter knowledge and pedagogic content knowledge but Ragnemalm has most probably focusing on conventional students and overlook e-learning students. Hence, our student model has included demographic profile which better reflect the diverse requirements of the e-learning or ODL learners.

CONCLUSION AND FUTURE WORK

A student model that is domain-specific is very much important in developing an effective ITS. We have proposed a student model for OOP that could be "ported" into a ITS. This student model was developed based on our observations and interactions with the students enrolled in OOP course at OUM. This paper has give us some insights about the elements that need to be considered to develop a student model for ITS in the domain of OOP. We plan to refine this student model using a quantitative experiment using a large number of students and to correlate with some determined variables. Later, an ITS will be designed and developed which will use the student model proposed in this paper.

REFERENCES

Alexander, S., Amillo, J., Boyle, R., Clark, M., Daniels, M., Laxer, C., et. al. (2003). Case studies in admissions to and early performance in Computer Science degrees. *ACM SIGCSE Bulletin*, 35(4), 137-147.

Alpert, S. R., Singley M. K., & Fairweather, P. G. (1999). "Deploying intelligent tutors on the Web: An architecture and an example". *International Journal of Artificial Intelligence in Education Vol. 10, 183-197*.

Boyle, R., Carter, J., & Clark, M. (2002). What makes them succeed? Entry, progression and graduation in Computer Science. *Journal of Further & Higher Education*, 26(1), 3-18.

Gantenbein, R. E. (1989). Programming as Process: A novel approach to teaching programming ACM SIGCSE. *Proceedings of the twentieth SIGCSE technical symposium on Computer science education*, 21(1).

Hadjerrouit, S. (1999). ACM SIGCSE Bulletin. *Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education ITiCSE '99,* 31(3).

Hua S. (2004). BITS: *A Bayesian Intelligent Tutoring System for computer programming*. Programme, University of Regina. Location.

Kölling, M. (1999). The Problem of teaching object-oriented programming, part 1: Languages, *Journal of Object-Oriented Programming*, 11(8), 8-15.

Morrison, M., & Newman, T. S. (2001). A Study of the impact of student background and preparedness on outcomes in CS1. ACM SIGCSE Bulletin, 33(1), 179-183.

Ragnemalm, E. L. (1996). Student diagnosis in practice; Bringing a gap. *User Modeling and User-Adaptive Interaction*, 5(2), 93-116.

Ramalingam, V., & Wiedenbeck S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research*, 19(4), 367-381.

Rountree, N., Rountree, J., & Robins, A. (2002). Predictors of success and failure in a CS1 course. *ACM SIGCSE Bulletin*, 34(4), 121-124.

Stauffer, K. (1996). Student modeling and web-based learning systems. Retrieved July 29, 2006 from http://ccism.pc.athabascau.ca/html/students/stupage/Project/initsm.htm#cognitive.

VanLehn, K. (1998). Student modelling. In M.C., Polson, J.J., Richardson, (eds.), *Foundations of intelligent tutoring systems*. Lawrence Erlbaum: Hilldale, NJ, 55-78.

Ventura, P. (2005). Identifying Predictors of success for an object-first CS1. *Computer Science Education*, 15(3), 223-243.

Zachary, J. L. (1994). Tutorial-based teaching introductory programming classes. *ACM IGCSE Bulletin. Proceedings of the twenty-fifth SIGCSE symposium on Computer science education*, 26(1).

Zhu, H., & Meng, C. Z. (2003). Methodology first and language second: A way to teach object-oriented programming. *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages and applications*. Location: Publisher.