Ogata, H. et al. (Eds.) (2015). Proceedings of the 23rd International Conference on Computers in Education. China: Asia-Pacific Society for Computers in Education

Building Open Student Model based on Assessment Framework of Iteration Programming

Chih-Yueh CHOU[ad*], Chien-Tse WANG[a], Zhi-Hong CHEN[bd], Shu-Fen TSENG[cd], & Po-Yao CHAO[bd],

[a]*Department of Computer Engineering and Science, Yuan Ze University, Taiwan*
[b]*Department of Information Communication, Yuan Ze University, Taiwan*
[c]*Department of Information Management, Yuan Ze University, Taiwan*
[d]*Innovation Center for Big Data and Digital Convergence, Yuan Ze University, Taiwan*
*cychou@saturn.yzu.edu.tw

Abstract: Programming involves complex skills and knowledge. Novice students encounter many difficulties and lack information for reflecting and improving their proficiencies in programming skills and knowledge. This paper presents a work-in-progress research to design a system with an open student model based on an assessment framework of iteration programming. The assessment framework includes strategies of counter-controlled loop and sentinel-controlled loop, implementation skills of *while*, *for*, and *do-while*, and program tracing abilities. The system presents open student models to students in order to assist students in reflecting their proficiencies in iteration programming.

Keywords: Open student model, computer assisted programming learning, Iteration programming

1. Introduction

Programming is a problem-solving activity, which involves many cognitive processes, such as analyzing problems and adopting problem-solving strategies and plans to design a program, implementing codes to generate the program, and testing, tracing, and debugging the program (Robin, Rountree, and Rountree, 2003). Students need many knowledge, strategies, models, and skills to complete these cognitive processes. Studies reveal that novice programmers encounter many difficulties and lack of many skills (Perkins, Hancock, Hobbs, Martin, and Simmons, 1986). For example, many novice programmers have poor tracing skills (Lister *et al.* 2004), fail to recognize strategies applied in example programs (Thompson, Whalley, Lister, and Simon, 2006), and seldom apply strategies and plans in their programs (de Raadt, Toleman, and Watson, 2007). Researchers suggested explicitly teaching and assessing strategies and plans (de Raadt, Watson, & Toleman, 2009; Soloway, 1986). However, few mechanisms assist students in reflecting their abilities of strategy planning, implementation, and program tracing. This study proposes a system with open student models based on an assessment framework to assist students in reflecting their proficiencies in iteration programming, including strategies, implementation, and program tracing. Iteration is a basic and vital component of programming to repeatedly execute a set of program codes. Iteration programming includes two main strategies of counter-controlled loop and sentinel-controlled loop and three implementation skills of *while*, *for*, and *do-while*. Open student models indicate that computer assisted learning system opens student models (system assessments of students' knowledge) to students in order to promote reflection, self-assessment, and metacognition (Bull, 2004; Bull and Kay, 2013; Mitrovic and Martin; 2007).

2. A System with Open Student Models on Iteration Programming

The system contains an assessment framework of iteration programming and computer assisted assessment mechanisms to build student models, and interfaces of presenting open student models.

2.1 Assessment Framework of Iteration Programming

Table 1 lists an assessment framework of iteration programming, including strategies, implementation skills, and program tracing ability. Iteration programming consists of two main strategies: counter-controlled loop and sentinel-controlled loop (de Raadt, Watson, & Toleman, 2009; Soloway, 1986). Counter-controlled loop repeats the loop specific times, which is determined by the counter variable. Sentinel-controlled loop repeats the loop until the sentinel value is inputted or is computed. Students' proficient levels of strategies are assessed based on their performance of strategy applications and program tracing ability. Assessments of strategy application include program strategy classification, problem strategy classification, and programming in specific strategy. Program strategy classification assesses whether students correctly classify a program according to the applied strategy in the program. Problem strategy classification assesses whether students correctly classify a problem according to the appropriate strategy for solving the problem. Programming in specific strategy assesses whether students successfully write a program in a specific strategy. Program tracing ability of strategies assesses whether students correctly predict the output of a program in a specific strategy.

Implementation skills of iteration programming include *while*, *for*, and *do-while*. Each iteration strategy could be implemented in *while*, *for*, or *do-while*. Application of implementation skills assesses whether students successfully write a problem in a specific implementation. Program tracing ability of implementation skills assesses whether students correctly predict the output of a program in a specific implementation. A question may involve multiple concepts or skills. For instance, a program output predication question of a program with counter-controlled loop and implemented in *while* involves program tracing abilities of counter-controlled loop and *while* implementation.

Table 1: Assessment framework of iteration programming.

	Application	Program Tracing
Strategies (counter-controlled loop, sentinel-controlled loop)	Program strategy classification Problem strategy classification Programming in specific strategy	Program output prediction
Implementation skills (while, for, do-while)	Programming in specific implementation	Program output prediction

2.2 Computer Assisted Assessment Mechanisms and Open Student Models

Based on the assessment framework, the system supports computer assisted assessment mechanisms to build student models. The system enables teachers to design questions to assess student abilities. The system supports multiple-choice questions and fill-in-blank questions. Multiple-choice questions can be applied to program strategy classification and problem strategy classification. Fill-in-blank questions can be applied to program output predication. Multiple-choice question and fill-in-blank question can be graded by the system. In addition, teachers assign programming assignments to assess programming in specific strategy and specific implementation. The programming assignments are graded by teachers or teacher assistants.

A question or a programming assignment may involve multiple concepts or skills. The system enables teachers to assign involved concepts or skills in each question and assignment and then the system builds a question concept relationship table. The system builds student models by assessing students' proficient levels of each concept and skills based on the question concept relationship table and students' answering records of questions and assignments (Hwang, 2003). The system provides open student models to present students' assessed proficient levels of iteration programming abilities. Students can click at a specific concept or skill to inspect their answering records of related questions and assignments. It helps students reflect and improve their iteration programming abilities.

3. Practical Applications and Future Works

This section presents practical applications and future works of the open student models of iteration programing.

(1) The open student models of iteration programming provide students with a framework for reflecting and improving their iteration programming abilities, including strategies, implementation skills, and program tracing ability. The open student models present system assessments of iteration programming abilities. It helps students reflect their weakness. In addition, the open student models present students' records of question-answering and assignments related to specific iteration programming abilities. It indicates the types of questions and assignments for students to practice to improve specific iteration programming abilities.

(2) Students can be asked to self-assess their proficient levels of programming abilities. The results of self-assessments and system assessments can be compared to develop negotiable open student models. Negotiable open student models promote better self-assessments (Chou *et al.* 2015).

(3) Student proficiencies in strategies, implementation skills, and program tracing ability can be collected for analyzing students' difficulties in learning iteration programming. The analysis can assist teachers in tutoring students and adjusting instruction and assist in developing adaptive tutoring mechanisms for helping students.

(4) Open student models can be used for students to set their goals of improving iteration programming abilities and to trace their improvement.

Acknowledgements

The authors would like to thank the support of the Ministry of Science and Technology, Taiwan (MOST 103-2511-S-155 -003)

References

Bull, S. (2004). Supporting Learning with Open Learner Models, *Proceedings of 4th Hellenic Conference with International Participation: Information and Communication Technologies in Education*, Athens, Greece. Keynote.

Bull, S., & Kay, J. (2013). Open learner models as drivers for metacognitive processes. *International Handbook of Metacognition and Learning Technologies* (pp. 349-365). Springer New York.

Chou, C. Y., Lai, K. R., Chao, P. Y., Lan, C. H., & Chen, T. H. (2015). Negotiation based adaptive learning sequences: Combining adaptivity and adaptability, *Computers & Education*, 88, pp.215-226.

Hwang, G. J. (2003). A conceptual map model for developing intelligent tutoring systems. *Computers & Education*, *40*(3), 217-235.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Mostr¨om, J. E., Sanders, K., Sepp¨al¨a, O., Simon, B. & Thomas. L. (2004) A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, vol. 36, no. 4, pp.119–150.

Mitrovic, A., & Martin, B. (2007). Evaluating the effect of open student models on self-assessment. *International Journal of Artificial Intelligence in Education*, 17 (2), 121-144.

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F. & Simmons. R. (1986). Conditions of learning in novice programmers. *Journal of Educational Computing Research*. vol. 2, no. 1. pp. 37-55.

de Raadt, M., Toleman, M., & Watson, R. (2007). Incorporating programming strategies explicitly into curricula. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88* (pp. 41-52). Australian Computer Society, Inc.

de Raadt, M., Watson, R., & Toleman, M. (2009). Teaching and assessing programming strategies explicitly. *Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95* (pp. 45-54). Australian Computer Society, Inc.

Robin, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion, *Computer Science Education*, *13*(2), 137-172.

Soloway, E. (1986). Learning to Program = Learning to Construct Mechanisms and Explanations, *Communications of the ACM*, 29(9), 850–858.

Thompson, E., Whalley, J., Lister, R. & Simon, B. (2006). Code classification as a learning and assessment exercise for novice programmers. *Proceedings of the 19th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2006)*, pp. 291–298.