Studying problem-solving strategies in the early stages of learning programming

E. CAMBRANES-MARTINEZ
Universidad Autónoma de Yucatán, México
AND
J. GOOD
University of Sussex, United Kingdom

Computer science and engineering undergraduates must develop problem-solving skills, which are particularly important when solving programming problems which use simple algorithms. Ideally these skills are developed in the early stages of learning programming, in particular during the first weeks or months of a student's first course. For complete beginners, problem solving and logical thinking are key to the development of programming skills. This article presents an initial proposal for a study of complete beginner programmers, i.e. students with no prior formal experience of programming. This research is particularly focused on the period in which students learn basic programming concepts to solve problems. The investigation will include the study and analysis of students' interactions with tools and systems during the problem solving process. Data mining techniques will be used to uncover underlying patterns in the interactions. The goal is to exploit these patterns to enhance tutors' teaching processes.

Key Words and Phrases: learning programming, patterns, data mining, programming beginners.

1. INTRODUCTION

Students pursuing a bachelor's degree in computer science or engineering must develop good problem-solving skills if they are to become competent programmers. These skills are best acquired in the early stages of learning programming, in particular during the first weeks or months of a student's first course. Since this competence is essential for the success of future programming courses, the study of the development of this ability is important.

2. RELATED WORK

Most of the research in this field has compared novice and expert programmers, [Bateson et al 1984, du Boulay 1986, Schneiderman 1976, Weiser and Shertz 1983, Webb et al. 1986, Mayer 1987] reporting on a variety of skills and knowledge based on the contrast between these groups. However, in the literature the concept of 'novice programmer' can be ambiguous: in some experiments novice programmers had no experience; while in others they had completed a couple of programming courses. Schneiderman [1976] proposed a classification of programmers in which the lowest category is 'naive', referring to students without any formal instruction in programming. Nowaczyk [1984] argued that knowledge of the approaches or procedures used by skilled programmers could be useful for teaching novice students. Alternatively, tutors could refine and utilise any basic strategies employed by complete beginner programmers.

Authors' addresses: E. Cambranes, Facultad de Matemáticas, Universidad Autónoma de Yucatan, Mexico. E-mail: edgar.cambranes@uady.mx; J. Good, School of Informatics, University of Sussex, United Kingdom. E-mail J.Good@sussex.ac.uk

3. PROPOSAL

In the early stages of learning the basic concepts of structured programming (inputs, outputs, assignments and control flow) a tutor typically uses pseudo code or flow charts. Some tutors prefer flow charts because they involve minimal coding and they also often use interactive tools that can facilitate better comprehension and easier testing. In an experiment conducted with students between 12 and 13 years of age, where interactions were recorded using an interactive tool, Kiesmüller found that it was possible to detect some patterns in the problem-solving process using the chronology of the interactions, [Kiesmüller 2009]. The proposal for this research is based on a study of the problem-solving strategies that first-year undergraduate students use when they solve basic algorithmic problems. It is proposed to use a tool which allows students to create and test flow charts [Rodriguez et al 2009]. This tool will be adapted to record significant interactions during the problem solving process: time taken to complete an exercise, number of blocks used (added or deleted), expressions used in the blocks of the flow chart (conditions or assignments), testing and errors. Every interaction will be recorded with a time stamp to preserve the chronology of the problem-solving process. Based on a sample of 150 students completing 30 algorithmic exercises, a large number of interactions will be recorded. Therefore it will be possible to analyse these interactions using a data mining approach. Initially descriptive data mining techniques will be used to find frequent patterns, associations and classifications. Other patterns can later be extracted using the students' profiles and interactions with other systems such as e-learning platforms. This knowledge will then be used to attempt to improve the didactic practice of tutors when teaching basic programming concepts.

ACKNOWLEDGEMENTS

The authors wish to thank James Jackson for providing valuable comments on this article. This work is supported by Programa de Mejoramiento del Profesorado PROMEP (Mexico).

REFERENCES

BATESON, A. G., ALEXANDER, R., AND MURPHYA, M. 1987. Cognitive processing differences between novice and expert computer programmers. *International Journal of Man-Machine Studies* 26: 649-660.

DU BOULAY, B. 1986. Some Difficulties of Learning to Program. *Journal of Educational Computing Research* 2, no. 1: 57 - 73. http://baywood.metapress.com/link.asp?id=3lfx9rrf67t8uvk9.

KIESMÜLLER, U. 2009. Diagnosing Learners' Problem-Solving Strategies Using Learning Environments with Algorithmic Problems in Secondary Education. *ACM Transactions on Computing Education* 9, no. 3 (September): 1-26. doi:10.1145/1594399.1594402.

NOWACZYC, R. 1984. The relationship of problem-solving ability and course performance among novice programmers. *International Journal of Man-Machine Studies* 21, no. 2 (August): 149-160. doi:10.1016/S0020-7373(84)80064-4. http://dx.doi.org/10.1016/S0020-7373(84)80064-4.

SCHEIDERMAN, B. 1976. Exploratory Experiments in Programmer Behavior. *International Journal of Computer and Information Sciences* 5, no. 2: 123-143. http://www.springerlink.com/content/m852q12168q76224/fulltext.pdf

WEBB, N. M., ENDER, P., AND LEWIS, S. 1986. Problem-Solving Strategies and Group Processes in Small Groups Learning Computer Programming. *American Educational Research Journal* 23, no. 2 (January): 243-261. doi:10.3102/00028312023002243.

WEISER, M., AND SHERTZ, J. 1983. Programming problem representation in novice and expert programmers. *International Journal of Man-Machine Studies* 19, no. 4 (October): 391-398. doi:10.1016/S0020-7373(83)80061-3.

MAYER, R. 1997. From Novice to Expert. In *Handbook of Human-Computer Interaction*, 781-795. Elsevier. doi:10.1016/B978-044481862-1.50099-6. http://dx.doi.org/10.1016/B978-044481862-1.50099-6.

RODRÍGUEZ-CÁMARA, V., KU-QUINTANA J., AND CAMBRANES-MARTÍNEZ E.. 2009. Origami: Tool to support novice programmers based on flowcharts. In *Extended Proceedings of the 4th. Latin-American Conference on Human Computer Interaction*, 88-87. Mérida, Yucatán, México: Universidad Autónoma de Baja California.