Schema Acquisition: Implications for the Instructional Design of Examples

Siti Soraya ABDUL RAHMAN and Benedict DU BOULAY
IDEAS Lab, Department of Informatics, School of Science and Technology
University of Sussex, UK
{s.abdul-rahman, b.du-boulay}@sussex.ac.uk

Keywords. Analogical reasoning, schema acquisition, example-based learning

Students often use analogical reasoning to solve programming problems. The use of examples is one of three types of analogical reasoning in problem solving [1]. Research has shown that examples play an important role in learning and problem solving [2, 3] and are crucial to the acquisition of initial cognitive skills [4]. Schema acquisition is one of the underlying processes in acquiring such skills in learning programming [5].

The concept of example-based learning has received a significant amount of interest from researchers in the programming education domain and they have developed systems to support such learning [i.e. 6-11]. Nevertheless, evidence from worked-example research points out some limitations of example-based learning. Although several of the systems have attempted to address these limitations, various questions remain open. For instance, it is not clear whether these systems, (apart from [7,10]) have been sufficiently evaluated against student learning outcomes including transfer, and more importantly, the relationship between individual learning style and learning outcome or cognitive load effects resulting from using the system needs elucidation. Note that [12] have identified the relationship between working memory capacity and learning styles. Indeed, several empirical findings within programming education literature have pointed out that reflective students perform better than active students in programming performance. Another issue worth exploring is why examples are so seldom used by students [8] and often neglected in programming instruction [5] given the fact that these are an effective way to learn a complex cognitive skill such as problem solving [13]. As a final point, only a limited amount of research on instructional design involving worked-examples has been carried out in the area of programming education [i.e. 7, 10, 11, 14].

In an attempt to improve the effectiveness of worked-examples, [4] suggest three moderating factors. These include intra-example features, inter-example features, and individual differences in example processing [3, 15]. In addition to this work focusing on the instructional principles of worked-examples, recent research is also focusing on techniques to optimise cognitive load for learning from worked-examples, [see 13].

Taking all these aspects into account, the purpose of this research is to bridge the gaps identified above by extending previous research on example-based learning systems with regard to the instructional design of the examples themselves. This can be done by taking into consideration instructional principles from the worked-examples research [4] and by drawing from assumptions laid down within the current developments of Cognitive Load Theory (CLT). The aim of this research is to explore the roles that worked-examples bring into play on novices' analogical problem solving in programming. In particular, the research into effective strategies for learning from worked-examples seeks to promote schema acquisition and transfer. The research investigates the differential effects on the different kinds of cognitive load and transfer performance for three worked-example formats while taking into account student's learning style. Specifically, the research makes the following prediction with regard to the comparison of combined format (the completion strategy [7, 11, 14] with the introduction of structural example-based format [10]) that is designed to improve schema acquisition and transfer with that of the structural example-based format and the completion strategy format. That is, given the same amount of time on task with similar instructional content, the combined format leads to better learning on both active and reflective students than either structural example-based format or completion strategy format. The answer to these questions will advance our knowledge about CLT and will guide us in the practical implications for learning from worked-example in the area of programming instruction. More importantly, it provides preliminary work towards a macro-adaptive system for example-based learning.

References

[1] Reimann, P., Schult, T.J., Turning examples into cases: Acquiring knowledge structures for analogical problem solving, *Educational Psychologist* **31** (1996), 123-132.

[2] Pirolli, P.L., Anderson, J.R., The Role of Learning from Examples in the Acquisition of Recursive Programming Skills, *Canadian Journal of Psychology* **39** (1985), 240-272.

[3] Chi, M.T.H., Bassok, M., Lewis, M.W., Reimann, P., Glaser, R., Self-explanations: How students study and use examples in learning to solve problems, *Cognitive Science: A Multidisciplinary Journal* **13** (1989), 145 – 182.

[4] Atkinson, R.K., Derry S.J., Renkl, A., Wortham, D., Learning from examples: instructional principles from the worked examples research, *Review of Educational Research* **70** (2000), 181-214.

[5] Van Merrienboer, J.J.G., Paas, F.G.W.C., Automation and Schema Acquisition in Learning Elementary Computer Programming: Implications for the Design of Practice, *Computers in Human Behaviour* **6** (1990), 273-289.

[6] Weber, G., Analogies in an Intelligent Programming Environment for Learning LISP, In: Lemut, E., du Boulay, B., Dettori, G. (eds.), *Cognitive Models and Intelligent Environments for Learning Programming*. Springer-Verlag, Berlin (1993), 210-219.

[7] Chang, K-E., Chiao, B-C., Chen, S-W., Hsiao, R-S., A programming learning system for beginners – A completion strategy approach, *IEEE Transaction on Education* **43** (2000), 211-220.

[8] Weber, G., Brusilovsky, P., ELM-ART: An Adaptive Versatile System for Web-based Instruction, *International Journal of Artificial Intelligence in Education* **12** (2001), 351-384.

[9] Brusilovsky, P., *WebEx: Learning from examples in a programming course*, In Proceedings of the World Conference on the WWW and Internet, Orlando, Florida (2001).

[10] Davidovic, A., Warren, J., Trichina, E., Learning benefits of structural example-based adaptive tutoring systems, *IEEE Transactions on Education* **46** (2003), 241-251.

[11] Garner, S., An exploration of how a technology-facilitated part-complete solution method supports the learning of computer programming, *Issues in Informing Science and Information Technology* **4** (2007), 491-501.

[12] Graf, S., Lin, T., Kinshuk, The relationship between learning styles and cognitive traits – Getting additional information for improving student modelling, *Computers in Human Behaviour* **24** (2008), 122-137.

[13] Paas, F., Van Gog, T., Optimising worked example instruction: Different ways to increase germane cognitive load, *Learning and Instruction* **16** (2006), 87-91.

[14] Van Merrienboer, J.J.G., Instructional strategies for teaching computer programming: Interactions with the cognitive style reflection-impulsivity, *Journal of Research on Computing in Education* **23** (1990), 45-53.

[15] Renkl, A., Learning from worked-out examples: A study on individual differences, *Cognitive Science* **21** (1997), 1-29.