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
towards a macro area of strategic behavior and will investigate the introduction of structural example-based format. That is, given the same amount of time on task with the completion strategy format, active and similar instructional content, the combined format leads to better learning on both the completion strategy format 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