Chapter 6
Using Simulation Games in Teaching Formal Methods for Software Development

Štefan Korečko
Technical University of Košice, Slovakia

Ján Sorád
Technical University of Košice, Slovakia

ABSTRACT

Because of the current trend of massification of higher education, motivation of students is a serious issue, especially in courses closely related to mathematics. The ones that undoubtedly belong to this group are courses dealing with formal methods for software development, such as Z notation, B-Method, or VDM. The chapter shows how a customized simulation game can be used to bring a domain typical for utilization of formal methods, the railway domain, to students and thus motivate them to learn these sophisticated ways of software development. By means of two examples, it demonstrates that such a tool, despite its limited scope, can be used to teach a variety of concepts related to formal methods. It also discusses related approaches to teaching formal methods, describes the customized game and its application in teaching, and evaluates experience with the application.

INTRODUCTION

We live in the era of massification of higher education. We encounter not only highly motivated and interested students but also average ones, where didactic methods, usually used on lower types of schools become relevant. All kinds of subjects in university curricula are affected by this situation but maybe the most suffering ones are those closely related to the field of mathematics. And formal methods courses definitely belong to this group.

Formal methods (FM) are rigorous mathematically based techniques for the specification, analysis, development and verification of software and hardware. Rigorous means that a formal method provides a formal language with unambiguously defined syntax and semantics and mathematically based means that some mathematical apparatus (formal logic, set theory, etc.) is used to define the

DOI: 10.4018/978-1-4666-7304-5.ch006

language. But as Cerone, Roggenbach, Schlingloff, Schneider and Shaikh (2013) note, a language is not enough to constitute a formal method. To call it a method, procedures that allow doing something with specifications written in the language have to be present, too. An example of a well-known FM are regular expressions (Cerone et al., 2013): Syntax of its language can be specified by a context-free grammar. For the semantics there are several ways how to define it, for example by specifying corresponding sets of words or constructing a finite automaton that recognizes words satisfying given expression. A procedure can, for example, be a replacement of every word that satisfies given expression by another word. There are many ways how to classify FM and one, especially interesting from the educational point of view, is a taxonomy based on automation of their procedures and on how easy it is to use them. This taxonomy distinguishes between lightweight and heavyweight formal methods. Lightweight formal methods usually do not require deep expertise. The heavyweight ones are more complex, less automatic, but also more finely grained and powerful (Almeida, Frade, Pinto, & de Sousa, 2011). We can say that for a lightweight FM it is enough to learn its language and know what button to hit in corresponding software tool to do this or that. Often it is not even necessary to learn formal semantics of its language, an explanation in a natural language is sufficient. The aforementioned regular expressions are a lightweight FM. To use them for a text search or replacement in a text editor one just has to read few lines in the editor user's manual, write an expression to an appropriate text field and press a button next to it. On the other hand, significant examples of heavyweight FM are those involving theorem proving as a method of software correctness verification. In principle, the theorem proving cannot be fully automated because underlying theories are usually not decidable. So, to prove assertions about a system a human assistance is often required and to be able to assist one has to possess knowledge about the syntax and formal semantics of the language of given FM and operation of its prover. This means a lot of effort but as Harrison (2008) points out, theorem proving brings substantial benefits over other, highly automated, verification methods (e.g. model checking). Provided that properties of a system are correctly specified, its formal verification can ensure that the properties will hold in any state of the system. In an ideal world all software should be like this – 100% verified before its delivery to users. But in reality we use to get faulty software, be it games, operating systems or firmware, and faults are fixed afterwards by means of updates.

As university teachers we sometimes experience resistance from students when a new language or method is introduced, even if it is a widely used one. And position of formal methods courses in software engineering curricula is much worse. Not only are FM too close to the unpopular math but there are not many companies using them in practice. And, especially in the case of the heavyweight ones, we can find them only in specific application areas where their use and cost are justified (Almeida et al., 2011). Of course, we would like to see more widespread utilization of FM and we hope to achieve it by introducing as much students as possible to the art of their application. A big obstacle here is an elective status of many FM courses. So, the essential question is how to motivate students to take FM courses and to stay in them. It is critical to properly choose an application area on which the use of FM will be demonstrated and for which the students will develop something using formal methods. An area where a software fault is able to cause too much damage or loss of lives before any update can be applied. In addition, it should be an area where formal methods have already been successfully applied. According to the comprehensive survey (Woodcock, Larsen, Bicarregui, & Fitzgerald, 2009) and its recent update (Fitzgerald, Bicarregui, Larsen, & Woodcock, 2013) the most of FM industrial success stories can be found in