Learning Programming Patterns Using Games

Alexis Vinicius de Aquino Leal, Education Department, Federal Institute of Mato Grosso, Barra do Garças, Brazil
Deller James Ferreira, Informatics Institute, Federal University of Goiás, Goiânia, Brazil

ABSTRACT

There is still no pedagogy to teach programming that stands out significantly from others and no consensus on what is the best way for learning programming. There is still a need to develop new teaching methods for learning in introductory programming courses. This paper presents a pedagogic approach in support of creativity in programming and the results of a successful case study, where the teacher facilitates the learning of programming patterns by means of students’ involvement in game creation. The results show an improvement in motivation and learning introductory programming, when we combine programming patterns with games using concrete materials.

KEYWORDS

Collaborative Learning, Creativity in Programming, Games, Learning Programming, Programming Patterns

INTRODUCTION

There is still no pedagogy to teach programming that stands out significantly from others and no consensus on what is the best way for learning programming. There is still a need to develop new teaching methods for learning in introductory programming courses. Researches have constantly addressed this issue (Quinson & Oster, 2015; Valerie & Guzdial, 2015; Dasgupta & Resnick, 2014; Kalelioglu & Gulbahar, 2014; Flannery et al., 2013).

Even though creativity is demanded in programming, the number of pedagogical methods addressing creative aspects of programming is low in computer science research. Despite students should be motivated to improve their creative abilities, computer science education researchers and teachers does not address creative aspects very often.

According to Donald E. Knuth, computer programming can be compared to an art form like the creation of music or poetry. This analogy has the intention to make a point of the intrinsic creative aspects of programming. Indeed, computer science experts and apprentices should improve their creativity. This is due to the fact that creativity is very evident and relevant in programming processes. Thus, the development of creativity is an important issue in teaching and learning programming.

Novice programming students show many bottlenecks when programming, such as issues related to the lack of ability of abstraction and lack of the creative ability to deal with problem-solving necessary to create computational solutions.

The two main challenges for students are learning a new programming language and learning how to create a solution for a given computational problem (Soloway, 1986). Considering the first issue, learning the syntax and semantics of a programming language is complex for novice learners. Nevertheless, problem interpretation, solving problems algorithmically and coding an algorithm into a programming language is also challenging for beginner programmers.
Experienced programmers can readily solve a problem, identifying what data types involved, which statements to use, as well as different ways to solve the same problem using distinct strategies. Previous research addressing programming experiences of students during learning programming (Johnson & Soloway, 1985) discuss that experienced programmers are able to solve new problems using prior experience, adapting previous solutions to solve them.

Some studies (Robins et al., 2003; Muller et al., 2004; Soloway & Ehrlich, 1984) show that programming expertise is greatly represented by a knowledge base of pattern-like chunks. Programming patterns are simple design patterns that can help programming novices, because they are solutions to basic recurring algorithmic problems and form the building blocks for the construction of new programs.

Programming patterns are solutions that commonly appear in computational problem solving. Programming patterns reflect programming strategies designed by experts. So, students take advantage of good programming practices by means of the application of programming patterns.

Alexander Christopher (1977) was the first researcher to approach design patterns. He originally defined design patterns as reusable solutions to approach frequently recurring problems in architectural design.

Design patterns have been adopted in various computer science fields such as software engineering, computer science education, human-computer interaction, and e-learning (Kolfschoten et al., 2010). When students apply programming patterns they can “stand on the shoulders of giants”. By means of programming patterns, students can use a repertory of building blocks to exercise their creativity combining and recombining these blocks in an unlimited way. Besides, they know many pieces of solutions becoming able to concentrate more attention in problem solving process.

Some patterns examples are: declare-use pattern, highlights that every programming language utilizes declared identifiers to represent different kinds of entities in a program; encapsulation pattern, brings attention to the need to build and use tool; sequential choices pattern, applied in a situation where each action depends on a single testable condition; flag variable pattern, a flag is a variable you define to have one value until some condition is true, in which case you change the variable’s value, it is a variable you can use to control the flow of a function or statement, allowing you to check for certain conditions while your function progresses; loop with sentinel pattern, used when the programmer needs to repeat a set of actions while a condition is true; counting loop pattern, used when the programmer wants to repeat a set of actions a determined number of times; accumulator pattern, an accumulator is a variable that need to be initiated by zero and be inside a loop to accumulate values; and linear traversal pattern, it is a design process that allows to look at all relevant data items one at a time.

Programming patterns (Polter & Calder, 2004) encompass fundamental structures in which the beginners in computer programming can take advantage to solve a computational problem. Programming patterns form a basis for initial micro solutions that can be creatively adapted and combined by the students during the development of their programs.

A programming pattern describes a solution path to the problem as well as provides information about the context in which it should be applied (Dehnadi, 2009). Thereby, students starting their studies in programming, when faced with a particular kind of problem, can direct their resolution to most appropriate known patterns.

Concerning patterns-based instruction, a problem addressed in novice programmers is the lack of abstraction ability. Haberman and Muller (2008), proposed a teaching method of abstraction using programming patterns. In this study, the authors investigated the influence of the combination of patterns-based instruction with abstract data types. This study showed that patterns-based instruction is promising, however, they noticed that it still had some gaps.

Muller (2005) discusses other issues concerning patterns-based instruction. One issue is the student’s barriers preventing them to recognize similarities between the problems at hand and problems encountered and solved previously. Novice students have difficulty in identifying patterns related to
Hypermedia Modules for Distance Education and Virtual University: Some Examples
Nicoletta Sala (2003). International Journal of Distance Education Technologies (pp. 78-95).
www.igi-global.com/article/hypermedia-modules-distance-education-virtual/1605?camid=4v1a

Chinese Brush Calligraphy Character Retrieval and Learning
www.igi-global.com/article/chinese-brush-calligraphy-character-retrieval/1701?camid=4v1a