Computers in Human Behavior 31 (2014) 558–570

Contents lists available at SciVerse ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

Serious games in tertiary education: A case study concerning the comprehension of basic concepts in computer language implementation courses

Daniel Rodríguez-Cerezo, Antonio Sarasa-Cabezuelo, Mercedes Gómez-Albarrán, José-Luis Sierra *

Dpto. Ingeniería del Software e Inteligencia Artificial, Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain

ARTICLE INFO

Article history:
Available online 11 July 2013

Keywords:
Serious game
Computer Science education
Authoring tool
Learning analytics
Attribute grammar

ABSTRACT

This paper describes *Evaluators*, a system for the development of educational serious games oriented to introductory computer language implementation courses similar to those included in Computer Science tertiary curricula. *Evaluators* lets instructors generate games from collections of exercises addressing basic concepts about the design and implementation of computer languages (in particular, the processing of artificial languages according to the model of attribute grammars). By playing the generated games, students interactively learn the fundamentals of the semantic evaluation process behind attribute grammars. Indeed, they implicitly find solutions to the exercises presented, and they receive immediate feedback about successful and incorrect actions. In addition, the games log students' actions, which can subsequently be analyzed by the instructors using a specialized analytic tool that is included in *Evaluators*. Assessment of the system, which was performed according to three different dimensions (the instructors' perspective, the students' perspective and educational effectiveness perspective), (a) indicates that the exercise-driven approach of *Evaluators* is a cost-effective approach amenable to extrapolation to other areas of Computer Science tertiary education, (b) shows a positive attitude of students toward the serious games built with *Evaluators*, and (c) evidences a positive effect of the system and its pedagogical strategy on long-term student performance.

1. Introduction

Computer language implementation is a subject included in mainstream Computer Science curricular recommendations as a key aspect of a computer scientist's basic education (ACM/IEEE, 2008). Students usually consider it a difficult subject due to its intrinsically abstract nature, which in turn results in a lack of motivation in students compared to more practical topics in the curricula, such as programming technologies (Waite, Jarrahian, Jackson, & Diwan, 2006).

To facilitate the learning process of the students enrolled in a Compiler Construction course at the Complutense University of Madrid (Spain), we decided to use syntax-directed translation (Aho, Lam, Sethi, & Ullman, 2007) as a main paradigm for architecting language processors and *attribute grammars* (Paakki, 1995) as a basic formalism for the specification tasks. Our aim was to make students aware of the importance of clearly separating the specification of the different aspects of a language processor from its subsequent implementation. Based on our experience, the success of this teaching strategy depends heavily on the success that we have in teaching the basic concepts of the formalisms of attribute grammars and their underlying computational model during the early stages of the course. To help our students assimilate the attribute grammar essentials, in the 2009–2010 edition of the course, we created and provided students with batteries of exercises related to the comprehension level of Bloom's taxonomy (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956). Students completed exercises, each consisting of the following components:

- An informal description of a language processing task. This is a free-text description of the language processing problem to be addressed using attribute grammars.
- A formalization of the task by means of an attribute grammar. This is a formal specification using the attribute grammars' formalisms of the processing problem stated in the informal description.
- A sentence in the processed language (e.g., a program or a program fragment if the processed language is a programming language, which is the type of language commonly used in introductory Compiler Construction courses).

* Corresponding author. Tel.: +34 913947548; fax: +34 913947547.
E-mail address: jlsierra@fdi.ucm.es (J.-L. Sierra).

- The parse tree of this sentence, along with the semantic attribute instances whose values should be determined as a result of processing the sentence (e.g., the type of expression or the code generated for this expression if a programming language is considered).

To complete the exercises, students must enumerate an evaluation order for every semantic attribute to explain how the values of these attributes could be calculated in the attributed parse tree provided.

The learning reinforcement provided by solving the exercises proved satisfactory for helping students assimilate fundamental concepts of attribute grammars. For instance, the exercises helped them understand the dependency-driven computation strategy, according to which the evaluation order does not matter as long as the dependencies among attributes are preserved, similarly, for instance, to what happens in a spreadsheet. Unfortunately, however, we also observed that the lack of motivation remained. Indeed, many students found it boring to solve these exercises using pencil and paper. As a consequence, many of them provided incorrect or unfinished solutions, as the students' general tendency was to extrapolate the instructors' solution templates to the new exercises; the predominance of rote learning was alarming. Therefore, we considered creating more effective methods for overcoming the deficiencies detected.

Taking into account that serious games are a growing trend in the field of e-learning, as well as the significant success of this type of environment in recent years (Amory et al.,1999; de Freitas & Liarokapis, 2011; Gee, 2003; Jiménez-Díaz, Gómez-Albarrán, & González-Calero, 2007; Minua, Andreas, & Lakhmi, 2011), we decided to develop *Evaluators*, an educational system that lets instructors easily generate serious games following an exercise-driven approach. The serious games in *Evaluators* are generated from batteries of exercises concerning basic concepts of attribute grammars of the type described above. These exercises are developed using the authoring tool provided in *Evaluators*. The games generated involve students in interactive simulations of the semantic evaluation processes behind attribute grammars. Students determine the correct evaluation order for semantic attributes, and they receive immediate feedback on both successful and incorrect actions. Student actions are logged and can later be analyzed by the instructors using a specialized analytic tool included in *Evaluators*. *Evaluators* benefits from our experience in the development of educational simulations and games (Jiménez-Díaz, González-Calero, & Gómez-Albarrán, 2012; Jiménez-Díaz et al., 2007) and is an evolution of our previous work in the development of specialized tools related to the learning of language processing (Sierra, Fernández-Pampillón, & Fernández-Valmayor, 2008), in which students had to provide solutions to processing problems that were similar to the ones used in *Evaluators* but focused on writing specifications instead of comprehending them.

The structure of the rest of the paper is as follows: Section 2 presents several works that are related to this project; Section 3 provides an in-depth description of *Evaluators*; Section 4 reports some results of our assessment of the system; and Section 5 presents the conclusions and future work.

2. Related work

In this section, we summarize some works related to ours: pedagogical approaches followed in Compiler Construction courses (Section 2.1), visualization and simulation tools used in Computer Science education (Section 2.2), educational tools for enhancing the teaching and learning of concepts related to Compiler Construction (Section 2.3), and game-based teaching and learning of Computer Science topics (Section 2.4).

2.1. Teaching of Compiler Construction courses

There are different strategies for facilitating the assimilation of concepts in Compiler Construction courses. These strategies can be roughly grouped in the following categories:

- *Implementation of a processor for a small programming language*. This is the approach that is most widely used by instructors. In this approach, instructors describe a small programming language with a couple of basic data types, basic operations for these data types, control structures (such as loops and conditionals) and an abstraction mechanism. The students have to implement a compiler for the programming language. There are different prototypical languages used for this strategy, such as COOL (Aiken, 1996), MINIML (Baldwin, 2003) and CHIRP (Xu & Fred, 2006). Other less conventional proposals also exist, such as those in which students use languages for graph representation (Werner, 2003), simple figure drawing (Ruckert, 2007) or robot action programming (Xu & Fred, 2006). The main advantage of these languages is the motivational component that they possess.
- *Small language processing projects*. This strategy aims to solve two major problems based on the implementation of a whole language processor: the tangible possibility that students get stuck in the first phases of development and cannot accomplish the project on time and the need for mastering concepts for developing the language processor that are not taught until later in the course. For this purpose, using small projects focuses students on the concepts covered in class. Thus, as the course progresses, students are able to immediately apply the concepts studied in class. The works by Ledgard (1971) and Shapiro and Mickunas (1976) provide an in-depth discussion of this strategy.
- *Analysis and debugging of processors for real programming languages*. The main idea behind this strategy is that the implementation of any language compiler (a small one or small languages projects) is not sufficient to teach all the concepts involved in the development of real programming language processors. Thus, the work of White, Sen, and Stewart (2005) proposes teaching language processor internals by debugging the source code of real-world compilers. The instructors, by using breakpoints and monitoring the values of certain variables, can focus on different aspects of the processing and drive the sessions according to the concepts taught in the lectures.

While these methodologies are mainly focused on the implementation aspects of language processors, we have realized from our experience in teaching this topic at the Complutense University of Madrid that, in addition to emphasizing implementation, it is necessary to emphasize specification aspects, as the specification of language processors is a critical concern throughout the whole development process. Thus, strategies strongly geared toward implementation aspects are not capable of successfully helping students understand the formal specifications taught in lectures. Our main aim with *Evaluators* is to palliate this inconvenience from the beginning, teaching the basic concepts of attribute grammars, which is a specification formalism that fits well with the syntax-directed translation paradigm that we encourage at the Complutense University of Madrid.

2.2. Visualization and simulation tools for teaching Computer Science

The main objective of the serious games generated with *Evaluators* is to assist students of Compiler Construction in their learning by immersing them in the semantic evaluation process involved in attribute grammars. To this end, *Evaluators* games are similar to