Chapter 13

Scripting Environments of Gamified Learning Management Systems for Programming Education

Jakub Swacha
University of Szczecin, Poland

ABSTRACT

This chapter is devoted to the scripting environments that are pivotal for the development and usage of gamified Learning Management Systems in programming education. They are classified into four types, which include: the student’s code execution environment, the scripting wrapper for processing input and output of the code execution environment, the game-based rule processing environment, and the environment for processing scripts embedded in the course content. The chapter describes the functionality of each of these environments, with exemplary usage scenarios provided. Also, the key requirements for each environment are specified, and adequate design and technological solutions suggested. For open design decisions, consequences of choosing particular options are investigated.

INTRODUCTION

The terms ‘script’ and ‘scripting’ in educational contexts, and especially computer-supported educational contexts, are most often used referring to computer-supported collaboration scripts (see e.g. Fischer et al., 2007). The focus of this chapter, however, is on the computer programming scripts that facilitate various functions of gamified Learning Management Systems used for computer programming education. A ‘script’ is therefore understood here as a program, written for and run in an environment of a specific software system (e.g. a Learning Management System), whose role is to automate the execution of tasks within that system, which would otherwise have to be executed by a human (e.g. a course administrator or an instructor). Although scripts have access to (usually selected) functions of the software system they are written for and data stored therein, they are not an integral part of the system: they should rather be...
considered as a specific type of content processed by the system. A scripting environment is a part of
the software system dedicated to execution of scripts, keeping their runtime data and controlling their
access to functions and data of the system itself. With a number of distinct system areas that can be
scripted for various purposes, a single software system can include more than one scripting environment.

The architecture of gamified Learning Management Systems proposed by this author (Swacha, 2014)
considers scripting environments as its key components, providing various rationale for their inclusion
(e.g., processing gamification rules, automatic assessment of exercise solutions, and providing dynamism
or interactivity to the embedded educational resources). It, however, does not go down to the details
of how they should be implemented, more than differentiating scripts that are supposed to be executed
server- and client-side.

In fact, although the design and implementation aspects of such scripting environments are of pri-
mary interest for researchers and practitioners interested in constructing gamified computer-supported
programming education environments, this topic has not yet been investigated in the existing literature,
as presented in the next section.

The aim of this chapter is therefore to cover this gap by:

• Identifying the four primary scripting environments for gamified Learning Management Systems
  in the programming education context,
• Explaining the specific role of theirs, illustrated with exemplary usage scenarios, and specifying
  key requirements for the respective scripting environments,
• Suggesting appropriate design and technological solutions capable of meeting the specified
  requirements,
• Formulating the key design decisions and investigating their consequences, including ensuing
  problems.

BACKGROUND

The effective teaching of computer programming can hardly be realized without providing the students
the ability to try the code they write as an exercise and check whether it is syntactically and semanti-
cally correct. Although the check for the syntax correctness of a program is typically a prerequisite for
its execution, checking its semantic correctness often requires well-planned tests and the knowledge of
the problem, solution method and the programming language that often only the instructor (i.e. not the
student) has.

Automating the verification of students’ programs brings a number of benefits, as not only the instruc-
tor is freed from much of his/her duties, but also the students can proceed with their own pace, getting
an appropriate and instant feedback, whenever they fail or succeed. It was even found to yield superior
educational results (see Fernandez, 2011).

Such feature can only be achieved by providing a code execution environment capable of safely
running programs submitted by students. For the sake of consistency with the other environments to
be described later, the code execution environment can also be called a scripting environment, as the
students’ exercise programs can be considered a specific form of scripts.

The code execution environment must be able to execute scripts in the language being taught in the
course – or at least, a subset of it. Note also that the possible ability of the code execution environment
Related Content

Online Prevention and Early Intervention in the Field of Psychiatry Using Gamification in Internet Interventions: Practical Experience with a Program Targeting Eating Disorders in Czech Republic

Effects of the Digital Game-Development Approach on Elementary School Students' Learning Motivation, Problem Solving, and Learning Achievement

MMORPG Roles, Civic Participation and Leadership Among Generation Y
[www.igi-global.com/article/mmorpg-roles-civic-participation-leadership/66072?camid=4v1a](www.igi-global.com/article/mmorpg-roles-civic-participation-leadership/66072?camid=4v1a)

Moves in Mind: The Psychology of Board Games
[www.igi-global.com/article/moves-mind-psychology-board-games/74796?camid=4v1a](www.igi-global.com/article/moves-mind-psychology-board-games/74796?camid=4v1a)