Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 159 (2019) 2502–2511

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Defining Requirements for a Gamified Programming Exercises Format

Jakub Swacha[a,*], Ricardo Queirós[b,c], José Carlos Paiva[b], José Paulo Leal[b,d]

[a]*Institute of Information Technology in Management, University of Szczecin, Szczecin, Poland*
[b]*CRACS - INESC TEC, Porto, Portugal*
[c]*ESMAD, Polytechnic of Porto, Porto, Portugal*
[d]*DCC - FCUP, University of Porto, Porto, Portugal*

Abstract

Computer programming is a complex domain both to teach and learn. This incited endeavors to find methods that could mitigate at least some of the existing barriers. In the last years, automatic assessment has been playing an important role in reducing the burden of teachers in the assessment of students' attempts to solve programming exercises and fostering the autonomy of students by allowing them to practice in any place and at any time with timely feedback.

Even more recent development is the use of gamification in computer programming education in order to raise the enjoyment and engagement of students. Despite its rising spread, until now, there is not a programming exercise specification format addressing the needs of gamification, such as the definition of challenges, the underlying storyline, including the links to other exercises, or the rewards for solving challenges in form of points, badges or virtual items. Such a data format would allow the exchange of ready-to-use programming exercises along with the gamification-related data among different educational institutions and courses, providing instructors a possibility to make use of gamification in their courses without having to invest their own time in defining gamification rules themselves.

In this paper, we analyze a set of concepts related to programming gamification developed in our previous work to identify the requirements for the specification of a gamified exercise format.

Keywords: gamification; computer programming; e-learning; interoperability; format requirements

1. Introduction

The combined use of automated assessment, which provides fast feedback to the students experimenting with their code, and gamification, which provides additional motivation for the students to intensify their learning effort, can

* Corresponding author. Tel.: +48-91-444-1908; fax: +48-91-444-2127.
E-mail address: jakubs@uoo.univ.szczecin.pl

help students to overcome the barrier of difficulty in acquiring computer programming skills. Learning programming relies on practice. While there is a number of open software and programming exercise collections supporting automated assessment, so far there are no available public collections of gamified programming exercises, neither an open interactive programming learning environment that would support such exercises, nor a standard for the representation of such exercises, so that they could be developed and shared among different educational institutions.

1.1. Framework for Gamified Programming Education Project

In order to address the gap consisting of the lack of open collections of reusable gamified programming exercises, four elements are needed: (1) a frame of reference for programming course gamification (including featured gamification concepts and the intended area of their application); (2) a specification of a format for exchanging gamified programming exercises based on the above frame of reference; (3) tools for authoring exercises in the above format; (4) a programming learning environment allowing to set up and manage gamified programming courses making use of such exercises. These four elements constitute a framework for the application of gamification to programming education developed within the Framework for Gamified Programming Education (FGPE) project under the Erasmus+ programme. For the first element, the project consortium developed a catalogue of gamification techniques tailored for programming education [2].

In this paper, we extend this work by identifying the requirements for the specification of a format for gamified programming exercises that are indispensable for it to cover all the information needed to make these gamification techniques feasible. Identifying such requirements is necessary to define the scope and form of data to be handled by the format under development. We are aware, though, that gamification-related data, being undoubtedly the key component for this format, are not the only component that has to be specified. For this reason, while keeping the focus on the gamification-related aspects of the format, we also present its three-tier architecture, consisting of gamification, exercise, and organizational tiers, and provide basic information on the remaining two tiers.

The remainder of this paper is organized as follows. Section 2 discusses the relevant work on formats for programming exercises. Section 3 introduces the three-tier architecture on which the data exchange format for gamified programming exercises is based and explains the role of each of the tiers. Section 4 is the core of the paper; in it, based on the gamification techniques defined in our previous work [2], we identify the requirements that the envisaged format has to meet to address each of them respectively. Finally, Section 5 summarizes the main contributions of this work and presents the perspective for future research.

2. Programming Exercises Formats

The increasing popularity of programming contests worldwide resulted in the creation of several contest management systems supported by repositories of programming exercises. At the same time, computer science courses use programming exercises to encourage the practice of programming. As a consequence, each tool uses a self-established format for storing exercises, without adhering to a common format which hinders the sharing of these problems. Hence, the interoperability between these systems is a topic of great interest in the scientific community, resulting in the development of several proposals for a common programming exercise format. The next subsections detail six formats: FreeProblemSet, Peach Exchange Format, Kattis Problem Format, Mooshak Exchange Format, SIPE, and PExIL. Then, their features are synthesized based on a specific exercises format expressiveness model [9].

2.1. FreeProblemSet

FreeProblemSet (FPS) [3] is a standard for ACM-ICPC contest problem storage serialized in XML format. It aims to provide free problem sets for Online Judges managers by transporting data from one judge to another. The format uses XML to describe a programming problem and Document Type Definition (DTD) to formalize it. It includes information on the problem, test data, special judger (optional), and answer (optional). Currently, the FPS format is supported by several popular online judge systems including HUSTOJ, ACM-Server4, and Woj-land (Online Judge from Wuhan University).

2.2. *Peach Exchange Format*

Peach [10] is a system for the presentation, collection, storage, management, and evaluation (automated and/or manual) of assignments. The Peach Exchange Format (PEF) [12] is a specific format for programming task packages used in Peach. Peach task packages are stored in a directory tree with a predefined structure. Currently, Peach is being used by the Eindhoven University of Technology.

2.3. *Kattis Problem Format*

Kattis [4] is a free problem-set with hundreds of problems formalized with the Kattis Problem Format (KPF). The problem format is a directory with a top-level file problem.yaml and a number of subdirectories (and sub-subdirectories). The contents include test data, validators which evaluate the correctness of a program's input-output, problem statement and correct (and incorrect) submissions. In order to reference all these files, a mandatory file called problem.yaml is included with metadata about the problem, such as authorship, license, judging flags, etc.

2.4. *Mooshak Exchange Format*

Mooshak [6] is a web-based competitive learning system originally developed for managing programming contests over the Internet [7]. Despite the context where it is used, Mooshak has its own internal format to describe problems called Mooshak Exchange Format (MEF). MEF includes an XML manifest file referring several types of resources, such as problem statements (e.g., PDF or HTML), image files, input/output test files, correctors (static and dynamic), and solution programs. The manifest also allows the inclusion of feedback messages and points associated with each test. Currently, Mooshak is being used in several Universities worldwide to support learning activities. In the competitive context, it was used as the official evaluation system for several international programming contests.

2.5. *SIPE*

SIPE (Specification of Interactive Programming Exercises) [11] is a lightweight markdown-based format for small interactive programming exercises. While it provides extensive functionality for code and output checks, it lacks many features heavier formats have such as the definition of special correctors and run-time requirements.

2.6. *PExIL*

PExIL (Programming Exercises Interoperability Language) is an XML dialect that aims to consolidate all the data required in the programming exercise life-cycle [8]. This definition is formalized through the creation of an XML Schema organized in three groups of elements:

- Textual – elements with general information about the exercise to be presented to the learner. (e.g., title, date, challenge);
- Specification – elements with a set of restrictions that can be used for generating specialized resources (e.g., test cases, feedback);
- Programs – elements with references to programs as external resources (e.g., solution program, correctors) and metadata about those resources (e.g., compilation, execution line, hints).

2.7. *Comparison of formats*

The evaluation of the expressiveness of programming assignments formats is already tackled in several works [1, 5, 12]. This subsection synthesizes the formats described previously according to the model proposed by Verhoeff. This model describes conceptually the notion of a task package as a unit for collecting, storing, archiving, and exchanging all information concerning with a programming task. The choice of the Verhoeff model over the alternatives is due to its more comprehensive coverage of the required features. This model organizes the programming exercise data into five facets:

1. Textual information - programming task human readable texts;
2. Data files - source files and test data;
3. Configuration and recommendation parameters - resource limits;
4. Tools - generic and task-specific tools;
5. Metadata - data to foster the exercises discovery among systems.

Table 1 compares all the investigated formats based on these five facets.

Table 1. Expressiveness of programming assignments formats.

Facet	Feature	FPS	PEF	KPF	MEF	SIPE	PExIL
Textual	Multilingual		X		X	X	X
	HTML format	X	X	X	X	X	X
	$\LaTeX$format				X		
	Image	X	X	X	X	X	X
	Attach files		X				
	Description	X	X	X	X	X	X
Data files	Solution	X	X	X	X		X
	Skeleton					X	X
	Multi-language	X	X	X			X
	Tests	X	X	X	X	X	X
	Test groups		X				X
	Sample tests	X					
	Grading		X	X	X		X
	Feedback				X	X	X
Configuration & recommendation	Compiler						X
	Executer						X
	Memory limit	X	X	X			X
	Size limit					X	X
	Time limit	X		X			X
	Code lines					X	X
Tools	Compiler		X				X
	Test gen.						X
	Feedback gen.				X	X	X
	Checker		X			X	X
	Corrector				X		X
	Library		X		X		X
Metadata	Exercise	X	X	X	X	X	X
	Author		X			X	X
	Event	X	X				X
	Keywords		X		X		X
	Platform		X				
	Management		X				

This study confirms the disparity of programming exercise formats, highlighting both their differences and similarities. This heterogeneity hinders the interoperability among the typical systems featuring the automatic evaluation of exercises. From all these options, PExIL, when compared with the other formats, is the format that covers more features of all the facets. Still, even PExIL has no support for gamification.

3. Format for Gamified Programming Exercises

3.1. Architecture

Although gamification-related data are central to the gamified programming education framework, they are not the only data that the envisaged format for gamified programming exercises has to cover. There is also data related to the organization and logical and conceptual sequencing of resources at the level of a course and/or even an educational institution, as well as data defining programming exercises in their full life-cycle (creation, selection, presentation, solving, and evaluation) – notwithstanding the gamification.

The architecture of the proposed format, depicted in Fig. 1, defines three separate data tiers: organizational, exercise, and gamification. The aim of such a distinction is the complete separation of the design of the gamification aspect from the definition of programming exercises as well as from their placement in a course provided by an educational institution. This way the exercises can be used without the gamification if needed, any layer can be modified separately from the others, and existing tools and formats can be used to handle the first two tiers (organizational and exercise) without the need of any adaptation.

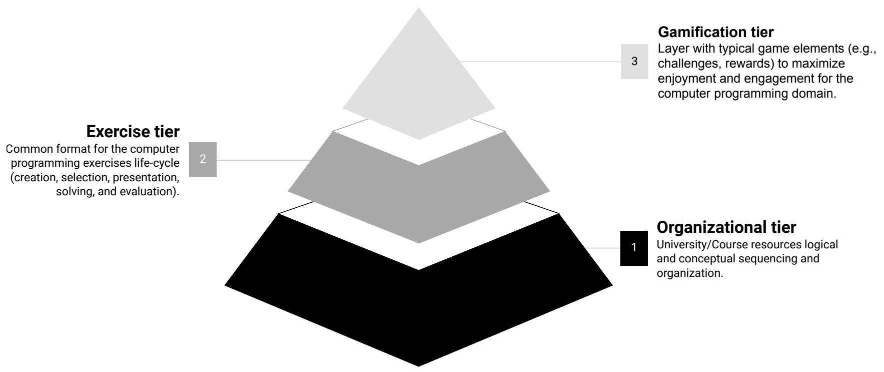
Gamification tier
3
Layer with typical game elements (e.g., challenges, rewards) to maximize enjoyment and engagement for the computer programming domain.
Exercise tier
2
Common format for the computer programming exercises life-cycle (creation, selection, presentation, solving, and evaluation).
Organizational tier
1
University/Course resources logical and conceptual sequencing and organization.

Fig. 1. Framework three-tier architecture

3.2. Tiers

The **Organizational tier** covers all the aspects of the organization of exercises in the course, and its integration in the learning environment (e.g., a learning management system) used by an educational institution. Instructors use it to order the content of their courses, regardless of what the content actually is.

The **Exercise tier** defines the specification for programming exercises, regardless of whether they are gamified or not. It should address the needs of all the phases of its intrinsic life-cycle (Fig. 2), which comprises several phases as follows. In the **creation phase**, the content author should have the means to automatically create some of the resources (assets) related to the programming exercise, such as the exercise description and test cases, and the possibility to package and distribute them in a standard format across all the compatible systems (e.g., learning management systems, learning objects repositories). In the **selection phase**, the teacher must be able to find a programming exercise based on its metadata in a repository of learning objects and store a reference to it in a learning management system. In the **presentation phase**, the student must be able to choose the exercise description in its native language and a proper format (e.g., HTML or PDF). In the **solving phase**, the learner should have the possibility to use test cases to test his/her attempt to solve the exercise as well as the possibility to automatically generate new ones. In the **evaluation phase**, the evaluation engine should receive specialized metadata to properly evaluate the learner's attempt and return relevant feedback to help them to correct the code and resubmit it.

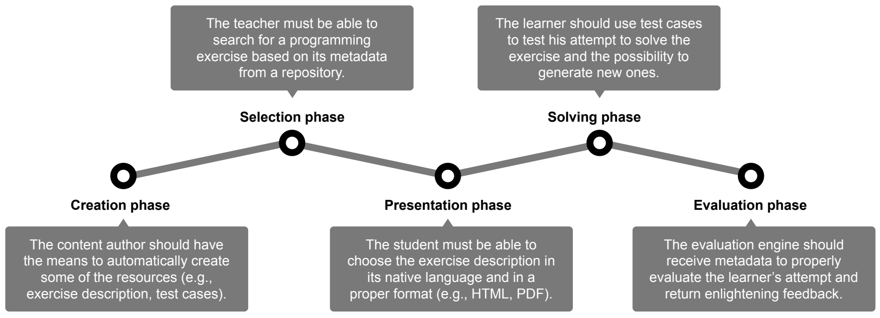
The teacher must be able to search for a programming exercise based on its metadata from a repository.
The learner should use test cases to test his attempt to solve the exercise and the possibility to generate new ones.
Selection phase
Solving phase
Creation phase
Presentation phase
Evaluation phase
The content author should have the means to automatically create some of the resources (e.g., exercise description, test cases).
The student must be able to choose the exercise description in its native language and in a proper format (e.g., HTML, PDF).
The evaluation engine should receive metadata to properly evaluate the learner's attempt and return enlightening feedback.

Fig. 2. Programming exercises life-cycle

The **Gamification tier** covers all the elements specific to gamification that should be included to foster the motivation and enjoyment of students during the realization of the course (e.g., challenges and rewards). This is the main novelty element, as while existing formats can be effectively used for the two other tiers, no existing programming exercise format provides the capability to store an adequate scope of gamification-related data.

The gamification tier has to convey data about gamification concepts implemented in programming exercises. In our previous work [2], we defined a set of gamification concepts relevant to programming education, composed of three parts: basic gamification concepts (related to course organization, goals set for students, rewards for achieving these goals, and conditions that should be considered in such cases), exercise-level concepts (defining types and subtypes of programming exercises and specific rewards applicable for solving a single exercise), and course-level concepts (defining challenges traversing over multiple exercises and specific rewards for solving them).

In the next section, we iterate over these concepts, and for each concept, we identify the requirements that the envisaged format has to meet to address it adequately.

4. Requirements for Specification of Gamified Programming Exercises

This section presents the results of the identification of the necessary gamification-related requirements for the envisaged format based on the gamification concepts and techniques previously chosen as appropriate for programming education [2]. Subsection 4.1 covers requirements stemming from general gamification concepts. Subsection 4.2 comprehends requirements stemming from gamification concepts on the exercise level. Subsection 4.3 comprises requirements stemming from gamification concepts on the course level.

4.1. *Basic Gamification Concepts*

Basic gamification concepts comprise common concepts applied both to exercise and course levels, and abstract concepts on which more specific concepts are based (at the exercise or course level). For instance, this group includes concepts related to the organization of the course, concepts related to types of rewards (not specific rewards), among others.

Table 2 lists requirements posed by the organization of the course. These include metadata pertaining to gamified exercise, less often gamified courses, such as its affiliation, content type, and difficulty. Table 3 presents requirements pertaining to the various types of goals that could be defined for the students taking part in a gamified programming course, according to [2].

Table 2. Format requirements of gamification concepts related to course organization.

Concept	**Description**	**Format Requirements**
Course Module	A subset of course content related to a specific topic. Further modules may be locked by default.	Exercise definition has a field specifying the module(s) it belongs to.
Exercise Type	A kind of programming exercise to be solved.	Exercise definition has a field specifying the type.
Exercise Mode	Non-default exercise modes define special guidelines regarding how an exercise should be presented or provide modified requirements to make the student interested in solving an exercise again.	Exercise definition has a field specifying its mode and associated parameters.
Locked Content	Player is aware of locked content but cannot access it before certain requirements are met.	Exercise definition has a field specifying whether it is initially locked.
Secret	A hidden content whose existence is unknown to the player beforehand, only after making certain defined steps. Its disclosure may be random or fixed, depending on certain conditions.	Exercise definition has a field specifying whether it is initially hidden.
Difficulty Level	A subset of course content on a similar difficulty level. Higher levels may be locked by default.	Both course and exercise definitions have fields specifying their difficulty levels.

Table 3. Format requirements of gamification concepts related to goals definition.

Concept	**Description**	**Format Requirements**
Challenge	A single programming exercise to be solved.	The basic unit of course content organization is an exercise.
Requirements	Additional conditions that make it more difficult to pass a challenge.	A field in exercise definition specifying requirements to solve it.
Quest	A set of related programming exercises to be solved.	Course definition includes quest specification (including a set of requirements and rewards).
Streak	A sequence of time units in which certain goals were consistently achieved by the player.	Course definition includes streak specification (including a set of requirements and rewards).
Record	The highest value of a certain metric achieved by this or any player before.	Not content-specific, hence no requirements on the format.

There are several types of rewards envisaged for students who achieved goals. Each of these has different traits and forms of use. For instance, Badge is a visual indicator of a specific accomplishment which, since awarding, can be presented in the student's profile for others to see, whereas Content unlock is an event after which the student gains access to previously inaccessible content. Table 4 presents the requirements for the format pertaining to the respective types of rewards. Rewards are granted when a course participant meets certain conditions towards an exercise or a course. These conditions are defined according to metrics collected during students' interaction with the learning environment (the catalogue of available metrics relevant to, respectively, exercises and courses is given in Tables 4.5 and 5.3 of [2]). Table 5 presents the requirements for the format pertaining to the respective condition types.

Table 4. Format requirements of gamification concepts related to definition of rewards.

Concept	Description	Format Requirements
Point	Increases player's score, showing their progress. There may be different types of points.	Reward item specifies the kind of points. Reward value specifies the number of points.
Level	Depends on the player score, and possibly on other factors. Higher levels may be required to access certain course areas or difficulty levels.	Course definition includes rules for level progression and possible locks for difficulty levels based on player level.
Held Record	The fact that the highest value of a certain metric till now has been achieved by this player. Visible to other players.	This concept is not content-specific and places no requirements on the format.
Current Rank	The current rank of this player in their group leaderboard. Visible to other players (possibly with limitations, e.g., only the top of the leaderboard and the neighbours).	This concept is not content-specific and places no requirements on the format.
Badge	Graphics certifying player's achievement. A single type of badge may have levels.	Reward item specifies the kind of badge. Reward value specifies the level of badge (if applicable).
Virtual Item	Received for certain achievements. In contrast to badge, it can possibly be lost, traded, combined, or used for certain purposes; it also does not have to be visible to others.	Reward item specifies the kind of virtual item. Reward value specifies the number of items.
Coupon	Received for certain achievements. In contrast to Virtual Item, it is intended only for practical purposes, not a kind of collectible: it is only for a single-use for certain purpose (e.g., getting a hint or unlocking a course area). It may (and usually should) have a defined validity period.	Reward item specifies the kind of coupon. Reward value specifies the number of coupons.
Content Discovery	Reveal secret content which was hidden beforehand.	Reward item specifies what should be revealed.
Content Unlock	Unlock access to some content. Note: unlocking can be direct, i.e., granted as a reward for completing e.g., a course area, or indirect, by granting coupons or items that can be used to unlock content (possibly chosen by player), or as a result of achieving a certain level.	Reward item specifies what should be unlocked.
Hint	Text (possibly also reveals a snippet of a correct solution) displayed to a player failing to solve a challenge: on his/her own choice (possibly paid with specific coupons) or automatically after a specific kind of a failure.	Exercise definition has a field specifying hint messages and conditions for displaying them.
Congratulations	Text (possibly accompanied with visual and/or sound effects) congratulating player on achievements.	Both exercise and course definitions have a field for specifying a message to be given on completion of each, respectively. Reward definition has a field specifying congratulations message for receiving it (if applicable).

Table 5. Format requirements of the types of conditions upon which rewards are granted.

Concept	Description	Format Requirements
Attempt	Something player tries to achieve (e.g., opens an exercise).	Exercise definition has a field specifying possible rewards for attempts (number of attempts is a possible condition).
Achievement	Something player achieves (e.g., solves an exercise or completes a quest).	Exercise definition has a field specifying possible rewards for achievements (solution metrics are possible conditions).
Failure	Something player fails to achieve.	Exercise definition has a field specifying possible rewards for failures (number of failures is a possible condition).
Progress threshold	Player passes a specified threshold of a given progress metric.	Course definition has a field specifying possible progress rewards (relevant metric and its threshold level are necessary parameters).
Progress in competition	Player passes another player in a given progress metric.	Course definition has a list of metrics for which a leaderboard should be maintained.

4.2. *Exercise-level Gamification Concepts*

Exercise-level gamification concepts are those that may only be applied in the context of a single exercise. A primary such concept is the type of a programming exercise. Not every programming exercise requires a student to code a complete solution neither to code at all. For instance, some exercises may challenge the student to fill-in gaps in provided source code or just spot bugs in a block of code. Table 6 lists the requirements set by distinct kinds of exercises to be covered by the format.

Table 6. Format requirements of different types of gamified programming exercises.

Concept	Description	Format Requirements
Blank sheet	This kind of exercise provides a blank sheet for the student to write his/her solution source code from the scratch.	Exercise definition has a field specifying the goal to be attained.
Code extension	This kind of exercise provides partially finished solution source code (the provided parts are not subject to change by the student) which the student has to complete.	Exercise definition has a field specifying the goal to be attained. Exercise definition has a field specifying the initial code.
Code improvement	This kind of exercise provides correct initial source code which does not yet achieve all the goals specified in the exercise specification, so the student has to modify it to solve the exercise.	Exercise definition has a field specifying the goal to be attained. Exercise definition has a field specifying the initial code.
Buggy code	This kind of exercise provides code with bugs (and failed tests) to foster the student to find the right code.	Exercise definition has a field specifying the goal to be attained. Exercise definition has a field specifying the initial code.
Fill-in the gap	This kind of exercise provides code with missing parts and asks students to fill them with the right code.	Exercise definition has a field specifying the goal to be attained. Exercise definition has a field specifying the initial code with the snippets to be presented as gaps being marked up.
Mixed code	This kind of exercise breaks a solution into several blocks of code, mixes them, and asks students to sort them (e.g., quicksort algorithm).	Exercise definition has a field specifying the goal to be attained. Exercise definition has a field specifying the initial code with the blocks marked up.
Show me	This kind of exercise defines a small set of primitives that can be used to solve the challenge (e.g., `move('red', 'left')` meaning move red block to left stack) and provides a visual animation of the code execution.	Exercise definition has a field specifying the goal to be attained. Exercise definition has an interactive animation attached/linked and a list of instructions available to the student to solve the exercise.
Spot the bug	This kind of exercise provides code with bugs and asks students to merely indicate the location of the bugs.	Exercise definition has a field specifying the goal to be attained. Exercise definition has a field specifying the initial code with the buggy snippets being marked up.

Exercises may also be gamified by providing alternative requirements for solving them (this also allows to reuse exercises in the same course). For instance, we can limit the time to solve the exercise, ask the student to reduce code size or execution time below a specific threshold, among many others. Table 7 presents the requirements needed for specification of different exercise modes.

Table 7. Format requirements of gamified programming exercise modes.

Concept	Description	Format Requirements
Shapeshifter	This kind of exercise is composed by a set of very similar exercises which switch between them when the metamorphosis timer end. This motivates students to solve the exercise fast but can also help if they are struggling with minor issues less related with what we want them to learn.	Exercise definition has a list of alternative goal specifications and field for the time of change parameter.
Shortening challenge	This kind of exercise provides a bonus reward for shortening the submitted solution to below specified number of lines offered if the submitted solution is correct but exceeds some threshold number of lines.	Exercise definition has a field for the threshold number of lines.
Speedup challenge	This kind of exercise provides a bonus reward for speeding up the submitted solution to below specified execution time, offered if the submitted solution is correct but exceeds some threshold execution time.	Exercise definition has a field for the threshold execution time (in relative units to some reference code execution time, so that it could be scaled depending on the machine).
Hack the problem	This kind of exercise rewards students who solve the exercise in a tricky way without prior information about that.	Exercise definition has a field for the trick detection code.
Time bomb	This kind of exercise is only available for a certain amount of time once revealed.	Exercise definition has a field for specifying the amount of time it is available.

Students who demonstrate "good" behaviors while solving an exercise may be awarded a badge, to foster similar behaviors in the future. Table 8 details the format requirements pertaining to badges granted for specific exercises.

Table 8. Format requirements for defining badges granted on exercise-level.

Concept	Description	Format Requirements
Hardworker	A badge awarded when a student fails more than *n* times to solve an exercise, but ends up solving it.	Reward requirement can refer to the number of attempts submitted before one is accepted.
Scientist	A badge awarded when the student makes several tests to check his solution, before submitting.	Reward requirement can refer to the number of tests run before submitting the solution.
Keyword	A badge awarded when the student uses a specific keyword (e.g., array.map).	Reward requirement can refer to the existence of certain statements in the solution source code.
Straight	A badge awarded when the submitted solution has less or equal number of loops, for instance, than the specified correct solution.	Reward requirement can refer to the number of appearance of certain statements in the solution source code.

4.3. *Course-level Gamification Concepts*

Gamification concepts on the course level may only be applied to series of exercises (not single ones) and/or whole courses. As such they may be used to reward consistency (as a student has to pass through a number of exercises to achieve a defined goal), but also link the exercises logically (with a story explaining the context) and provide means for direct competition (in the form of duels or tournaments). Table 9 covers the relevant requirements, whereas Table 10 catalogs requirements imposed by different types of badges granted for achievements surpassing single exercises.

Table 9. Format requirements of types of challenges spanning beyond a single programming exercise.

Concept	Description	Format Requirements
Duel	Students can challenge another online student for a 3 exercises' match. The exercises must be related to the concepts being studied. It may also involve exhaustible resources to limit the number of challenges.	Exercise definition has a field specifying whether the exercise is suitable for duels.
Quest	A set of conditions on a player's progress that grants a reward once met. The player is aware of his/her active quest and its requirements (and possibly can choose the next quest after completing one).	Course definition has a list of quests within the course with relevant fields specifying the requirements and rewards.
Streak	A sequence of time units in which certain goals were consistently achieved by the player.	Course definition has a field specifying whether a streak is observed within the course and additional fields specifying the time unit (e.g., day or week), requirements (to be met on every time unit), and a list of rewards (to be granted on specified time units).
Story	Set of exercises wrapped in a storyline which develops as students complete an exercise. May be connected to one or more quests.	Course includes relevant story content (text possibly with audio-visual materials) to be presented in-between the challenges.
Tournament	Instructors may schedule tournaments composed of a set of exercises where every student can enroll, and winners are picked according to a predefined criterion.	Course definition may include a list of predefined sets of exercises selected for tournaments.
Mystery Track	Exercises that reveal additional exercises about the same (or related) concepts to offer more opportunity to practice.	Exercise reward type can be a content unlock. Reward item specifies what should be unlocked.

Table 10. Format requirements of badges available on course-level.

Concept	Description	Format Requirements
Solver	A badge awarded by solving N exercises in a row without a wrong submission, for N in {3, 5, 10, 15, }.	Reward requirement can refer to the number of exercises solved without wrong submissions in a row.
Man of Duty	A badge awarded by following a streak for N time units, for N in {3, 5, 10, 15, }.	Reward requirement can refer to the number of time units the player's streak lasts.
Runner	A badge awarded to students who were the first (second, third) to complete a full course or one of its modules.	Reward requirement can specify that it goes only to the first (second, third) player who met it.
Explorer	A badge awarded to students who revealed a specified number of secret content elements in a course.	Reward requirement can refer to the number of hidden content elements uncovered in a course.
Pathfinder	A badge awarded to students who were the first to complete a specified number of exercises in a course.	Reward requirement can refer to the number of exercises that the player solved the first.

5. Conclusion

Gamification is a promising remedy for the difficulty of learning and teaching computer programming. Its practical implementation relies on an open access to gamified programming exercises and tools to edit and use them. A necessary prerequisite for this to happen is the availability of an appropriate format for the effective specification and exchange of gamified programming exercises.

Having previously gathered and adopted some of the gamification concepts for the purposes of programming education [2], in this paper, we make one step further towards a framework for gamified programming education by identifying the requirements for the format that could be used to specify gamified programming exercises. Our key contributions are the design of a three-tier architecture of the format, addressing the three kinds of information that need to be specified for gamified programming exercises, and the identification of detailed requirements for the format with regard to the respective gamification concepts proposed for programming education.

The work described here will continue in the future according to the plan of the Framework for Gamified Programming Education project. After developing the format meeting the requirements presented here, the consecutive next steps are to develop exercise editing tools handling data in this format, programming exercises conforming to it, and learning environments presenting such exercises to students.

Acknowledgment

This paper is based on the work done within the Framework for Gamified Programming Education project supported by the European Union's Erasmus Plus programme (agreement no. 2018-1-PL01-KA203-050803).

References

[1] Edwards, S.H., Börstler, J., Cassel, L.N., Hall, M.S., Hollingsworth, J., 2008. Developing a common format for sharing programming assignments. SIGCSE Bull. 40, 167–182. URL: http://doi.acm.org/10.1145/1473195.1473240, doi:10.1145/1473195.1473240.

[2] FGPE Project Consortium, 2019. IO1: Gamification Scheme for Programming Exercises. Version 1.0. http://fgpe.usz.edu.pl/wp-content/uploads/FGPE_IO1_Gamification_Scheme_for_Programming_Exercises.pdf. accessed on April 2019.

[3] Haobin, Z., 2012. freeproblemset. URL: https://github.com/zhblue/freeproblemset. accessed on April 2019.

[4] Kattis, 2019. Kattis. URL: https://open.kattis.com/. accessed on April 2019.

[5] Klenin, A., 2011. Common problem description format: Requirements. URL: https://ciiwiki.ecs.baylor.edu/images/1/1a/CPDF_Requirements.pdf. accessed on April 2019.

[6] Leal, J.P., 2018. Mooshak. URL: https://mooshak.dcc.fc.up.pt/. accessed on April 2019.

[7] Leal, J.P., Silva, F., 2003. Mooshak: a web-based multi-site programming contest system. Software: Practice and Experience 33, 567–581. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.522, doi:10.1002/spe.522.

[8] Queirós, R., Leal, J.P., 2011. PExIL: Programming exercises interoperability language, in: Conferência Nacional XATA: XML, aplicações e tecnologias associadas, 9. ^{a}, ESEIG. pp. 37–48.

[9] Queiros, R., Leal, J.P., 2013. BabeLO - an extensible converter of programming exercises formats. IEEE Trans. Learn. Technol. 6, 38–45. URL: http://dx.doi.org/10.1109/TLT.2012.21, doi:10.1109/TLT.2012.21.

[10] Scheffers, E., Verhoeff, T., Geuns, S., Kruisselbrink, M., Wagener, P., Leenders, R., Macesanu, I., Steneker, M., 2017. peach3. URL: https://peach3.nl. accessed on April 2019.

[11] Swacha, J., 2018. SIPE: A Domain-Specific Language for Specifying Interactive Programming Exercises, in: Towards a Synergistic Combination of Research and Practice in Software Engineering, Springer, Cham, Switzerland. pp. 15–29.

[12] Verhoeff, T., 2008. Programming task packages: Peach exchange format. International Journal Olympiads In Informatics 2, 192–207.