information

MDPI

Article

GEdIL—Gamified Education Interoperability Language

Jakub Swacha [1,*], José Carlos Paiva [2,3,*], José Paulo Leal [2,3,*], Ricardo Queirós [2,4,*], Raffaele Montella [5] and Sokol Kosta [6]

1 Department of Information Technology in Management, University of Szczecin, 70-453 Szczecin, Poland
2 CRACS—INESC Porto LA, 4169-007 Porto, Portugal
3 Department of Computer Science, Faculty of Sciences, University of Porto, 4169-007 Porto, Portugal
4 uniMAD—ESMAD, Polytechnic of Porto, 4480-876 Vila do Conde, Portugal
5 Department of Science and Technology, University of Naples "Parthenope", 80133 Naples, Italy; raffaele.montella@uniparthenope.it
6 Department of Electronic Systems, Aalborg University, 9220 Aalborg, Denmark; sok@es.aau.dk
* Correspondence: jakub.swacha@usz.edu.pl (J.S.); jose.c.paiva@inesctec.pt (J.C.P.); zp@dcc.fc.up.pt (J.P.L.); ricardoqueiros@esmad.ipp.pt (R.Q.)

Received: 30 April 2020; Accepted: 26 May 2020; Published: 28 May 2020

Abstract: The paper introduces Gamified Education Interoperability Language (GEdIL), designed as a means to represent the set of gamification concepts and rules applied to courses and exercises separately from their actual educational content. This way, GEdIL allows not only for an easy yet effective specification of gamification schemes for educational purposes, but also sharing them among instructors and reusing in various courses. GEdIL is published as an open format, independent from any commercial vendor, and supported with dedicated open-source software.

Keywords: gamification in education; gamification language; gamification data format

1. Introduction

With the world becoming more and more dependent on software, there is a growing demand for those capable of developing it. According to recent research, while the number of software developers in the world has already surpassed 24 million, it is expected to increase by an additional 5 million in the forthcoming five years [1]. This obviously stresses the importance of learning to program, which has already become, as aptly put by Sedgewick et al. [2], "an essential part of the education of every student in the sciences and engineering". The problem with learning programming is that it is difficult as observed by various researchers (see, e.g., [3] and works cited therein).

Gamification, consisting of the use of game design elements outside of games, was proven as an effective means to counteract the decreasing engagement of the students coping with the difficulty of learning programming [4]. While there are several reports on the use of gamification in programming education (see, e.g., [5], especially Table I, for a review of a selection of such attempts), considering the scale of programming education worldwide, this approach is far from widespread. In our opinion, one of the main barriers for its wider adoption is the closedness of existing solutions: one can either use existing gamified courses and platforms or develop one's own from scratch, but there is no open repository of programming exercises with attached sets of gamification rules nor open platforms that would handle them so that the exercises could be reused in various contexts and the rules adapted for specific needs.

Overcoming this gap is the main goal of the Framework for Gamified Programming Education project [6]. This paper is devoted to one of the key results of this project, namely an open format for the specification of the gamification layer of educational contents such as programming exercises. The need for such a format arises from both the complexity of the existing languages/formats and

their lack of reusability, as they attempt to cover the whole board of gamification elements, without focusing on the specific needs of a domain such as education, use intermediate compilations, or are close to a particular service [7–9].

Note that while the proposed format was conceived originally for gamification of computer programming exercises, it is in no way constrained to this area of use, and in this paper, we emphasize its applicability to gamification of any kind of educational exercises. The rest of this paper is organized as follows: the related work panorama is framed in Section 2; GEdIL, the modeling language for gamification in educational contexts which represents the novel contribution proposed in this work, is detailed in Section 3; Section 4 describes the fulfillment of the requirements for the programming domain; finally, the concluding remarks and some future directions are in Section 5.

2. Related Work

While many gamified applications use gamification features embedded in their code, separating the gamification layer has many virtues, such as the increase in reuse, adaptability, and maintainability.

The separation of the gamification layer does not imply a dedicated format as the rules can be expressed in any general-purpose programming language. However, their specific nature advocates the use of a domain-specific language. Perhaps the best-known example of the latter approach is GaML (Gamification Modeling Language), a textual, declarative, and platform-independent gamification modeling language [7]. It includes most of the widely used gamification concepts: `GameLevel`, `Point`, `Skill`, `Mission`, `Role`, `Leaderboard`, `Level`, `Goods`, `Badge`, and `Event`. The actions that could be featured in the rules include `User Actions` (something done by the user of the gamified system), `External Events` (e.g. specific time of day), `Interim Events` (being an interim result of another gamification rule), `Context` (achieving a specific state, e.g., passing a score threshold), `Constraints` (temporal, spatial, boolean, numeric or random, e.g., performing some action within a specified time after achieving something), `Randomness` (limiting the probability of the rule producing its result), and `Joint Actions` (performing some action by several users together). The language has been designed as "fully writable for IT experts and partially writable by domain experts", meaning that the latter should be able to understand all but the most complex expressions of GaML.

While GaML relies on the external (gamified) system to identify events that are relevant for triggering the gamification rules before exposing them, the notation for the representation of events and rules proposed in [10] makes it a part of the rule specification. The only requirement for the gamified system is, therefore, to produce a standardized stream of communications exposing all the potentially relevant, possibly low-level events happening therein. Each such event is represented as a tuple of nine elements: `Player`, `Software Client`, `Area`, `Location`, `Object`, `Action`, `Action Result`, `Date`, and `Time`. The rules are defined in two separate parts linked by the same rule name: `Event Selector` and `Rule Result` for the sake of both clarity and the ability to match multiple selectors with a single result and vice versa. The selectors can be based on the external events (which have then to conform to defined conditions), their repetitions (possibly in a specific sequence) and game-state-related events (caused indirectly by other rules), and can be combined to form compound rules. The allowed rule results include feedback to the user, rewards, challenges being offered, started or finished, and other manipulation of the game state properties.

Ašeriškis, in his PhD thesis, introduces the UAREI model for formal specification of gamification, coupled with a visual modeling language for graphical representation of game mechanics [8]. Its name stems from the five gamification elements it models: `User`, `Action`, `Rule`, `Entities` and `Interface`. `Users` trigger `Rules` with their `Actions`, the `Rules` interact with data represented in `Entities`, and `Interface` defines what is displayed to `Users`. While UAREI is primarily intended for design and analysis of gamification systems, it can be transformed to the simplified UAREI JSON model (serializable using JavaScript Object Notation, hence its name) which can, in turn, be transformed into executable JavaScript code for the purposes of development and simulation of gamification systems.

GameLayer [9] is a cloud-based platform that enables the creation of mechanics for applications and services to increase engagement from its users. At its core, GameLayer uses a gamification system based on a well-defined structure composed of several entities such as players, rewards, achievements, levels, events, and missions. Players are the cornerstone of the gamification system. As they progress in the gamified application, they are rewarded, increasing their status, unblocking new accesses, or receiving more power (e.g., gaining permissions to moderate or ban users on chats and forums). Rewards in GameLayer consist mainly of points or credits.

There are also proprietary solutions (e.g., Gametize [11], IActionable [12], Bunchball [13]), more focused on business and client/employee engagement, which offer complementary features such as a content management system, customizable achievements, multiple mechanisms to motivate social behaviors, on-boarding, reports, and analytics.

Even if it cannot be used for actual gamification implementation (i.e., it cannot be automatically interpreted by a rule engine), the Machinations framework [14] represents an interesting browser-based solution for collaborative game design and prototyping. Using a methodology not dissimilar than other collaborative tools leveraging on cloud SaaS (Software as a Service) model, enables a group of designers to create game logic, levels, challenges, rewards, and events. The creation of the game-field logic uses a diagram schema. The output is a graph data structure in which the nodes are the game objects (levels, quests, etc.) and the arcs the actions the player has to perform to unlock the next game object.

3. GEdIL

Gamified Education Interoperability Language (GEdIL) is a modeling language for gamification in educational contexts, that can be serialized as JSON. It describes gamification layers in a way that they can be built using a simple multistep form, without any programming knowledge. GEdIL sets up in the assumption that its primary consumer is a Gamification Engine, which uses it as an itinerary to update the state when an event occurs or an action is performed. The only requirement that the state must meet, according to GEdIL, is to include information on the state of (1) the environment and (2) each player.

GEdIL was initially designed to fulfil specific requirements of gamification applied in programming courses [15]. These requirements identify a vast collection of rewarding mechanisms such as points, badges, virtual items, and social status (e.g., through leaderboards), to provide extrinsic motivation, but can also affect the educational content directly through unlockable and secret content, different activity modes (e.g., speedup and duels), among others. Nonetheless, GEdIL completely separates the gamification layer from the activities being gamified, which makes it sufficiently generic to be applied to any other educational subjects, provided that activities have a unique ID.

The next subsections detail the structure of GEdIL with an example use case.

3.1. Multiplication Tables (Toy Domain)

To demonstrate the aptitude of GEdIL to be employed in any educational domain as well as to facilitate its understanding by the reader, the next subsection will make use of a primary school module as a toy domain, the multiplication tables. Each activity of this module provides the student with two integers, between 1 and 10, and expects the result of their multiplication. Activities are identified as `Ax.y`, where `x` and `y` are the first and second operands of the multiplication, respectively. For instance, activity `A2.4` asks the student to write the result of 2×4.

The gamification layer of the toy domain includes an (RQ1) bonus point for each accepted exercise, (RQ2) a badge for solving the ten activities with prefix `A2`, and an (RQ3) leaderboard with the top point collectors. Activities with prefix `A10` are unlocked only after `A9.9` is solved (RQ4).

3.2. Structure

GEdIL defines a hierarchy of Challenges rooted by a Gamification Layer, in which both elements can hold Rewards, Rules, and Leaderboards. The vertical position of an element in the hierarchy determines its scope, as elements closer to the root may be accessed in lower levels, but not the other way around. For instance, leaderboards attached to the gamification layer are global and consider every challenge, whereas a leaderboard within a challenge only uses data of that same challenge and its branches. Figure 1 presents the data model encoded in GEdIL.

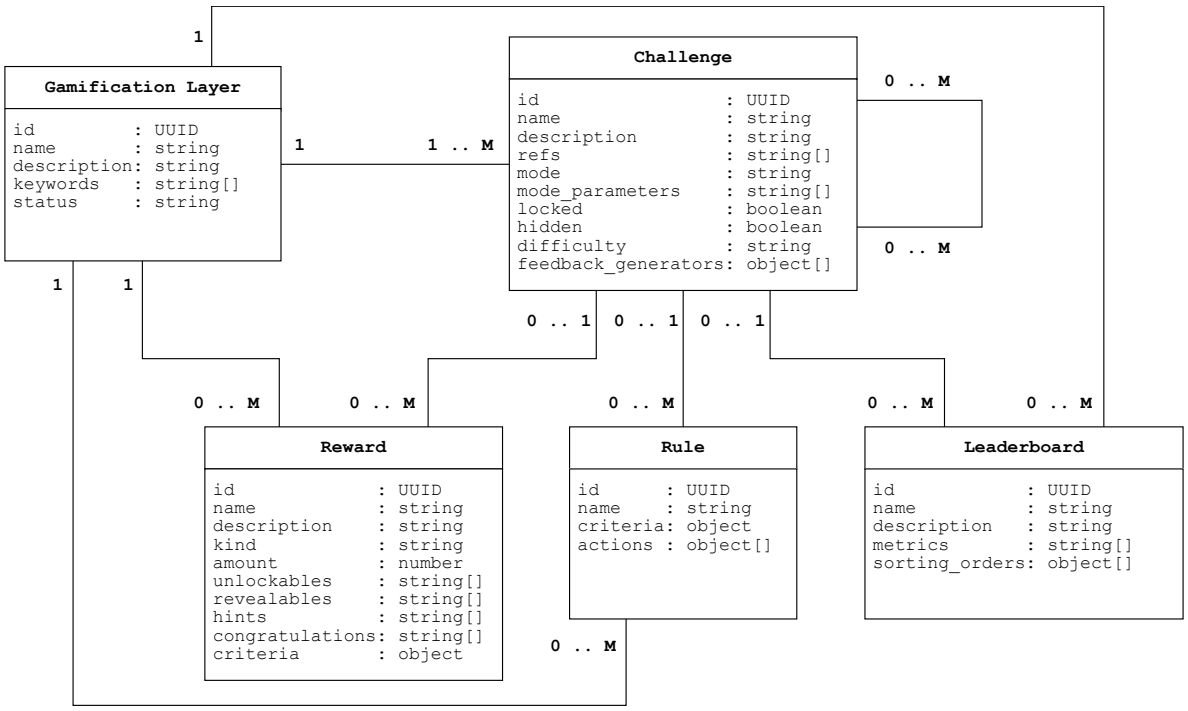

Figure 1. Data model defined by the GEdIL format.

The Gamification Layer holds the metadata for identifying the layer, in particular, the Universally Unique Identifier (UUID) [16] of the layer (`id`), the `name` of the layer, a `description` of the layer purpose and contents, the set of `keywords` that best categorizes the layer, and the `status` of the layer (possible values are `DRAFT`, `PUBLISHED`, `UNPUBLISHED`, `TRASH`); as well as pointers to global rewards, rules, leaderboards, and the first-level of challenges. Despite the fact that it does not encode any logic and, thus, it is not relevant for gamification (i.e., could be replaced with a top-level challenge), the proper identification of the layers is the key to ensure their reusability and possibility of replacement.

Challenges are the core of GEdIL. They "wrap" one or more non-gamified activities with a gamification envelope, which allows locking, hiding, and/or leveling-up the summon to solve it (e.g., by using `mode` and `mode_parameters` the activity could be limited to a certain timeframe). In fact, challenges are activities themselves, distinct from the original, with a `name`, a `description` of the goals, a `difficulty` level (`BEGINNER`, `EASY`, `AVERAGE`, `HARD`, or `MASTER`), and, possibly, feedback on completion (provided through an executable script in `feedback_generators`). Furthermore, challenges also serve the purpose of organizing the content as they can be chained, not only providing the student with some structure to follow but also enabling the composition of more complex gamification mechanisms, such as quests, stories, or duels.

With regard to the multiplication tables domain, a possible gamification layer meeting the requirements could include three challenges connected to the root: C1—which wraps all activities with prefix `A2`; C2—which wraps `A9.9`; and C3—which is initially locked and wraps all activities with prefix `A10`. In this way, the only missing stem to cover RQ2 is to attach a Reward to challenge C1 of `kind` `BADGE` (other possible values are `POINT`, `VIRTUAL_ITEM`, `COUPON`, `REVEAL`, `UNLOCK`, `HINT`, and `MESSAGE`),

with a suggestive `name` and `description`. As the badge is directly linked to the challenge, it should be delivered on its completion without the need for any additional criteria. The same applies for RQ4, as a reward of `kind` `UNLOCK` linking to challenge C3, through property `unlockables`, can be a child node of challenge C2. Then, when and if a student solves C2, he/she will unlock C3.

However, satisfying RQ1 is not so straightforward as it must be triggered on the completion of any activity, to reward the student. A possible approach is to wrap each activity in a challenge with a reward of type `POINT`, which would produce a heavy layer. Yet, GEdIL offers a fairly better strategy to accomplish this by using a Rule. Rules are triggered based on events or user actions, and they will modify the state with the value of an `action` attribute if their `criteria` are met. An action of a rule is a verb (e.g., `GIVE`, `TAKE`, or `UPDATE`) followed by any number of parameters. The criteria, which can also be used in rewards, is a list of `conditions` connected with `junctors` (`AND` and `OR`), where each condition has: a `left_entity/right_entity`—the type of entity holding the property to test, which is either `ENVIRONMENT` (i.e., current state of the environment), `PLAYER` (i.e., current state of the player related to the trigger), `ACTION` (i.e., the action performed by the student, if this check results of an action), `EVENT` (i.e., the event object, when activated in response to an internal state change or at a given time), or `FIXED` (i.e., a constant); a `left_property/right_property`—a reference to the property to test (e.g., using JSONPath); and a `comparing_function`—the function to use while comparing both sides.

Hence, RQ1 can be accomplished with a rule that gives a reward of `kind` `POINT` and an `amount` of 1 when an event has the type of submission with an accepted result. As the specific names and paths of these properties (in this case, type and result) will invariably depend on the underlying Gamification Engine, GEdIL does not set any special constraints upon them.

Finally, GEdIL also "imports" a well-known gamification mechanics, the Leaderboard. A leaderboard consists of a list of `metrics` and their respective `sorting_orders` to rank the players. This component is a prerequisite to fulfilling the missing requirement of the toy domain, RQ3, which now comes down to attaching a leaderboard sorted in descending order by points. Figure 2 presents a diagram of the complete gamification layer built for the multiplication tables example, masking IDs, and ignoring some empty fields.

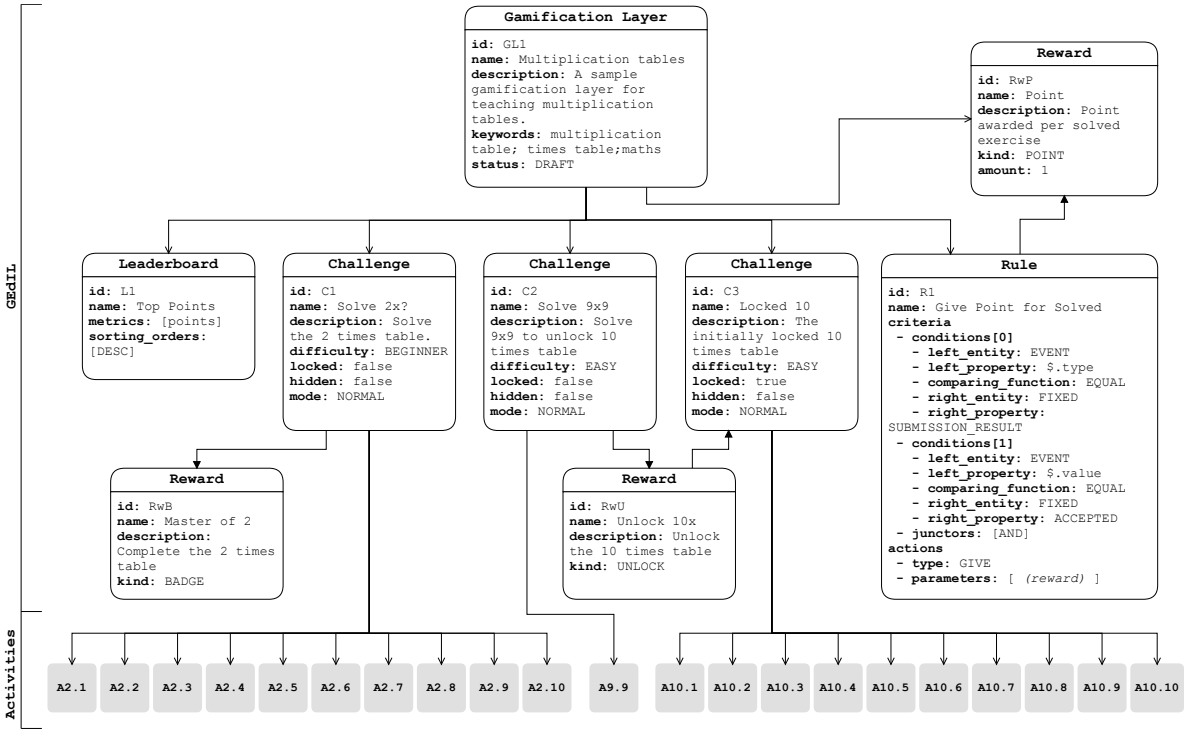

Figure 2. Diagram of the gamification layer of the toy domain (IDs masked for the sake of simplicity).

4. Requirements Fulfillment for the Programming Domain

As stated previously, GEdIL was designed to fulfill a specific list of requirements classified as relevant for the gamification of programming education [15]. This section raises each of the identified requirements and describes how it is accomplished using GEdIL.

The exercise types are out of the scope of GEdIL as they pertain to the programming exercises' format, so their requirements were neglected. For the sake of simplicity, the different badges, awarded either on the course-level or the exercise-level, were also ignored as they have all the same requirements only varying their name and criteria.

Table 1 targets the requirements of aspects related to the course organization. Table 2 describes the fulfillment of concepts related to the definition of goals. In Table 3, the requirements of the relevant rewards are accomplished, whereas Table 4 handles the conditions upon which they are granted. Table 5 covers the needs of the distinct modes in which an activity can be solved. Finally, Table 6 addresses the fulfillment of gamification concepts involving multiple challenges.

Table 1. Fulfillment of requirements from gamification concepts related to course organization.

Concept	Description of Fulfillment
Course Module	GEdIL supports a hierarchy of challenges. A module could be a challenge of the first-level.
Exercise Type	Out of scope (i.e., it should be handled by the programming exercises' format).
Exercise Mode	A challenge supports `mode` and `mode_parameters`, so it may wrap the activity with those modifiers.
Locked Content	A challenge supports a `locked` modifier.
Secret	A challenge supports a `hidden` modifier.
Difficulty Level	A challenge supports a `difficulty` property, with 5 levels.

Table 2. Fulfillment of requirements from gamification concepts related to goals definition.

Concept	Description of Fulfillment
Challenge	It is a component of the format.
Requirements	A challenge may describe its requirements in its `description`, but requirements of the main task (i.e., the exercise) should be handled by the programming exercises' format.
Quest	A challenge may wrap several related activities and give a reward on completion.
Streak	A rule may be added to check if a goal has been accomplished when an event is triggered, in which case a dedicated counter of the player is increased. Another rule may be added to check the counter and give a reward if it reaches the threshold.
Record	Leaderboards display the top-ranked players by any metric. Moreover, they can be attached at any level of the hierarchy.

Table 3. Fulfillment of requirements from gamification concepts related to definition of rewards.

Concept	Description of Fulfillment
Point	Reward of `kind` `POINT` with a certain `amount`.
Level	A rule to handle an event triggered when player's points increases may deal with level progression.
Held Record	Leaderboards display the top-ranked players by any metric. Moreover, they can be attached at any level of the hierarchy.
Current Rank	Same as above.
Badge	Reward of `kind` `BADGE`.
Virtual Item	Reward of `kind` `VIRTUAL_ITEM` with a certain `amount`.
Coupon	Reward of `kind` `COUPON` with a certain `amount`.
Content Discovery	Reward of `kind` `REVEAL` with a set of `revealables`.
Content Unlock	Reward of `kind` `UNLOCK` with a set of `unlockables`.
Hint	Reward of `kind` `HINT` with a set of `hints`.
Congratulations	Reward of `kind` `MESSAGE` with a set of `congratulations`.

Table 4. Fulfillment of requirements of the types of conditions upon which rewards are granted.

Concept	Description of Fulfillment
Attempt	A rule can be attached to the root or a specific challenge to give rewards based on attempts.
Achievement	A reward attached to the challenge is given on its completion (if the rest of the criteria is met).
Failure	A rule can be attached to the root or a specific challenge to give rewards based on failure count of the player submitting.
Progress threshold	A rule to handle an event triggered when player's points increases may give a reward when a certain threshold is achieved.
Progress in competition	Leaderboard is a component of GEdIL and supports several metrics.

Table 5. Fulfillment of requirements of gamified programming exercise modes.

Concept	Description of Fulfillment
Shapeshifter	A challenge supports `mode` (which can be `SHAPESHIFTER`) and `mode_parameters` (to specify other exercises), so it may wrap the activity with those modifiers.
Shortening challenge	A challenge supports `mode` (which can be `SHORTENING`) and `mode_parameters` (to specify threshold), so it may wrap the activity with those modifiers.
Speedup challenge	A challenge supports `mode` (which can be `SPEEDUP`) and `mode_parameters` (to specify time), so it may wrap the activity with those modifiers.
Hack the problem	A challenge supports `mode` (which can be `HACK_IT`) and `mode_parameters` (to specify the trick to search for), so it may wrap the activity with those modifiers. Rules may also be attached to detect tricks.
Time bomb	A challenge supports `mode` (which can be `TIME_BOMB`) and `mode_parameters` (to specify time), so it may wrap the activity with those modifiers.

Table 6. Fulfillment of requirements of complex challenges.

Concept	Description of Fulfillment
Duel	A challenge supports `mode DUEL` and may have inner challenges and reference multiple exercises.
Quest	See Table 2.
Streak	See Table 2.
Story	A challenge supports `feedback_generator` to generate any feedback needed to make the story between challenges and may have inner challenges (the chapters of the story).
Tournament	The complete layer can represent a tournament.
Mystery Track	A composition of challenges with rewards of `kind REVEAL`.

5. Conclusions

Gamification is a promising educational tool and its capability of rising engagement can be especially useful in the areas of teaching that require students to solve a large number of exercises (e.g., computer programming, mathematics, or physics). The effort needed to implement gamification in a specific course is nonetheless considerable. This effort can be significantly reduced by allowing the separation of the gamification layer from the educational content it is applied to so that the solicitously designed gamification rules could be easily reused and adapted to various subjects and courses.

In this paper, we introduced GEdIL, a domain-specific language and data format developed to represent the gamification layer of educational courses and exercises, and showed its practical applicability. Compared to the existing gamification specification formats, it is simple, designed purely for educational gamification (rather than being business-oriented), independent from the format used to represent the actual educational content, and open (with supporting open-source software provided).

As GEdIL is a textual language, it could be edited even with a generic text editor; this would require, however, very good knowledge of its specification, making it unappealing for teachers and trainers. Therefore, for the sake of making it easy to develop complex gamification scenarios for educational purposes, share them among instructors and reuse them in various courses, a collaborative web editor has been developed as open-source software [17].

Our most immediate future work is the development of an open library of reusable gamification components defined in GEdIL, from which educational course designers and instructors could select those most suitable for their own courses and apply them there in an effortless way.

Author Contributions: Conceptualization, all authors; data curation, J.S., J.C.P., R.Q., and J.P.L.; formal analysis, J.C.P., J.S., R.Q., and J.P.L.; funding acquisition, J.S.; investigation, all authors; methodology, all authors; project administration, J.S., R.Q., J.P.L., R.M., and S.K.; resources, J.C.P., J.S., R.Q., and J.P.L.; software, J.C.P.; supervision, J.S., R.Q., and J.P.L.; validation, J.C.P., R.Q., J.P.L., and J.S.; visualization, J.C.P., R.Q., J.P.L., and J.S.; writing—original draft preparation, J.C.P., J.S., R.Q. and R.M.; writing—review and editing, all authors. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Erasmus+ grant number 2018-1-PL01-KA203-050803.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lieby, V. Worldwide Professional Developer Population of 24 Million Projected to Grow amid Shifting Geographical Concentrations. 2019. Available online: https://evansdata.com/press/viewRelease.php?pressID=278 (accessed on 22 April 2020).
2. Sedgewick, R.; Wayne, K.D.; Dondero, R. *Introduction to Programming in Python: An Interdisciplinary Approach*; Addison-Wesley: New York, NY, USA, 2015.
3. Bosse, Y.; Gerosa, M.A. Why is programming so difficult to learn?: Patterns of difficulties related to programming learning mid-stage. *ACM SIGSOFT Softw. Eng. Notes* **2017**, *41*, 1–6. [CrossRef]

4. Rojas-López, A.; Rincón-Flores, E.G.; Mena, J.; García-Peñalvo, F.J.; Ramírez-Montoya, M.S. Engagement in the course of programming in higher education through the use of gamification. *Univers. Access Inf. Soc.* **2019**, *18*, 583–597. [CrossRef]
5. Swacha, J.; Queirós, R.; Paiva, J.C. Towards a Framework for Gamified Programming Education. In Proceedings of the 2019 International Symposium on Educational Technology (ISET), Hradec Kralove, Czech Republic, 2–4 July 2019; pp. 144–149. [CrossRef]
6. Framework for Gamified Programming Education, 2018. Project Website. Available online: http://fgpe.usz.edu.pl (accessed on 28 April 2020).
7. Herzig, P.; Jugel, K.; Momm, C.; Ameling, M.; Schill, A. GaML—A Modeling Language for Gamification. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, Dresden, Germany, 9–12 December 2013; pp. 494–499.
8. Aaeriakis, D.; Blažauskas, T.; Damaševičius, R. UAREI: A model for formal description and visual representation /software gamification. *DYNA* **2017**, *84*, 326–334. [CrossRef]
9. GameLayer. 2020. Available online: http://gamelayer.co/ (accessed on 28 April 2020).
10. Swacha, J. Representation of Events and Rules in Gamification Systems. *Procedia Comput. Sci.* **2018**, *126*, 2040–2049. [CrossRef]
11. Gametize. 2020. Available online: https://gametize.com/ (accessed on 28 April 2020).
12. IActionable. 2020. Available online: http://iactionable.com/ (accessed on 28 April 2020).
13. Bunchball. 2020. Available online: https://www.bunchball.com/ (accessed on 28 April 2020).
14. Dormans, J. Machinations: Elemental feedback structures for game design. In Proceedings of the GAMEON-NA Conference, Atlanta, GA, USA, 26–28 August 2009; pp. 33–40.
15. Swacha, J.; Queirós, R.; Paiva, J.C.; Leal, J.P. Defining Requirements for a Gamified Programming Exercises Format. *Procedia Comput. Sci.* **2019**, *159*, 2502–2511. [CrossRef]
16. Leach, P.; Mealling, M.; Salz, R. RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace. *RFC Editor* **2005**, *4122*, 1–32. [CrossRef]
17. FGPE AuthorKit. 2020. Available online: http://fgpe.dcc.fc.up.pt (accessed on 28 April 2020).