Generator of Personalised Training Games Activities: A Conceptual Design Approach

Bérénice Lemoine, Pierre Laforcade

Generator of Personalised Training Games Activities: A Conceptual Design Approach

First Author[0000−1111−2222−3333], Second Author[1111−2222−3333−4444], and Third Author[2222−−3333−4444−5555]

NOWHERE
{no-one}@research.com

Abstract. Memorizing declarative knowledge requires repetition, which can become wearing for learners. In addition, redundant game activities, offering unbalanced challenges in relation to the player's skills, can also lead to a sense of boredom. To reduce this feeling, learning games must provide adapted and varied activities. Automated generation is one way of building such activities. This article proposes a conceptual framework for the design of activity generators for training declarative knowledge in Roguelite games. The framework has been applied in the context of the {anonymised} project for multiplication tables training.

Keywords: Training Activities · Generation Design · Serious Games.

1 Introduction

Long and short term memorization of declarative knowledge (DK), such as facts and laws, requires repetition [12]. However, repetitive tasks can become boring for learners [17], which can lead to their abandonment. Furthermore, serious games that offer redundant activities with challenges that do not match their skills can also become wearing for learners [18]. Thus, to limit the feeling of boredom, serious games aimed at working on DK must offer a wide variety of varied and adapted activities. Adaptation can be implemented in various ways and can be aimed at one or more targets (e.g., game preferences, learning content, difficulty). Personalisation can be defined as the use of models for the purpose of tailoring systems to each person [4, 10].

Literature in cognitive psychology has shown that the process of retrieving concepts or facts through testing increases their long-term acquisition [6]. *Retrieval Practice* is a form of test-based learning consisting of repeated recall of what one has learned (e.g., through the use of flashcards, quizzes). Therefore, we define *training* on DK as a form of retrieval practice that consists of repeatedly providing the learners-players with various forms of questions on facts.

Activity generation is a solution for designing adapted and varied training activities that few works address in Technology-Enhanced Learning [5] (TEL). Generators are software components that use structured data to create elements

(e.g., text, documentation, activities). Building activity generators involves identifying and specifying the necessary elements for generation and their interactions. Our interest lies in the design of generators of learner-player personalized and varied game activities for DK training. To that extent, this article presents a conceptual approach for the design of such generators.

2 Research Context

Related Works. Although not widely addressed in TEL, content generation has been approached from three main angles: non-adapted content, learner-adapted content and player-adapted content. Most of these works have one thing in common: they use models to represent and structure the data required for generation (e.g., game elements, targeted knowledge, targeted content structure). Only, these models tend to be domain-dependant. Moreover, the elements not captured by the models, such as the adaptations rules, are captured in algorithms. Holohan et al. [9] defined an ontology to allow the description of relational databases. They then use this ontology to automatically generate online exercises for learning procedural knowledge (i.e., relational databases). Carpentier and Lourdeaux [8] have proposed an approach for dynamically generating scenarios adapted to learners' abilities and pedagogical needs in virtual environments. Their approach is part of a framework based on three models: the domain (i.e., static description of the world, its elements and their relationships), the activity (i.e., hierarchical structure of the observed activity), the causality (i.e., expresses the relevant causal chains occurring in the environment). Sehaba and Hussaan [16] have proposed a general architecture for learner-adapted (i.e., competencies, skills, needs) game scenario generation. This architecture is based on several models: the *domain model* models the domain concepts and their relations; the *learner model* models learners' personal information, motivation, skills and interactions; the *presentation model* describes the structures of the scenarios; the *serious game model* associates game resources to pedagogical ones. Moreover, adaptation knowledge is represented as a rule-based system. Laforcade and Laghouaouta [13] have proposed a Model Driven Engineering (MDE) approach to understand the specification of generators of activities sequences (i.e., game scenario) adapted to individual learner's needs. This approach is based on [16], and mainly use the same models. Callies et al. [7] have proposed an adaptive architecture consisting in generating adapted pedagogical plans as well as in adapting the behaviours of the non-player characters according to the players' actions in simulation-type games. This work is based on a player model composed of his/her game and domain knowledge, as well as an adaptation module.

Outside a generation standpoint, some works propose architectures to guide the design of personalised learning systems. Roepke et al. [15] proposed a modular, component-based architecture for implementing personalised pipelines for learning games for anti-phishing education. A pipeline is a three steps process: data collection, content generation, content delivery. Ismail and Belkhouche [10] proposed a reusable architecture for the design of personalised learning soft-

ware systems decomposed in 4 units: learner (i.e., maintains data about the learner), knowledge (i.e., maintains learning resources), personalisation (i.e., maps learner's model to learning resources), and presentation (i.e., represents the software environment). However, although the design of game activity generators requires choices to be made about game design, their design varies from pure game design. Furthermore, Tang and Hanneghan [19] discovered that there was no game model that offered a complete representation of game concepts from a design viewpoint for the use of MDE. Therefore, they propose a game ontology representing design aspects of video games for simulation and role-play genres, which also describes concepts linked to game activities.

Research Positioning. Our work targets declarative knowledge training (i.e., repeatedly providing the learners-players with various forms of questions on facts), independently of a specific didactic domain, through games. Consequently, the proposal must be reusable in a similar way to [10] in order to be extended to domain-specific DK. Training requires repetitive activities [12] as well as varied and personalised activities to avoid boredom caused by redundancy and inadequate challenges [17, 18]. Depending on the genre, the structure of a game activity changes completely. Therefore, it was necessary to select a suitable game genre for DK training. Roguelite is a dungeon-like genre that meets DK training needs [2]. It is mainly characterized by the procedural generation of dungeons with pseudo-random content, permanent death (each death of the avatar forces the player to start a new game), and the limited possession of unlockable elements (e.g., avatars, items, power-ups). Therefore, a *training game activity* is a *dungeon*, i.e., a set of interconnected rooms in which the avatar moves around and in which the training takes place. In our context, adaptation focuses on both educational dimension and game dimension by considering teachers' viewpoint on training for each learner, learners' progress in their training, and players' preferences. Consequently, our research questions are as follows: 1) What are the elements involved in an activity generator? 2) What are the relationships between these elements? 3) How can the elements and their relations be structured to construct such generators? and 4) How to consider DK independently of a specific didactic domain? Our proposition is a framework extensible to domain specific DK. This extensible framework is a conceptual and software infrastructure composed of models and tools to formalize and guide the implementation of varied and adapted activity generators. This is a contribution in engineering research of TEL systems [20] contributing to the exploration and orientation of solutions for the generation of adapted activities. This article **focuses on the conceptual aspects** of the framework, describing the different models involved in the generation process, how they are built, and their relationships.

3 Activity Generation Needs

Activity generators are software components that take structured data as input and provide detailed descriptions of activities as output (e.g., XML files). They

are composed of knowledge about the structure of the data they need and a generation algorithm (e.g., rule-based system, procedural generation) they follow to create activities. Dungeon generators for DK training require different models to work, cf. Figure 1. First, a *domain model* describing the training path and the facts to be worked on. Second, a *learner-player model* that keeps track of the learner's progress in his/her training and in the game, and his/her game preferences so that the generator can personalise the activity on the basis of these data. Next, a *game model* which describes the game elements available (e.g., gameplays, game objects). Then, a *relation model* that describes the relations between the game elements and the training elements so that the generator builds coherent activities. In the literature, relationships are often implemented directly in the generation algorithm and rarely made explicit or modelled. Finally, to build an activity, the generators need to know its structure, which is why the *activity model* describes the structure of a dungeon for DK training.

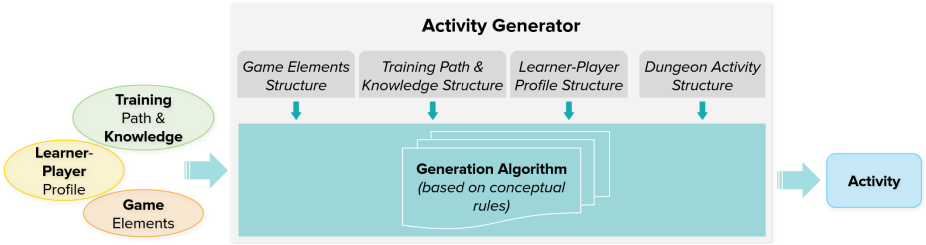

Fig. 1. Training Game Activity Generator Overview.

3.1 Domain Model: Training Paths & Knowledge

In the context of the {anonymised} project, that aims at building a game for multiplication tables training, we conducted an exploratory research [1] with the help of experts in mathematics (i.e., teachers and didacticiens). The exploratory research had two main objectives: 1) how to organize training, and 2) what adaptations to consider for training on multiplication tables (i.e., source and targets of adaptation). This led us to organize training into a structure called training path. A *training path* is a set of objectives to achieve, ordered by prerequisite relationships. An objective concerns a set of facts to work on, and is unlocked when its prerequisites are met (i.e., making training a step-by-step process with increasing difficulty). Each objective is decomposed into progressive difficulty levels. A level is composed of a series of tasks, having domain-specific parameters, which the learner must complete in order to progress in the training. Such training paths, from objectives to task parameters, are defined by a teacher for a learner or a group of learners. Therefore, each path is adapted, based on the teacher's viewpoint, to each learner individually.

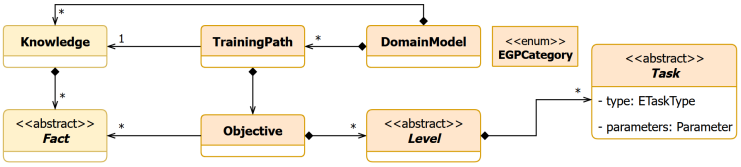

Fig. 2. Simplified UML class diagram of the domain model.

DK are specific to the targeted didactic domain. For example, a multiplication table fact can be represented as an object with three integers x (operand), y (table) and res (result). On the other hand, a historical date would rather be represented by an object with a string (event) and a date or a period (integers). In addition, level and task parameters are also domain-dependant. For example, multiplication tables facts can be built in different ways, operand × table or table × operand, which is solely dependent on the domain of mathematics. Moreover, for a task consisting of completing facts with a missing element (e.g., find the historical date, the result of a multiplication), the element sought depends on the domain too (e.g., result, operand, or table for multiplication, and event or date for history). As a result, this model needs to be extended, by an engineer, on the basis of data provided by the teacher, at three strategic points: Fact, Level, Task. Figure 2 presents a simplified overview of the model, extension points are represented as abstract classes. On the basis of this study and discussions with history-geography teachers, specific tasks were defined for each field. Observation of these tasks led to the definition of four types of task (i.e., super-classes of the different tasks): Completion (i.e., completing a fact with missing elements), Order (i.e., ordering facts using a heuristic), Identification (i.e., attesting the validity or invalidity of facts) and Membership Identification (i.e., identifying elements with a common property).

3.2 Game Model: Game Elements Structure

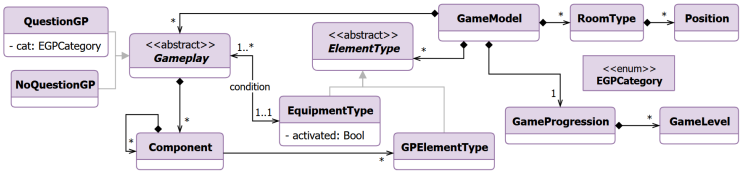

Fig. 3. Simplified UML class diagram of the game model.

Prensky [14] said that the main reason for learning game failure lies in their lack of gameplay. To that extent, we decided to provide a variety of gameplay (i.e., action that the avatar can perform within a dungeon that impacts the

learner-player's progress). On the basis of informal interviews with game designers, we designed gameplays and gameplay categories (i.e., they do not claim to be exhaustive) for DK training through Roguelite: SELECT (i.e., selecting objects with the right answer), MOVE (i.e., moving the correct objects to the expected areas), ORIENT (i.e., orienting the object to the right answers), POSITION (i.e., placing the avatar in the right positions) and DIRECT RESPONSE (i.e., typing in the right answers) [2]. Consequently, the game model is composed of gameplays. There are two types of gameplay: gameplays for answering questions and gameplays purely for gaming. Question gameplays all fall into one of the five categories mentioned above. The other gameplays describe game elements such as traps to avoid, objects to break to get more coins, and so on. Every gameplay are described by components that refer to a concrete type of game element (i.e., element that can be used within gameplays, such as pots, statues, traps).

Roguelites often feature an economic game mechanic, meaning that coins are collected throughout the levels, enabling the player to buy and activate items such as equipments. It is important to note that the creation of a game activity generator does not require the creation of a game (as it is an independent software component). However, creating game activities does require choices to be made in terms of game design. For this reason, our design choices included this mechanism. As a result, our game model features equipment and items that the learner-player can purchase and activate according to his or her preferences. To ensure that these preferences have an impact on generation, and are not just aesthetic preferences, our choice was to offer items which, once activated, unlock new gameplays. So if a learner does not like a gameplay, they can deactivate the item and make it unavailable to the generator. In order to have a variety of rooms, we have chosen to model room types that describe the different room shapes that the generator can use. Finally, the game model describes the progression of the game in terms of difficulty levels. A game level can modify the size of the dungeon (i.e., the number of rooms), its shape (i.e., linear or labyrinthine) and the possible traps (i.e., difficulty and number). This model must be specified by game designers or engineers. Figure 3 presents an overview of the model.

3.3 Relation Model

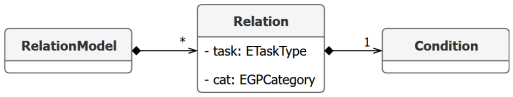

Fig. 4. Simplified UML class diagram of the relation model.

For the purpose of generating coherent activities, the generator must be able to associate the game and training elements correctly. In order to define the relationships between our elements, we proposed a systematic method for defining

machine-readable relationships between gameplay categories and task types for DK training. The method is described here [3]. The resulting relationships associate a gameplay category and a task type with a condition (i.e., a set of valued parameters required to associate them within an activity). Figure 4 describe an overview of the relation model. This model must be defined by engineers.

3.4 Learner-Player Model

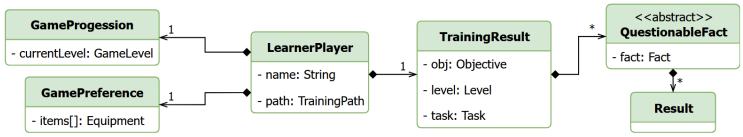

Fig. 5. Simplified UML class diagram of the learner-player model.

Facts present in the domain model are raw information such as $3 \times 5 = 15$. Depending on the task and its parameters, the questions about the facts to practice changes. For example, completing a fact with a missing element will yield questions such as $3 \times ? = 15$, while identifying if elements share a given property will yield questions such as "Which ones are results of tables three? $\{3, 5, 9, 13\}$". Therefore, like the facts, level and tasks, these questions need to be modelled dependently of the didactic domain (i.e., another extension point). The questions present in the dungeon will have both correct and incorrect propositions (when the answer modality is Choice). From a didactic perspective, it is more interesting if the incorrect answers vary constantly. However, in order to compare learners' results on a given fact, it is necessary to have a common base that does not change (i.e., keep the question format but not the elements that vary). To this end, we propose a two-stage fact transformation process. First, questionable facts are built based on the raw facts present in the domain model. A questionable fact represents a question about a fact without incorrect propositions. Such facts are used to retain learners' results in the learner-player model (e.g., response times, given answers). Secondly, questioned facts are built based on questionable facts (i.e., they are questionable facts with incorrect answers). These facts will be those present in the activity (cf. Figure 6).

As mentioned earlier, players' preferences reside in the items they purchase and activate. As a result, the learner-player model keeps track of which items are purchased/activated. In addition, the model keeps track of the player's game level (i.e., game progression). Figure 5 presents an overview of this model.

3.5 Dungeon Activity Model

Figure 6 describes the simplified structure of a Roguelite-like activity for DK training. An activity is a dungeon composed of rooms including an entry and

an exit. A room dispose of access to other rooms (i.e., its neighbours) as well as a type (i.e., its shape) and a gameplay (cf. Section 3.2). Moreover, rooms with question gameplays are also associated to a specific task and a set of questioned facts (i.e., facts that are questioned in the room). Finally, each room is described by their concrete positioned game elements that can be linked to questioned facts (e.g., elements wearing choices or statements).

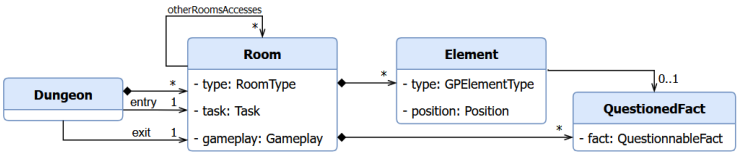

Fig. 6. Simplified UML class diagram of the activity model.

4 A Design Framework of Activity Generators

4.1 Design Framework: Presentation

For proof-of-concept purposes, the presented conceptual design approach has been implemented inside a framework (i.e., software infrastructure composed of models and tools) using Model-Driven Engineering [11] tools and principles. The aim of this framework is to guide the implementation of varied and adapted game activity generators for DK training.

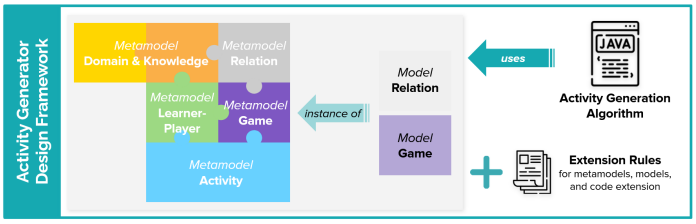

Fig. 7. Activity Generator Design and Implementation Framework Overview.

The framework (cf. Figure 7) is composed of *metamodels* (i.e., models whose instances are also models), *models* (i.e., instances of the metamodels), an *activity generation algorithm* implemented in Java, and a set of *extension rules* (i.e., description of the different parts of metamodels, models and code to extend in order to build a domain-specific activity generator). In this framework, every conceptual model has been translated into a metamodel. The framework also includes an instance of the relation and game metamodels (i.e., models) as default.

However, domain/knowledge model and learner-player model (i.e., only his/her personal information, such as name) must be instantiated and their metamodel extended by following the extension rules.

4.2 Design Framework: Application to {Anonymised}

{Anonymised} is a research project that aims at building a game for multiplication tables training. Currently, a game prototype has been implemented. This prototype uses an activity generator built by using the framework presented in Section 4 (i.e., creation of an extension of the framework for multiplication tables). This generator provides XML files describing the dungeons (i.e., each element and its position), which the game player (i.e., prototype) interprets to create a playable dungeon. Figure 8 shows screenshots from the prototype displaying two different gameplays. Both gameplays are questioning the same fact - $3 \times 5 = 15$ - and both questioned facts were built from the same questionable fact (cf. Section 3.4). However, both questioned facts are different in terms of their possible incorrect answers. In addition, one gameplay is of the SELECT category (i.e., the player must touch the correct rabbit) while the second is of the MOVE category (i.e., the player must place the correct pot on the tile).

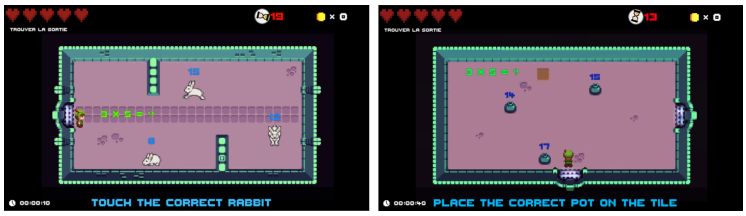

Fig. 8. Example of SELECT and MOVE gameplays for a Completion Task.

5 Conclusion & Perspectives

In conclusion, this article describes a conceptual approach for the design of game activity generators for DK training. This conceptual approach has been implemented in a model-driven engineering framework that is extensible to specific didactic domains. This framework has been extended to build an activity generator for multiplication table training that is currently being used in a game prototype. The main limitation of our work is that the models and metamodels present within the framework are dependent on the game genre, in the way that they target DK learning through Roguelite activities (i.e., dungeons). In addition, there are many ways of modelling the information presented, and each of our choices can be argued according to different viewpoints. In the future, we intend to extend that framework to a second didactic domain: history-geography facts of the *Diplôme National du Brevet* an exam taken in Year 10 in France.

References

1. Author: Title3. In: Anonyme (2022)
2. Author, N.: Title1. In: Anonyme (2023)
3. Author, N.: Title2. In: Anonyme (2023)
4. Bakkes, S., Tan, C.T., Pisan, Y.: Personalised gaming: A motivation and overview of literature. In: Proceedings of The 8th Australasian Conference on Interactive Entertainment: Playing the System. pp. 1–10. ACM, Auckland New Zealand (2012)
5. Bezza, A., Balla, A., Marir, F.: An approach for personalizing learning content in e-learning systems: A review. In: Second International Conference on E-Learning and E-Technologies in Education. pp. 218–223. IEEE, Lodz, Poland (2013)
6. Brame, C.J., Biel, R.: Test-Enhanced Learning: The Potential for Testing to Promote Greater Learning in Undergraduate Science Courses. LSE **14**(2) (2015)
7. Callies, S., Sola, N., Beaudry, E., Basque, J.: An empirical evaluation of a serious simulation game architecture for automatic adaptation. In: R. Munkvold & L. Kolas, Proceedings of the 9th ECGBL. pp. 107–116 (2015)
8. Carpentier, K., Lourdeaux, D.: Generation of Learning Situations According to the Learner's Profile Within a Virtual Environment. In: Agents and Artificial Intelligence, vol. 449, pp. 245–260. Springer, Berlin, Heidelberg (2014)
9. Holohan, E., Melia, M., McMullen, D., Pahl, C.: The Generation of E-Learning Exercise Problems from Subject Ontologies. In: 6th International Conference on Advanced Learning Technologies. pp. 967–969. IEEE, Kerkrade, Netherlands (2006)
10. Ismail, H., Belkhouche, B.: A Reusable Software Architecture for Personalized Learning Systems. In: 2018 International Conference on Innovations in Information Technology (IIT). pp. 105–110. IEEE, Al Ain (2018)
11. Kent, S.: Model Driven Engineering. In: Integrated Formal Methods. pp. 286–298. Springer Berlin Heidelberg (2002)
12. Kim, J.W., Ritter, F.E., Koubek, R.J.: An integrated theory for improved skill acquisition and retention in the three stages of learning. Theoretical Issues in Ergonomics Science **14**(1), 22–37 (2013)
13. Laforcade, P., Laghouaouta, Y.: Generation of Adapted Learning Game Scenarios: A Model-Driven Engineering Approach. In: 10th International Conference CSEDU, Funchal, Madeira, Portugal. vol. 1022, pp. 95–116. Springer (2018)
14. Prensky, M.: Computer Games and Learning: Digital Game-Based Learning. Handbook of Computer Game Studies (2005)
15. Roepke, R., Drury, V., Schroeder, U., Meyer, U.: A modular architecture for personalized learning content in anti-phishing learning games. In: Software Engineering (Satellite Events) (2021)
16. Sehaba, K., Hussaan, A.M.: GOALS: Generator of adaptive learning scenarios. IJLT **8**(3), 224 (2013)
17. Smith, R.P.: Boredom: A review. Human factors **23**(3), 329–340 (1981)
18. Streicher, A., Smeddinck, J.D.: Personalized and Adaptive Serious Games. In: Entertainment Computing and Serious Games, vol. 9970, pp. 332–377. Springer International Publishing, Cham (2016)
19. Tang, S., Hanneghan, M.: Game Content Model: An Ontology for Documenting Serious Game Design. In: 2011 Developments in E-systems Engineering. pp. 431–436. IEEE, Dubai, United Arab Emirates (2011)
20. Tchounikine, P., Mørch, A.I., Bannon, L.J.: A Computer Science Perspective on Technology-Enhanced Learning Research. In: Technology-Enhanced Learning, pp. 275–288. Springer Netherlands, Dordrecht (2009)