A Framework for Generators of Varied and Adapted Training Game Activities

Bérénice Lemoine, Pierre Laforcade, Sébastien George

A Framework for Generators of Varied and Adapted Training Game Activities

Bérénice Lemoine[1 [0000−0002−7608−3223]], Pierre Laforcade[1 [0000−0001−8498−2731]], and Sébastien George[1 [0000−0003−0812−0712]]

LIUM Computer Science Laboratory, Le Mans Université, Laval, France
{berenice.lemoine, pierre.laforcade, sebastien.george}@univ-lemans.fr

Abstract. This article addresses the design of activity generators in the context of serious games for declarative knowledge training. Designing these generators is a complex task requiring specific knowledge and guidance, which falls under the scope of research in engineering. Our proposal is a design and implementation framework for generators of training game activities. The framework is a model and metamodel oriented software infrastructure that must be extended for domain specific facts. The resulting generators provide adapted and varied training activities in the form of dungeon levels for Roguelite-oriented games.

Keywords: Generation · Adaptation · Serious Games · Modelling.

1 Introduction

Although the memorization of declarative knowledge, such as laws, facts, and rules, necessitates repetition for both short and long-term retention [13], this process can quickly become tiresome for learners [25]. In addition, educational games that involve repetitive tasks and present challenges that are not adapted to learner-players' knowledge and skills can lead to feelings of boredom [26], and ultimately, may cause learners to give up the task. As a result, educational games aimed at declarative knowledge (DK) should propose activities that are both 1) varied and 2) tailored to learner-players.

The literature in cognitive psychology indicates that retrieving concepts or facts through testing enhances their long-term acquisition [4]. In our context, DK training is defined as a type of retrieval practice that involves repetitively presenting learner-players with various forms of questions about facts. In educational games, questions about facts can have several forms. Our vision is to consider an *activity* as an *educational game level* consisting of a sequence of questions about facts aimed at a common training objective, i.e., each question representing both a training task and its representation as a game-oriented task.

Generation is a technique that facilitates the automatic creation of content both used in Technology Enhanced Learning (TEL) contexts (e.g., pedagogical scenarios, learning paths) and games (e.g., levels, stories, dialogues). This technique is based on a set of structured data and rules defined through algorithms.

However, the concept of generation is underexplored in the field of TEL [2], particularly when considering both educational and game dimensions conjointly.

To establish the boundaries of the research object (the generator) and to facilitate its examination, it appears pertinent to distinguish the training game component tasked with generating the formal description of training activities from the *game engine* component, which is accountable for their interpretation. This distinction also extends to the other shared components that contribute to the overall adaptive process, such as tracking the progression of learners and updating the profiles of learner-players, among others. This approach allows for a more focused and effective study of each individual component and their respective roles within the system.

Designing generators of varied and adapted activities aimed at declarative knowledge training poses numerous challenges in terms of specification and implementation, requiring the expertise of various stakeholders: serious game developers (e.g., design choices, technologies choices), experts in the targeted didactic domain (i.e., facts to work on, training strategies, adaptations strategies for learners), and video game experts (e.g., gaming knowledge, game design knowledge). Such design is a complex problem in research in engineering that cannot be reduced to a mere computer engineering issue [27].

Accordingly, our work relies on an exploratory study aimed at characterising generators (research object) and proposing tools and techniques to facilitate their design (from a computer science perspective) and development. Our contribution is a framework (i.e., a software infrastructure comprising models and tools) designed to guide the design and implementation of generators of varied and adapted game activities for declarative knowledge training. The proposed framework is based on a Model-Driven Engineering (MDE) approach [12].

2 Related Works

Generating adapted activities for training games requires looking up three research axes: game design (i.e., specifically, the design of game activities), adaptation, and generation. It is important to emphasize that **the design of game activity generators does not require designing serious games**. Generators are independent software components, which can be evaluated independently of a game. However, creating game activities requires making upfront choices in terms of game design.

Game & Serious Game Design. Numerous methods, frameworks, and approaches exist to guide the design of games and serious games [1, 7]. However, the majority of existing works either concentrate on analysing existing games or assisting in the overall design (i.e., high-level design/requirements specification) of the game to be developed [11]. To our knowledge, none of these works address the concept of adaptation to learners or players, nor explore the concept of generation.

Adaptation in TEL. Adaptation can be implemented in various ways and may have one or multiple targets (e.g., gaming preferences, learning content,

difficulty). In our work, we focus on individualization/personalization[1], which can be defined as a means to cater to students' specific skills or abilities, including special needs, by providing learning progressions tailored to their needs [10]. Some works propose to adapt gamified platform and educational content [8, 22]. However, adapting existent content and building adapted content are two different tasks with different challenges. Other works address the design or aid to the design of adapted game content. Natkin et al. [23] propose a quest recommendation system based on a user model. Marne et al. [20] presents an authoring tool designed to assist educators in creating adaptive and non-linear game scenarios. Bontchev et al. [3] propose a framework including a learner-player model for personalisation of labyrinthine serious games. These works share a common element: they rely on models and learner results for adaptation. Even though, the forms of adaptation differ, and the models used are context-specific, these approaches serve as a valuable source of inspiration for defining one's own models.

Generation in TEL & in Games. Content generation has been addressed from three angles in TEL: non-adapted, tailored to the learner, and tailored to the player. Diwan et al. [9] proposed a model to generate pedagogical paths from open-sources resources. Carpentier et al. [6] proposed a three model-based approach (i.e., domain, activity, causality) to dynamically generate learner-adapted scenarios (i.e., learners' skills and pedagogical needs) in virtual environments. Sehaba and Hussann [24] proposed a general architecture to generate learner-adapted game scenarios based on several models: domain model (i.e., domain and its relations), learner model (i.e., learner information, motivation, skills, etc.), presentation model (i.e., structure of scenarios), and game model (i.e., game description and relations with pedagogical elements). Laforcade and Laghouaouta [14] proposed an MDE approach, based on [24], to specify generators of activity sequences adapted to learners' individual needs. Callies et al. [5] proposed an adaptive model-based architecture aimed at generating adapted pedagogical plans with adapted non-player character behaviour based on players' actions.

Observations. Most of these works use structured data such as models or ontologies to adapt and generate content. Regarding the adaptation of educational content, with or without generation, the main used structured data are: data on the knowledge and learning content, data on the learner (e.g., knowledge, skills), data describing the activity structure (to generate) and adaptation rules. Regarding the adaptation of game content, with or without generation, the main used structured data are: data describing the game, data describing the activity structure, data on the player (e.g. game preferences, progress). Studies focusing on generating adapted game content need to include details about how learning and game elements are connected. However, these relations are often not clearly specified in models and are in general directly incorporated into the generation algorithm. Additionally, these models are usually specific to a certain didactic domain, making them challenging to use in different contexts. Furthermore, adaptation is typically approached from a singular viewpoint, where content is adapted either to the learner or the player but hardly to both simultaneously.

[1] In this article both word are considered as synonyms.

3 Research problem & Positioning

Building on our earlier observations, our research problem is: **How to guide the design of generators of varied and adapted game activities for declarative knowledge training?** From this, several questions arise: *How to propose an approach generic enough to address DK independently of a specific educational domain? What is a varied and adapted game training activity? What training and game elements these activities are composed of? How can the game and training elements be associated coherently? How to structure these elements and their relationships to guide the generation of coherent activities? How to specify this information computationally to build activity generators?*

Working on DK opens up the potential to reuse models across various didactic domains. In our context, training involves the repetition of various types of questions about facts, which are repeatedly presented to learner-players. Thus, the structure of an activity requires to identify a compatible game genre, i.e., capable of maintaining players' engagement while offering repetitive but varied gameplays. The Roguelite genre meets these requirements [19]. This genre is primarily characterized by the procedural generation of dungeons with pseudo-random content, permanent death (each avatar death requires the player to start a new game), and limited possession of unlockable game elements (e.g., characters, objects, power-ups...) facilitating progression in the next *run*. Therefore, a training game activity is a *dungeon*, a set of interconnected rooms that the player explores with an avatar and in which the training takes place.

To reduce the feeling of repetition, activities must be varied and personalized. Given that adaptation to both dimensions (i.e., educational and gaming) is rarely addressed conjointly, our objective is to personalize considering three perspectives: teacher (i.e., training strategy, pedagogical and didactic choices for learners), learner (i.e., knowledge, progress/results in training), player (i.e., game preferences). Adaptation will consider various aspects, including encountered facts, question formulation, the quantity of facts queried, sequencing based on the learner's level or previous results, and training parameters defined by the teacher. Moreover, Roguelites often feature a shop with a purchase/activation mechanism of items, enabling players to make decisions that can impact the generation of levels. Therefore, adapting preferences will involve focusing on the ability to activate/deactivate equipments unlocking various gameplays that can address the questioned facts, by directing or moving objects, for example. This design choice allows players to disable gameplays that they dislike in the game.

In addition to adaptation, the variety of activities is necessary to reduce the feeling of boredom generated by repetition. Roguelites are based on a procedural generation mechanism with randomness, making each game level different in terms of content (e.g., dungeon structure, elements and their positions in rooms, etc.). In our context, our approach involves modelling elements (educational and gaming) with a certain degree of variability. For example, facts are modelled so that incorrect choices are defined with each dungeon generation. Alternatively, gameplays are defined using abilities to allow for varying the selected game elements with each dungeon generation. The objective is to enable the generation

algorithm to pseudo-randomly choose, while maintaining the consistency of the training and gaming elements in a dungeon.

A Roguelite-oriented activity generator for DK training is a software component (i.e., software element intended to be incorporated as a detached component) whose algorithm allows for the creation of diverse activities based on three types of input data: information about training, information about the game, and information about the learner-player. These software components output detailed descriptions of activities (i.e., dungeon levels) tailored to learner-players.

Our proposition consists of a **design and implementation framework** of Roguelite-oriented, adapted and varied **activity generators** for DK training. This framework is a domain-specific extensible software infrastructure based on a conceptual design approach [16] (i.e., captures common or generic elements, while the extension mechanism guides the addition of domain-specific ones). Our proposal fits into the context of a Model-Driven Engineering [12] approach.

4 Design Framework of Activity Generators

This section introduces the proposal: a software infrastructure to guide the design and implementation of game activity generators for DK training.

4.1 Framework Overview

The proposed framework is a conceptual and software infrastructure composed of a set of models and tools to guide the implementation of varied and adapted activity generators. The produced generators are software elements that can be considered as components of a DK training game in the Roguelite genre. These generators enable the creation of a new training activity upon each request, i.e., a detailed textual description of a dungeon level for a given learner-player. These descriptions must then be interpreted by a game player to provide a playable game level to the learner-player.

This framework is divided into two parts: a set of generic components and rules allowing extensions to specific didactic domains. In the context of an MDE approach, the generic components comprise models, metamodels, and an algorithm for generating varied and adapted activities based on these models. Later on, we will elaborate on the fact that certain elements cannot be managed or generated independently of the didactic domain. Therefore, the extension rules enable the creation of domain-specific models, metamodels, and code necessary for the generator. Figure 1 illustrates the components of the framework.

In the context of MDE, the generation algorithm functions as a model transformation. Consequently, the generators expect input conforming to each input metamodel of the framework (i.e., concrete data: knowledge to work on, available game elements, learner-player's result/progression, etc.) and produce output that conforms to a metamodel of the framework. These generators include source code corresponding to the framework's metamodels, enabling them to read and utilize the data within the models. Additionally, they incorporate the generic

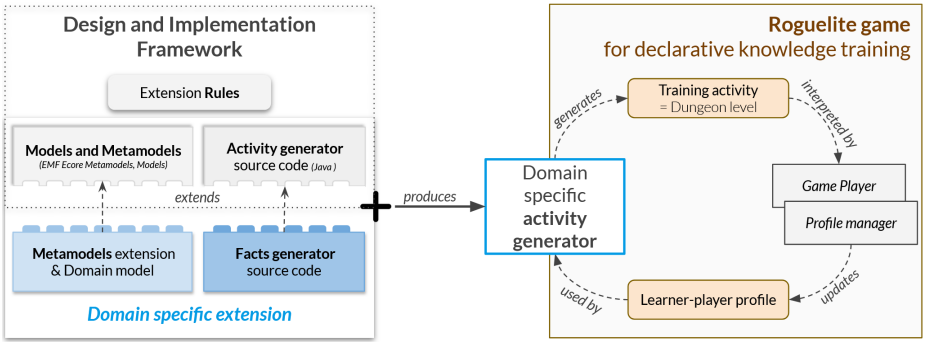


Fig. 1: Design & Implementation framework overview.

activity generation algorithm (i.e., domain-independent) as well as the models and facts generation algorithm developed during the framework extension based on the didactic domain.

As mentioned earlier, the generated activities must be varied and adapted. Furthermore, the framework should be domain-independent while still allowing the specification of domain-specific knowledge. Consequently, both the framework and the generators it produces must respect certain properties:

FP1 the framework must allow for the expression of various didactic domains (extensibility);
FP2 the framework must allow for the consideration of teachers' viewpoints on the training of individual learners (pedagogical/didactic adaptation);
GP1 the generators must provide activities adapted to the level and results of the learner in their training path (pedagogical adaptation);
GP2 the generators must provide activities adapted to the player's game preferences (gaming adaptation);
GP3 the generators must provide varied activities in terms of training and game elements (variety).

Research method. This framework was developed as part of an exploratory research focused on the iterative design of an initial case study, involving a user group and the participation of experts. Therefore, it is an inductive method where the results obtained, related to the initial domain (i.e., multiplication tables), are generalized and re-evaluated in this domain, as well as in two other domains, i.e., judo techniques/referee gestures and history-geography facts required for a French exam (*Diplôme National du Brevet*).

4.2 Models & Metamodels

As mentioned earlier, generation requires various structured information to produce varied and adapted activities. In previous work, we proposed a set of six conceptual models necessary for generating *Roguelite*-oriented game activities for

DK training [16]. These models were designed to be generic (i.e., applicable to every DK regardless of a specific domain). Consequently, certain models contain extension points, i.e., parts to be extended based on the targeted domain.

Knowledge Model. This model encompasses all raw facts (i.e., knowledge) that learners need to retain. Facts can take the form of textual information, like "World War I happened between 1914-1918", or visual information, such as the position of a region in France. This model is entirely domain-dependent, as facts rely on the specific didactic domain being targeted.

Training Model. This model describes a training path for a learner through objectives ordered by prerequisites relations, levels, and training tasks. This structuring was defined with the help of experts as a continuation of an exploratory research initiated in [15]. Four generic task types were defined: *Completion* (i.e., complete facts with missing elements, e.g., $3 \times 3 = 12$); *Identification* (i.e., identify if facts are correct or not, e.g., true/false: $3 \times 5 = 12$?); *Membership Identification* (i.e., identify facts that share a given property, e.g., {3,5,7,6} which are results of table 3?); *Ordering* (i.e., ordering facts based on a heuristic). A training path is defined by a teacher for a learner or a group of learner (i.e., each learner have a training path = training progression defined by the teacher specifically adapted to the learner).

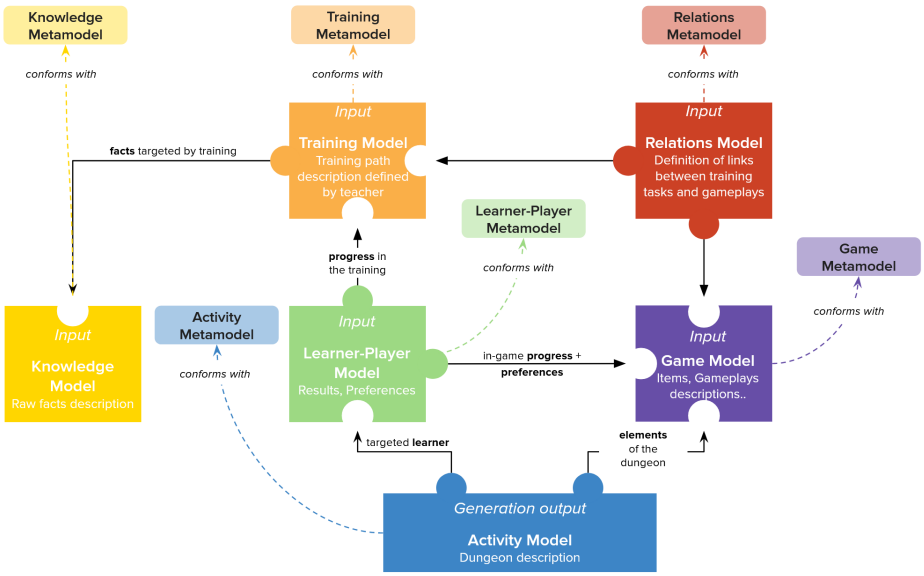


Fig. 2: Illustration of the six proposed models for generating game activities.

Game model. This model describes every game elements available for the generation algorithm, such as: gameplays, room types, game progression, game elements (e.g., block to push, jar to move), etc. Game elements are described

using abilities: behaviour of elements such a block can be pushed → *pushable*, a chest can be open → *openable*. This allows the description of more variable gameplays using abilities instead of static game elements.

Learner-Player Model. This model describes the learner progress in their training path (e.g., objective/level/task achieved), their results (e.g., questioned facts correctly or incorrectly answered, percentage of questions encountered, percentage of questions correctly answered), their current game level, and their game preferences (i.e., items they can buy and activate/deactivate which unlocks abilities) that are required to generate activities adapted to them.

Relations Model. This model specifies the relations between training task and gameplays. This model is used by the generation algorithm to build coherent activities. DK training is often done through questionnaires and quizzes. Therefore, we previously defined a systematic method, based on the use of digital questionnaires formats as a pivot, to establish conditional relationships between training tasks and gameplays [18]. Once the conditions are established, the relations model can be specified using the generic task types (i.e., Completion, Identification, Membership Identification, Ordering) independently of any didactic domain. Therefore, this model only needs to be specified once.

Activity Model. This model describes the activity structure: a dungeon = set of interconnected rooms that have positioned elements such as a gameplay, questioned facts/a task, etc. This model is specified by the generator each time the game (i.e., independent component of the framework which calls the generator for new levels) requires a new level.

These interconnected conceptual models defined, cf. Figure 2, were translated into metamodels (i.e., computerized representation in a Model-Driven Engineering approach) using the *Eclipse Modeling Framework* (EMF) in order to be "instantiated" into models (i.e., in conformance with the dedicated metamodels) that can be interpreted for generation.

4.3 Activity Generation Algorithm

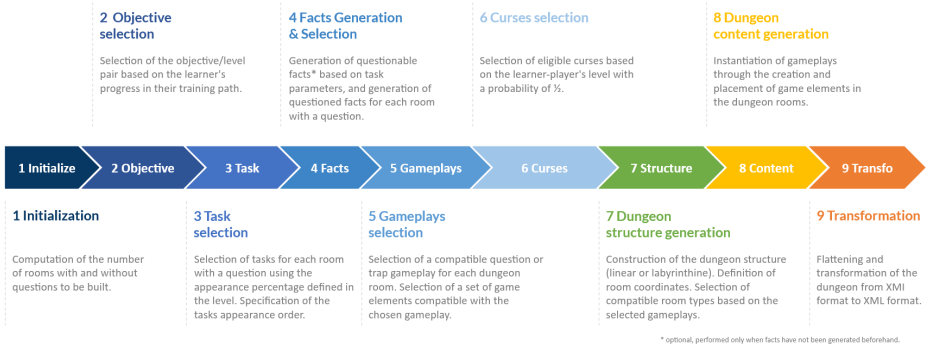


Fig. 3: Activity generation algorithm steps.

The activity generation algorithm is implemented in Java and occurs in multiple incremental steps (similar to [14, 24]). The algorithm is divided into four parts, broken down into nine steps: 1) selection of all dungeon elements (i.e., steps one to six), 2) creation of the structure (i.e., seventh step), 3) instantiation of the dungeon (i.e., eighth step), and 4) transformation of the dungeon (i.e., ninth step). Figure 3 outlines the nine steps of the algorithm.

First, it computes the number of rooms with and without questions. Next, it selects an eligible objective/level pair from the learner's training path based on their previous results and progress. Objective/level pairs are eligible either outright or as soon as the prerequisites are satisfied (i.e., the learner's results meet the conditions of the prerequisites). Currently, a pair is selected randomly from the set of eligible ones. However, a strategy, defined by teacher or experts for example, could be implemented to select a pair [21]. Then, the algorithm chooses the tasks for each question room in the future dungeon based on the selected level (i.e., each level is described by a set of training tasks to complete). These tasks are selected based on their percentage of appearance and learners' progress in their training path (i.e., once a task has been completed, it no longer appears in the dungeons, and its percentage of appearance is distributed proportionally to the other tasks). To continue, it generates the questioned facts (i.e., facts to be questioned in the dungeon) for each room with a question in the future dungeon. Questioned facts are build based on the parameters on the tasks. Currently, wrong choices are chosen randomly. However, heuristic or strategies could be used [21] such as using previous learners' wrong answers. After that, it selects gameplays and corresponding game elements for each room. Since player's preferences consists of bought and activate/deactivate items that unlocks abilities which describes gameplays, selected gameplays are restricted based on players' preferences. Therefore, if players do not like a specific interaction, they can deactivate the ability associated to that given interaction except for default ones. Subsequently, it selects the curses based on the player's level (i.e., a game mechanic often implemented in Roguelites, not detailed in this article). It then proceeds to build the shape of the dungeon: labyrinthine or linear, room types, etc. Based on every previously made choices, it follows by initializing the dungeon elements. Figure 4 presents a step-by-step example of the algorithm.

4.4 Extension rules

This framework needs to be extended at the level of metamodels, models, and code to allow the design of a domain-specific generator. In the presented metamodels, extension points are denoted by abstract classes (i.e., without concrete subclasses). Three metamodels are targeted for this extension: the knowledge metamodel, the training metamodel, and the learner-player metamodel. Facts take on different forms depending on the didactic domain. Moreover, levels and training tasks may have domain-specific parameters. Also, the missing element in a completion task (i.e., operand/table/result for multiplications and events/date-period for historical dates) depends on the domain.

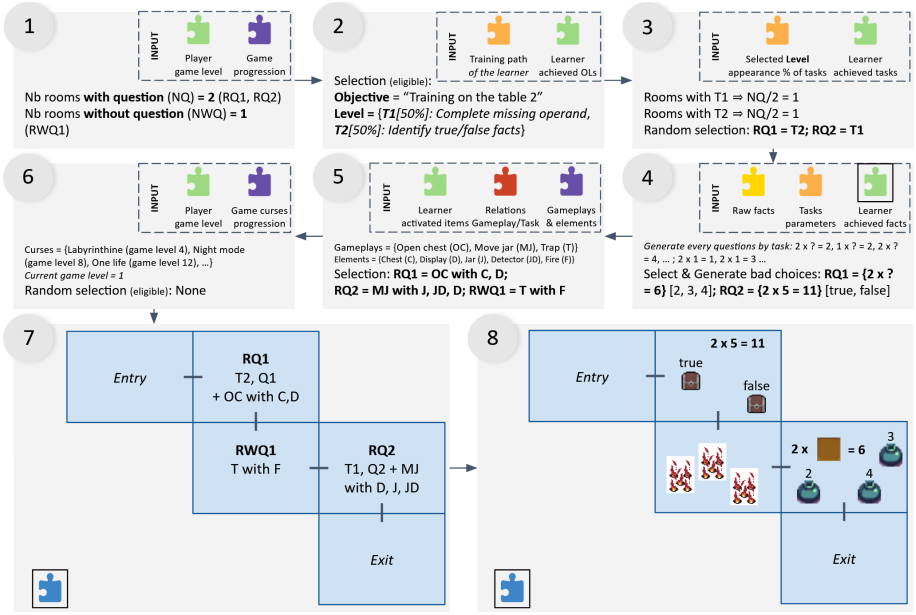


Fig. 4: Step-by-step example of the generation algorithm. Puzzle pieces colours correspond to Figure 2 and puzzle pieces with borders present data modified or created by the algorithm.

Consequently, generating facts is also domain-dependant. In order to reduce that dependency, our previous work [17] proposes a generic approach to model questioned facts and a generic algorithm to map questioned fact and gameplays (i.e., used in the step eight). Therefore, our framework guides the creation of generic questioned facts based on domain specific raw facts using a *template-method* design pattern. This pattern allows for a part of the fact generation algorithm to be independent of the domain, with domain-specific extensions.

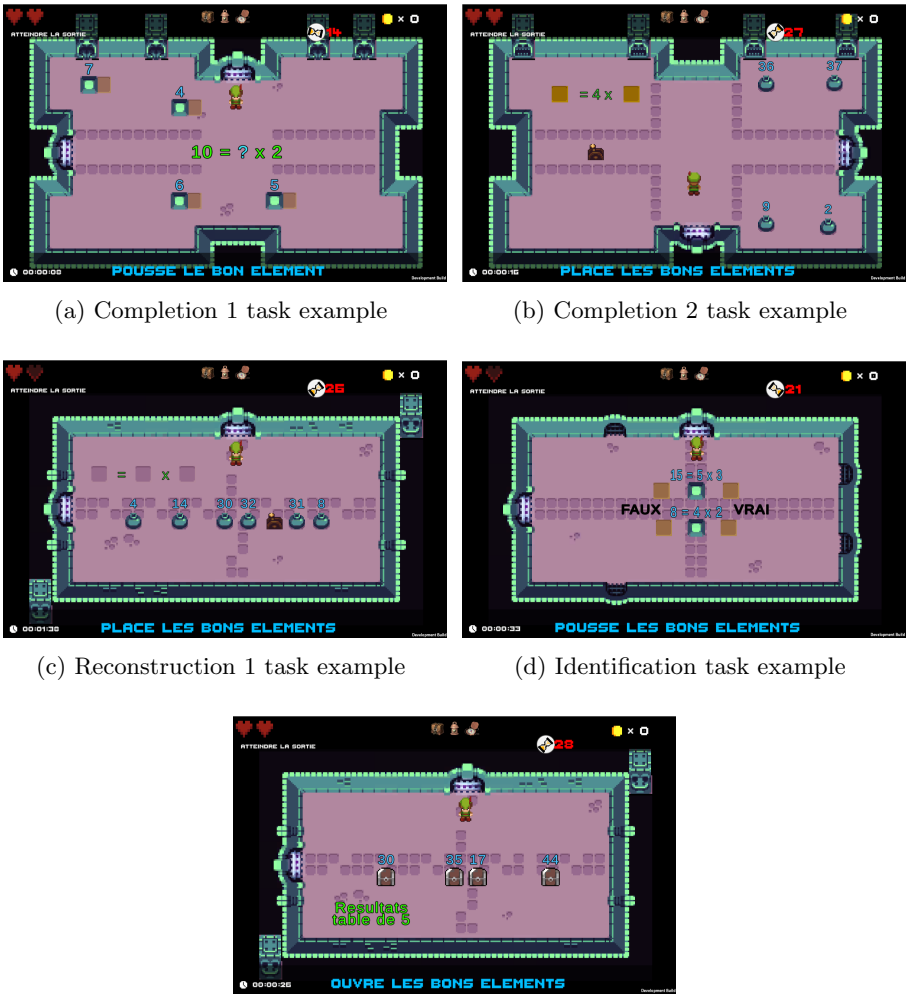
Finally, each piece of information must be declared in dedicated models conforming to each metamodel. Therefore, a model needs to be specified for each metamodel, except the activity metamodel (as it is produced by the generator).

5 Application & Evaluation

Our proposed framework and the produced generators must respect some defined properties (cf. Section 4.1). Accordingly, the evaluation involves validating the compliance to the specified properties. The evaluation of FP1 and FP2 is carried out through a proof of concept. To assess FP1, at least two extensions (i.e., two didactic domains) need to be designed. In this regard, we extended the framework to three didactic domains: multiplication tables, judo techniques/referee gestures, and history-geography facts of the *Brevet des Collèges*, a French exam,

(not presented in this article). To assess FP2, at least two training paths must be defined by different teachers. Therefore, we asked two teachers to design training paths. On the other hand, the generators must adhere to three properties. This was evaluated through automated system tests (JUnit), which involves verifying the adherence to a set of given intentions.

5.1 Framework properties evaluation



(a) Completion 1 task example

(b) Completion 2 task example

(c) Reconstruction 1 task example

(d) Identification task example

(e) Membership ID task example

Fig. 5: Generated dungeons rooms (in french) for multiplication tables training interpreted by the *game engine*.

As part of an exploratory study started in 2021 [15], we defined with mathematics experts five training tasks for multiplication tables: Completion 1, i.e., complete a fact having a missing element (e.g., $3 \times ? = 15$, $15 = ? \times 5$); Completion 2, i.e., complete a fact having two missing elements (e.g., $3 \times ? = ?$ with a set of given choices [3, 6, 5, 15]); Reconstruction, i.e., correctly replace every element of a fact (e.g., $? \times ? = ?$ with a set of given choices [3, 6, 5, 10, 15]); Identification, i.e., identify the correctness or incorrectness of facts (e.g., $3 \times 5 = 15$, true or false ?); Membership Identification, i.e., identify elements that share a given property (e.g., [3, 5, 9, 14, 21] which are results of table 3?).

In addition, we also defined parameters for the "levels" concerning the construction of questionable facts, such as the form of constructing tables "operand $\times$ table or table $\times$ operand", the position of the equal symbol "on the left or on the right", and the min/max interval of the questioned multiplications (i.e., a teacher may want to work on the table of three only from 1 to 5: 3×1, 3×2, 3×3, 3×4, 3×5). Following the extension rules, we specified the knowledge model and developed the fact generators for each of the five tasks. Currently, we have two training paths created by two teachers, for two different classes (CE1 and CE2, i.e., French elementary class groups), with average levels modelled.

As part of the AdapTABLES project, aimed at building a game for multiplication tables training, a *game engine* has been developed to interpret the XML files produced by the generators and enable users to play. Figure 5 presents generated dungeon rooms interpreted by the *game engine*.

5.2 Generators' properties evaluation

Learner's adaptation. The evaluation of learner adaptation involves verifying that the generated activities align with predictions deducible from an analysis of the learner's associated training path. The conformity of the activity is based on two levels of analysis: A) the selection of the objective/level pair to work on (i.e., selected objective/level must respect the progression describe in the learner's training path), and B) the tasks present in the dungeon must be consistent with the selected level, learner's progression, and the desired appearance percentage. To evaluate A), we check that: the selected objective/level pair is eligible; for 150 generations, all eligible objective/level pairs have appeared at least once; for 150 generations, none of the non-eligible objective/level pairs have appeared; the levels with the achieved percentage of encountered facts and success are considered eligible (and therefore never appear in the dungeons). To evaluate B), we ensure that the repartition of tasks aligns with their appearance percentage, considering the number of rooms in the dungeon, which is calculated based on the player's current game level.

Player's preferences adaptation. The player's game preferences are described through purchasable and activable equipments. These equipments unlock abilities, thus enabling the unlocking of new gameplays. Therefore, to evaluate player's preferences adaptation, we verify that: if the learner-player has made no purchases, only default gameplays are present in the dungeons; if the learner-player has purchased all possible equipment but has not activated any, only de-

fault gameplays are present in the dungeons; if the learner-player has purchased and activated all possible equipment, then all gameplays appear at least once in a dungeon; if the learner-player has purchased and activated some equipment, the gameplays associated with the purchased and activated equipment appear at least once in a dungeon at some point, and the others never appear.

Activity variety. An interpreter or game player for the level of play (i.e., activities/dungeons) was developed. This interpreter allows: 1) translating the generated XML into playable dungeons, 2) visualizing the dungeon maps, and 3) playing. This interpreter allows visualizing a map in which each room is coloured based on the associated task, and access between each room is defined by a small line (i.e., distinguishing labyrinthine from linear dungeons). Therefore, an initial evaluation consisted of a manual assessment, involving generating multiple dungeons for the same level and comparing their maps graphically. Figure 6 presents two maps of dungeons generated for the same progression. Other automatized tests were implemented to evaluate variety of gameplays, which showed that various gameplays were chosen per task (not detailed here).

Fig. 6: Two dungeons maps (same objective/level pair & same learner-player).

5.3 Extension rules evaluation

An experiment has been conducted with an engineer with two main objectives: 1) evaluating the usability of the framework (i.e., can the framework be extended by an engineer independent of the project?) and 2) improving the clarity of the provided guidelines (i.e., are the guidelines self-sufficient for creating an extension). To that extent, the engineer was provided with the specification of two tasks for solar system facts training[2] and had to extend the framework by creating the necessary components (metamodel, models, facts generators). The experiment was realised over a day and a half. Since our engineer had no notions of MDE, an MDE expert was available to answer questions and guide the use of the EMF. Encountered problems with the guidelines or incomprehension were discussed directly. At the end, the engineer had to answer a short questionnaire about the difficulty of each step realised. Overall, the engineer found the extension to be well guided and quite easy to realise. The required knowledge of MDE was considered low. However, some necessary clarifications and

[2] Since an engineer is not an expert in didactic or pedagogy our choice was to provide the specification of data related to the didactic domain.

improvements for the guidelines were highlighted during the experiment and the debugging (i.e., understanding and correcting the errors created from the added code). Currently, we dispose of a set of possible errors and their solutions that will be added in the guide to facilitate that phase.

6 Conclusion

This article presented a framework for the design of a varied and adapted activity generator for DK training. It is an extensible (i.e., domain-independent but extensible to specific didactic facts) Model-Driven software infrastructure that targets activities for Roguelite-oriented games. The framework was extended to build three generators for: multiplication tables, judo techniques and arbitration gestures, history-geography facts.

Currently, the generator for multiplication tables is used in an educational game. This game has been tested in ecological conditions. Additional experimentation of the framework, involving engineers, is also scheduled for the future.

References

1. Amory, A.: Game object model version II: A theoretical framework for educational game development. Education Tech Research Dev **55**(1), 51–77 (Jan 2007)
2. 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)
3. Bontchev, B.P., Terzieva, V., Paunova-Hubenova, E.: Personalization of serious games for learning. ITSE **18**(1), 50–68 (May 2021)
4. Brame, C.J., Biel, R.: Test-Enhanced Learning: The Potential for Testing to Promote Greater Learning in Undergraduate Science Courses. LSE **14**(2) (Jun 2015)
5. 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)
6. 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)
7. Carvalho, M.B., Bellotti, F., Berta, R., De Gloria, A., Sedano, C.I., Hauge, J.B., Hu, J., Rauterberg, M.: An activity theory-based model for serious games analysis and conceptual design. Computers & Education **87**, 166–181 (Sep 2015)
8. Codish, D., Ravid, G.: Adaptive Approach for Gamification Optimization. Proceedings - 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, UCC 2014 pp. 609–610 (Jan 2015)
9. Diwan, C., Srinivasa, S., Ram, P.: Automatic Generation of Coherent Learning Pathways for Open Educational Resources. In: Transforming Learning with Meaningful Technologies, vol. 11722, pp. 321–334. Springer Int. Pub., Cham (2019)
10. Grant, P., Basye, D.: Personalized Learning: A Guide for Engaging Students with Technology. International Society for Technology in Education (2014)
11. Junior, R., Silva, F.: Redefining the MDA Framework—The Pursuit of a Game Design Ontology. Information **12**(10) (Sep 2021)

12. Kent, S.: Model Driven Engineering. In: Integrated Formal Methods. pp. 286–298. Springer Berlin Heidelberg (2002)
13. 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 (Jan 2013)
14. Laforcade, P., Laghouaouta, Y.: Generation of Adapted Learning Game Scenarios: A Model-Driven Engineering Approach. In: Computer Supported Education - 10th CSEDU Funchal, Madeira, Portugal. CCIS, vol. 1022, pp. 95–116. Springer (2018)
15. Laforcade, P., Mottier, E., Jolivet, S., Lemoine, B.: Expressing adaptations to take into account in generator-based exercisers: An exploratory study about multiplication facts. In: 14th International Conference on Computer Supported Education. Online Streaming, France (Apr 2022)
16. Lemoine, B., Laforcade, P.: Generator of personalised training games activities: A conceptual design approach. In: Games and Learning Alliance - 12th International Conference, GALA 2023, Dublin, Ireland, November 29 - December 1, 2023, Proceedings. LNCS, vol. 14475, pp. 321–331. Springer (2023)
17. Lemoine, B., Laforcade, P.: Mapping facts to concrete game elements for generation purposes: A conceptual approach. In: Games and Learning Alliance - 12th International Conference, GALA 2023, Dublin, Ireland, November 29 - December 1, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14475, pp. 342–352. Springer (2023)
18. Lemoine, B., Laforcade, P., George, S.: An approach for mapping declarative knowledge training task types to gameplay categories. In: Computer Supported Education. pp. 47–68. Springer Nature Switzerland, Cham (2024)
19. Lemoine, B., Laforcade, P., George, S.: Designing declarative knowledge training games: An analysis framework based on the roguelite genre. In: Computer Supported Education. pp. 69–92. Springer Nature Switzerland, Cham (2024)
20. Marne, B., Carron, T., Labat, J.M., Marfisi-Schottman, I.: MoPPLiq: A Model for Pedagogical Adaptation of Serious Game Scenarios. In: IEEE 13th ICALT 2013, Beijing, China, July 15-18, 2013. pp. 291–293. IEEE Computer Society (2013)
21. Melero, J., El-Kechai, N., Yessad, A., Labat, J.M.: Adapting learning paths in serious games: An approach based on teachers' requirements. In: 7th International Conference, CSEDU 2015, Lisbon, Portugal, May 23-25, 2015, Revised Selected Papers, CCIS, vol. 583, pp. 376–394. Springer (2016)
22. Monterrat, B., Yessad, A., Bouchet, F., Lavoué, É., Luengo, V.: MAGAM: A Multi-Aspect Generic Adaptation Model for Learning Environments. In: Data Driven Approaches in Digital Education, vol. 10474, pp. 139–152. Springer International Publishing, Cham (2017)
23. Natkin, S., Yan, C., Jumpertz, S., Marquet, B.: Creating Multiplayer Ubiquitous Games using an adaptive narration model based on a user's model. In: Proceedings of the 2007 DiGRA International Conference: Situated Play, DiGRA 2007, Tokyo, Japan, September 24-28, 2007. Digital Games Research Association (2007)
24. Sehaba, K., Hussaan, A.M.: GOALS: Generator of adaptive learning scenarios. IJLT **8**(3), 224 (2013)
25. Smith, R.P.: Boredom: A review. Human factors **23**(3), 329–340 (1981)
26. 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)
27. 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)