An Approach for Mapping Declarative Knowledge Training Task Types to Gameplay Categories

Bérénice Lemoine, Pierre Laforcade, Sébastien George

An Approach for Mapping Declarative Knowledge Training Task Types to Gameplay Categories

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

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

Abstract. Training on declarative knowledge (DK) requires repetition, which can quickly become boring for learners. Consequently, games targeting such training must offer a wide variety of activities in order to keep learners-players engaged. Designing such situations remains a challenge because of the inherent entanglement of didactic elements and game elements. This chapter is an extended version of [14], which tackles the need to map training tasks with different gameplays for the design of relevant gameplay-oriented training activities. The proposed approach was identified during the design of a Roguelite-oriented training game for multiplication tables and has intentionally been specified towards a genericness purpose by using domain-independent task types and abstract gameplays. This chapter details the specification of task types (i.e., abstracted from two didactic domains) and abstract gameplays, the method used to identify the approach, and the resulting mappings when applied to our specific context.

Keywords: Serious Game · DK Training · Didactic-Game Mapping.

1 Introduction

The design and use of serious games has become a common practice this last decade [4]. However, due to their lack of gameplay, most learning games fail to be seen as real games [16]. Gameplay can be defined as the *fun things that can be controlled, decided, and done by players* [16]. Although combining real games' fun and educational content is not easy [16], it is a key component of a good learning game design. Correctly associating game and learning elements is mainly a difficult task because multiple context-dependant (i.e., didactic domain, targeted knowledge or game genre) variables must be considered.

Declarative knowledge (DK, i.e., knowledge about facts, laws, statements) are known to require repetition for encouraging their memorization, generalization, and retention [10,18]. Test-Based Learning (TBL) is defined in cognitive psychology as the idea that the process of retrieving (i.e., remembering) concepts or facts increases their long-term retention. Retrieval practice (i.e., repeated retrieval)

is a form of TBL that has been shown to improve long-term retention [18]. In addition, research suggests that the benefits of retrieval practice are not linked to a specific implementation (i.e., various tests formats enhance learning) [2]. In our work, *training* is defined as a form of retrieval practice that consists of providing learners with various forms of questions about facts repeatedly.

As learning games aimed at DK training offer repetitive training sessions, a deeper commitment from learners is required. Accordingly, training games should provide a wide variety of activities in terms of situations, game mechanisms or gameplays. Hand-crafted design of varied activities limits the scope for variety. A possibility to design a wide variety of activities is the use of automatic content generation. Content generation is a common technique in game development, especially in Roguelike and Roguelite games, to create unique game levels (e.g., different shapes, elements, elements' position). Accordingly, designers of training games are faced with different stakes, such as the automatic generation of activities and the design of the mechanism enabling the variety of activities. Our previous work, presents *Roguelite*, a well known and liked game genre, as an adequate genre for DK training [13]. *Hadès, Rogue Legacy, Binding of Isaac* are famous Roguelites. These games are dungeon-crawler games: players must explore dungeons, i.e., interconnected rooms where actions take place. Roguelites are based on a mechanism called permanent-death which involves repetition, i.e., players must start a new playthrough each time their avatar dies. Other characteristics of Roguelites games are the procedural generation of dungeons with randomized content ($\rightarrow$ variety) and the limited retention of unlockable items (e.g., avatars, powerups, equipments). In short, a Roguelite for DK training will successively propose generated dungeon levels to the learner-player, wherein they will be challenged to answer task-oriented questions.

Our research interests concern the generation of varied Roguelite activities for DK training. Our aim is to offer a variety of gameplays. The definition of gameplay, *fun things that can be controlled, decided, and done by players* [16] can be refined as: descriptions of contextualized actions that players can perform to interact with the environment, through their avatar, in order to answer questions. Since training games involve several domain-specific parametrized training tasks, numerous gameplays must be identified for each of them. Identifying how different training tasks can be implemented using these different game concepts requires conceptualizing and addressing a transdisciplinary Technology Enhanced Learning (TEL) problem: *how can didactic knowledge be mapped to different gameplays?*

This challenge emerged during the design of a Roguelite-oriented learning game to train multiplication tables. First, our approach is to address the challenge at a higher level of abstraction (task types instead of domain-specific tasks, and game categories instead of practical gameplays). This enables a more generic, domain-independent approach. In our previous work [14], task types were abstracted from a single didactic domain. In this chapter, task types are abstracted from two didactic domains. Second, the central thrust of our approach is to use a dedicated pivot to help identify the source (task types) and target (game cate-

gories) parameters whose values will guide the elicitation of practical mappings. This chapter explains how we identified this approach (method), what it consists of (proposition) and what mappings are obtained in our context (application). We assume that this approach is sufficiently generic and reusable to help multidisciplinary design teams identify and map gameplays for their specific tasks (contribution).

2 Elements for Training Game Activities

Our overall objective is to identify approaches, models and processes to help multidisciplinary teams design activity generators for DK training. But, as previously mentioned, building activities combining fun and educational content is not easy [16]. Prensky [16] proposed a three-step process to create digital game-based learning: "(1) Find or create a game with great gameplay that will engage our audience, (2) Find the learning activities and techniques that will teach what is required (doing each with the other in mind), and then (3) successfully blend the two". Accordingly, our work began with the identification of training tasks and possible gameplays with the experts.

2.1 Task Types for DK Training

An *exploratory research* (partially presented in [12]) conducted with 2^{nd} to 6^{th} grade teachers and mathematics experts led to the identification of training tasks for multiplication tables training, such as: *complete a fact where the result is missing* (e.g., $3 \times 5 = ?$), *complete a fact where the operand is missing* (e.g., $3 \times ? = 15$), *decide if a fact is correct* (e.g., $3 \times 5 = 15$), *identify the results of a table* (e.g., $[5, 6, 9, 12, 8]$ which are results of table 3?). These tasks also embed specific parameters to determine which facts to take into account, how to construct them and how to answer them. In addition, exchanges with history-geography teachers also led to the specification of training tasks for history-geography facts, such as: *place historical dates in chronological order* (e.g., World War II, Storming of the Bastille, Treaty of Rome), *name and locate countries of the European Union*, *decide if a fact is correct* (e.g., Did *World War II* happen between 1939 – 1945?).

An observation is that certain tasks appear to be similar in both domains. Therefore, in a perspective of genericness with other domain-related declarative knowledge, we expressed our tasks at a higher level of abstraction. Without being exhaustive, the following four types of task have been defined:

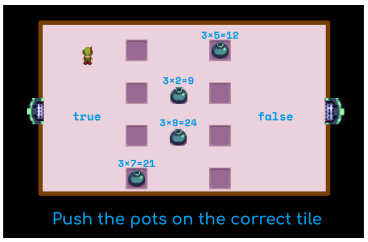
1. **Completion**: complete a fact that having missing elements (e.g., complete $3 \times ? = 15$, reconstitute $? \times ? = ?$ using elements in $[3, 6, 5, 10, 15]$, complete *World War II happened between* ? – ?);
2. **Order**: order facts based on a given heuristic (e.g., chronologically order: *World War II, Storming of the Bastille, Treaty of Rome*);
3. **Identification**: attest of the validity or invalidity of one or several facts (e.g., true or false: $3 \times 5 = 15$?, Did *World War I* happen between 1915–1919?);

4. **Membership Identification**: identify elements that share or not a given property (e.g., $[3, 5, 9, 12, 14, 21]$ which are results of the table 3? [France, Spain, England, Switzerland, Italy] which are part of the European Union?);

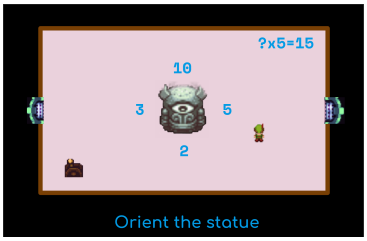
2.2 Gameplay Categories for Dungeon-Like Games

Prensky [16] stated that "Although learning games can fail as real games in many ways, the failure happens mostly commonly in their lack of gameplay". Consequently, our aim is to provide a wide variety of gameplays for each task. First, some ideas were discussed with the teachers. As a result, one constraint emerged: gameplays must be *simple*, i.e., interactions to answer must be quick. Then, informal interviews were conducted with game designers. The purpose was to gather ideas to design gameplay mock-ups. Moreover, a game prototype with a few gameplays was produced to try out some ideas and gather feedback.

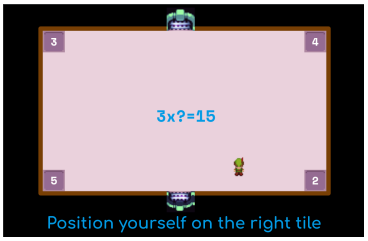
(a) Select example



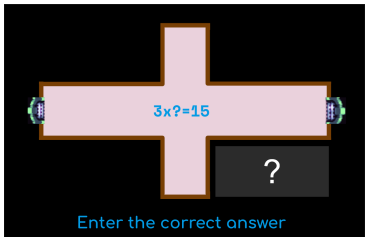
(b) Move example



(c) Orient example



(d) Position example



(e) Direct Response

Fig. 1: Example of mock-ups by gameplay categories [14].

After designing the mock-ups, an observation was made: *certain gameplays seemed to belong to the same category* (e.g., breaking a pot or opening a chest bearing an answer are similar ways of selecting an object). That observation is consistent with the game classification proposed by Djaouti et al.[5], which consists of describing games in terms of gameplay bricks (i.e., categories of actions that can be performed within the games). Consequently, further reflection resulted in the definition of 5 gameplay categories, cf. Figure 1, in our context (as with the task types, these categories do not claim to be exhaustive):

1. SELECT: select (e.g., touch, kill, break, open) objects wearing the correct answers, through avatar actions;
2. MOVE: correctly place objects at specific locations through avatar actions;
3. ORIENT: orient objects (e.g., rotate), through avatar actions, towards the correct answer;
4. POSITION: move the avatar to the necessary positions for choosing or typing the correct answers;
5. DIRECT RESPONSE: no action is required through the avatar, learners can directly type down their answer by using an input device (e.g., enter the correct answer through a keyboard).

3 Activity Generation: a Mapping Need

3.1 Research Question

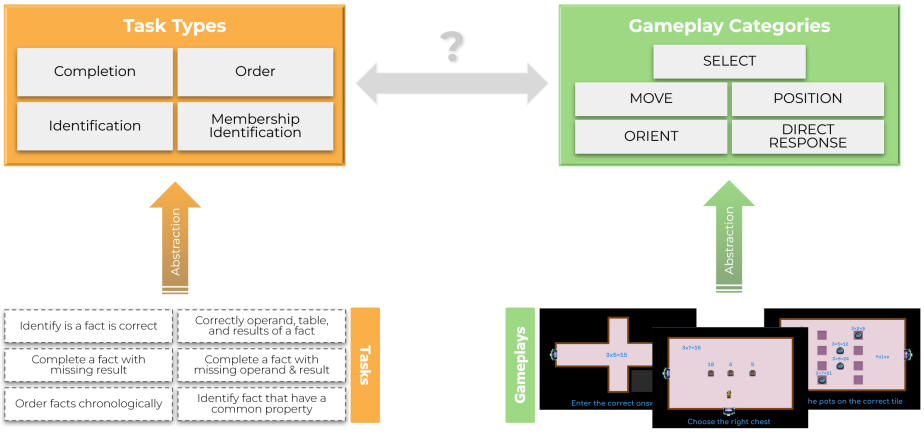


Fig. 2: Illustration of our research question.

As previously mentioned, our overall objective is to design activity generators. Activity generators are software components that automatically create content from structured data. Following Prensky's process [16], now that the elements have been determined, they must be correctly associated. Therefore,

our main question now is: *how to* ***determine and specify the relationships*** *between task types and game categories necessary to the design of learning game activities?* Indeed, knowledge about these relationships is essential at the design phase to guide the identification of practical gameplays for each specific task, and at the runtime to control the generation process. Our assumption is that answering this question at a higher level of abstraction (task types and abstract gameplays) will enable the reuse of the relationships in various declarative knowledge contexts.

This research question, illustrated in Figure 2, involves precisely answering the following questions: Which abstract gameplays are suitable for which task types? Is the mapping systematic or conditional? If conditional, how to find these conditions? According to Tchounikine et al. [21], this is typically a problem of research in TEL engineering falling into the "*elaborating powerful abstractions*" case where the problem must be addressed from a transdisciplinary perspective.

3.2 Related Work

Previous work have addressed the issue of identifying relationships between educational and game dimensions. First, several works focused on identifying relations between educational and game elements. Prensky [16] is a pioneer who proposed relations between game genres (e.g., action, role-play, adventure), knowledge to be learned (e.g., facts, skills, judgement, behaviour) and learning activities (e.g., questions, experiments, observation). Rapeepisarn et al. [17], proposed an extension of [16] by adding a relation to Chong et al.'s learning styles [3] (i.e., activists, reflectors, theorists, and pragmatists). Likewise, Sherry [19] identified relations between games genres and the six levels of Bloom's taxonomy [1]. Gosper et McNeill [7] proposed a framework to support the integration of technology in education. Their framework defines relations between learning outcomes (e.g., acquisition of basic facts, automation of skills and concepts), learning processes (e.g., memorization, analogical reasoning), assessment (e.g., self-assessment, peer assessment) and game genres. These works are very interesting from a general design viewpoint of learning games. However, the identified relations are between high-level concepts, and cannot be used at a specification stage to guide the generation of activities.

Second, some works attempt to provide relations at a specification level. Dondi et Moretti [6] linked learning objectives (e.g., memorization/repetition/ retention), knowledge types (e.g., factual knowledge), and game genres to high-level features that games should possess (e.g., presence of content engine, assessment engine). However, these high-level features do not describe how the relations are to be implemented in practice.

In addition, other works propose a framework to specify relations (i.e., either for analysing existing games or conceiving one). The LM-GM framework [15] supports the transition from learning objectives/practices to game elements through a concept called Serious Game Mechanic (SGM). It defines learning mechanics and game mechanics and uses SGM to associate both concepts. However, the presented mechanics are high-level ones (e.g., guidance, collaboration, explore)

and the relations are not meant to be implemented as such. Furthermore, Hall et al. [8] proposed a framework to guide the designer in specifying the transition from learning content to core-gameplay. It is composed of 5 categories (i.e., goal, choice, action, rules, feedback) in which a series of questions need to be answered from a real-world and a game-world perspective. However, the framework is more oriented towards the general design of the game rather than its implementation.

To conclude, existing approaches are more oriented towards defining relations for analysis purposes or to assist in the high-level design of games rather than specifying relations for low-level design purposes. Moreover, these works address specific learning targets or the contexts of specific game genres, as we do.

3.3 Research Positioning & Objectives

Our work seeks to propose an approach to specify relations between declarative knowledge training tasks (i.e., independent of a specific didactic domain) and gameplays from the Rogue*lite* genre. These relations have to meet one condition: their specification must enable their implementation.

Training and the assessment of declarative knowledge are well known to be carried out through questionnaires and quizzes. Furthermore, digital quizzes, compared to paper ones, allow interactions with the user that are closer to those found in basic training games (e.g., a multiplication table training game where the correct answers allow the avatar to run faster or jump onto higher platforms). Accordingly, it appears interesting to use exercise types of quiz formats as a pivot in particular because using existing content may reduce subjectivity. Hence, our work intends to propose a systematic mapping approach based on the use of quizzes exercise types as a pivot, cf. Figure 3. The next sections present the development of our approach, followed by the proposed approach, a proposition for modelling relations, and an application example.

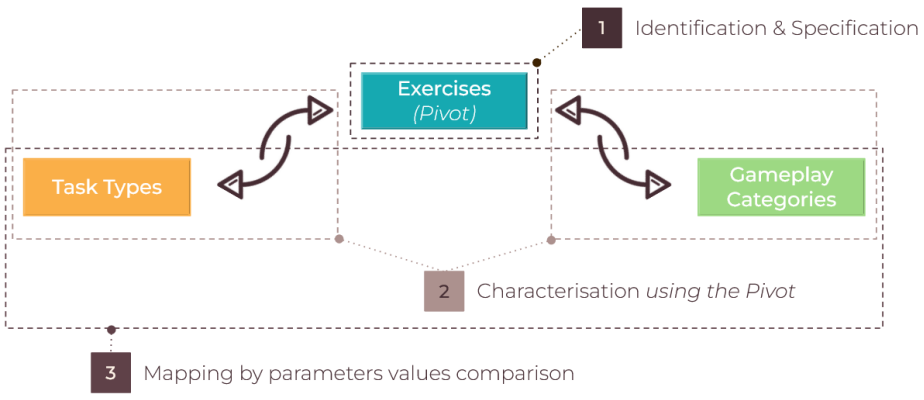


Fig. 3: Idea to map task types onto gameplay categories.

4 Mapping Approach Development

The elaboration of our approach required several stages. Initially, an analysis of quizzes design formats was carried out in order to define the types of existing exercises (i.e., our pivot). This work raised the following questions: (1) How can we draw a parallel between the types of tasks and the exercises identified? (2) How can we draw a parallel between the gameplay categories and the exercises identified? As previously mentioned, the interactions offered by each quiz exercise are closer to game interactions. In addition, each concept (i.e., task types, gameplay categories, and exercises) is characterised by its possible response modalities (e.g., enter an answer, choose between multiple propositions). Therefore, our second step consisted of using the exercise types to identify possible criteria and parameters to specify the task types and game categories and ease the identification of the mappings. Finally, these valued parameters (from both task types and gameplay categories perspectives) were used to compare and identify matches.

4.1 Identification of the *Pivot*

Foremost, exercises types of six tools, mostly extracted from Learning Management Systems (LMS), that allow the creation of numerical questionnaires/quizzes were analysed:

- ⋆ the eponymous and proprietary format from the *itsLearning* (#1) LMS;
- ⋆ *GIFT* (#2) a mark-up language for describing tests that is associated with the *Moodle* LMS;
- ⋆ *Performance Matters Assessment and Analytics* (#3) format associated with the *PowerSchool* LMS;
- ⋆ *NetQuizzPro* (#4) a software allowing the creation of questionnaires;
- ⋆ *QTI* (Question & Test Interoperability specification) (#5) from the IMS global learning consortium that defines a standard format to exchange and store assessment content;
- ⋆ *Tactileo – Maskott* (#6) format associated with the French pedagogical platform of the same name.

The analysis consisted of determining the different possibilities offered by these formats. More precisely, the various possible questions and their parameters were examined and compared. This led us to the definition of 12 different types of exercises useful for DK (i.e., only exercises for which the verification of results can be automated). Exercises (of different formats) which shared the same type of statement, the same number of desired answers and for which the interaction of the answers was similar were merged to form a single type of exercise. In addition, some formats combine several exercises into one (e.g., multiple choice and answers were merged into the *itsLearning* format). In those cases, the exercises were considered to be independent. Moreover, the possibility of having intruders (i.e., elements which should not be associated) has been requested by domain experts, but none of the "Associate" type exercises analysed from the

Table 1: Exercises by quiz format
(✓ present; ✗ absent; — present but incomplete) [14].

	itsLearning	*GIFT*	*PMAA*	*NetQuizzPro*	*QTI*	*Tactileo – Maskott*
Alternative	✓	—	—	✗	✗	✗
Multiple Choice	✓	✓	✓	✓	✓	✓
Multiple Resp.	✓	✓	✓	✓	✓	✓
Short Answer	✗	✓	✓	✗	✓	✓
Fill-in	✓	—	✓	✓	✗	✓
Fill-in Choice	✓	—	✗	✓	✓	✓
Reconstruction	✗	✗	✗	✓	✗	✗
Association	—	—	—	—	—	—
Order	✓	✗	✓	✓	✓	✓
G. Choice	—	✗	—	✗	✓	—
G. Identification	✗	✗	✗	✓	✗	✗
G. Association	✗	✗	✗	—	✓	✓

formats offers this possibility. Therefore, in our definition of exercises, it has been considered as a possibility. The types of exercise defined are as follows:

- ✏ *Alternative*: choosing one answer between 2 options;
- ✏ *Multiple choice*: choosing one answer between X (i.e., $X \geqslant 2$) options;
- ✏ *Multiple responses*: choosing Y (i.e., zero or more) answers between X (i.e., $X \geqslant 2$) options;
- ✏ *Short answer*: enter the correct answer. Multiple form of answers can be accepted, e.g., for example, *How much is 3 times 5?* as two possible answers, which are *15* and *fifteen*;
- ✏ *Fill-in-the-blanks*: enter for each gap of a text the wanted "short" answer;
- ✏ *Fill-in-the-blanks choices*: choose for each gap of a text the correct answer from a list. Each gap can have an associated list of options, or one list can be associated to all gaps;
- ✏ *Reconstruction*: reassemble each significant element of an information;
- ✏ *Associate–Group*: associate elements from a list or multiple lists together. The association can be done by pairs, or not. The elements can be associated with zero to several other ones;
- ✏ *Order*: replace a set of information in the correct order (i.e., following a heuristic);
- ✏ *Graphic choice*: point or locate X (i.e., $X \geqslant 1$) elements on a picture.
- ✏ *Graphic identification*: write the correct label for each area-to-complete of a picture;
- ✏ *Graphic association*: associate the correct labels to X areas of a picture.

As a reminder, these types of exercises aim to deal with declarative knowledge in general. As a result, some exercises offer a more visual approach that could be useful in the context of geographical facts for example.

These exercises are characterised by several parameters, cf. Table 2, such as: their *interactions*, their *response modality* (i.e., input or choice), their *statement*

Table 2: Characterisation of the exercises [14].

	Number of Facts	Statement Types	Response Modality	Number Answers	Number of Choices	Interactions
Alternative	1	Classic	Choice	1	2	*Select 1 from 2*
Multiple Choice	1	Classic	Choice	1	2 to ∞	*Select 1 from X*
Multiple Resp.	1	Classic	Choice	0 to ∞	2 to ∞	*Select Y from X*
Short Answer	1	Classic	Input	1	0	*Write 1*
Fill-in	1	Fill-in	Input	1 to ∞	0	*Write Y*
Fill-in Choice	1	Fill-in	Choice	1 to ∞	2 to ∞	*Y (Select 1 from X)* *Y (Select 1 from X_1 to X_Y)*
Reconstruction	1	Fill-in	Choice	2 to ∞	2 to ∞	*Match Y with X 1-to-1*
Association	2 to ∞	Classic	Choice	2 to ∞	4 to ∞	*Match Y with X 1-to-1* *Match Y with X*
Order	2 to ∞	Classic Fill-in	Choice	1 to ∞	2 to ∞	*Order Y*
G. Choice	1 to ∞	Graphic	Choice	1 to ∞	2 to ∞	*Point Y* or *Locate Y*
G. Identification	1 to ∞	Graphic	Input	1 to ∞	0	*Write Y*
G. Association	1 to ∞	Graphic	Choice	1 to ∞	1 to ∞	*Match Y with X 1-to-1*

type (i.e., format of the question asked), the *number of answers* desired, and the *number of propositions* presented (i.e., if the response modality of a concrete task of this type is "Choice"). Through our analysis, 6 types of interactions were identified:

- *Select Y From X* (i.e., the learner must select Y answers from a set of X values);
- *Y (Select 1 from X_1 to X_Y)* (i.e, the learner must make a selection of one answer from each set of proposals);
- a variant is *Y (Select 1 from X)* (i.e, the learner must select Y answers, one by one, from a set of proposals);
- *Write X* (i.e., the learner has to enter X answers);
- *Order X* (i.e., the learner must order X elements correctly);
- *Point X* or *Locate X* (i.e., the learner must point X elements on a picture or locate them);
- *Match Y with X 1-to-1* or *Match Y with X* (i.e., the learner must associate elements from Y with those from X by pairs or not).

In addition, 3 types of statement were found: 1) classic statement (i.e., a text question that can be accompanied by an image), 2) graphic statement (i.e., a classic statement accompanied by a graphic component with which interactions are required to answer), and 3) fill-in statement (i.e., a classic statement with incorporated fill-in areas). It is important to note that none of the formats allows for all possible forms of exercise.

4.2 Mapping Task Types onto Gameplay Categories

Having specified the pivot, the remaining questions are: How to map (1) task types onto exercises and (2) game categories onto exercises? Our main idea

involves using parameters that characterise each concept (i.e., task types, game categories and exercises) to map them.

Task types to exercises

Like exercises, task types are characterised by several parameters: the *number of facts* targeted by the task, the *types of statements* allowed for such a task, the *response modalities*, the *number of desired responses*, and the *number of propositions* presented (i.e., when the response modality for a concrete task of this type is "Choice"). For example, the **Identification** type is defined as follows: 1 to ∞ can be targeted, only classic statements are allowed, both response modalities can be used (i.e., input and choice), the number of desired answer is equal to the number of facts targeted, and at least 2 propositions must be presented when the modality is Choice.

It is important to note that the assignment of parameter values is not an easy task. Let's take a task T1 which consists of completing a fact having two missing elements (e.g., ? $\times$? = 12, i.e., *number of facts* = 1 and *number of expected answer* = 2). At first sight, it could seem possible to carry out T1 through the *Input* response modality. However, presenting a statement in the context of declarative knowledge, such as "? $\times$? = 12", does not give enough information about the fact to work with (i.e., is it 3×4 or 6×2). As another example, for a T1-like task, depending on how the choices are presented, it is possible to choose one or two answers. If the set of propositions represents numbers, such as [3, 5, 7, 4], two answers must be chosen. However, if each proposition is presented as a multiplication (without the result), such as [3×4; 4×5; 6×3], then only one answer is required. Table 3 presents our task types characterisation. Thus, except for the *interactions* parameter, task types and exercises are characterized by the same parameters.

Consequently, the mapping between task types and exercises consists of comparing the values of their common parameters. For example, **Identification** is mapped to *Short answer* because of the specification of *Short answer*, i.e., {number of facts = 1; type of statement = classic; modality = input; number of desired answers = 1}, is a possible configuration of a concrete task of the type **Identification** (i.e., the parameter values are included into those of the type **Identification**). This gives questions such as "Did World War II happened between 1914-1918?" and "Is 2 $\times$ 5 equal to 12?". **Completion** is mapped to *Fill-in-the-blanks choices* exercise specified as {number of facts = 1; type of statement = fill-in; modality = choice; number of desired answers = $[1 - \infty]$; number of choices = $[2 - \infty]$}. This gives questions such as "_ times 5 equals 15".

Thanks to these mappings, each task type can also be described by its possible interactions (i.e., the interactions of the exercises corresponding to the task type considering the number of requested answers). As an example, the *Short answer* exercise type asks the learner to write down one short answer to a textual question. Thus, its interaction parameter has for value *Write 1*. A task of the **Identification** type could ask questions such as "Did World War II happen

Table 3: Characterisation of the task types.

	Number of Facts	Statement Types	Response Modalities	Number of Answers	Number of Choices
Completion	1	Classic Graphic Fill-in	Input	1	0
			Choice		2 to ∞
			Choice	∞	2 to ∞
	2 to ∞		Input	*Nb Facts*	0
			Choice	$\geqslant$ *Nb Facts*	2 to ∞
Order	2 to ∞	Classic Fill-in	Choice	*Nb Facts*	*Nb Facts*
Identification	1 to ∞	Classic	Input	*Nb Facts*	0
			Choice		2 to ∞
Membership Identification	1 to ∞	Classic	Input	2 to ∞	0
		Graphic	Choice		2 to ∞

between 1914-1918?", leaving the learner to write down *True* or *T*. Therefore, **Identification** has *Write 1* for possible interaction. As a result of all mappings, **Membership Identification** can be achieved through the following interactions: Y *Write Y* (i.e., $Y \in [2, \infty]$), *Point Y*, *Select Y from X*, *Match Y with X 1-to-1*.

Gameplay categories to exercises

Each game category represents gameplays that are similar in terms of the actions to be performed, such as opening the right chest, choosing the right pot, passing through the right bridge which belong to the SELECT category. Therefore, the common parameters of these gameplays (e.g., number of facts interrogated, number of possible answers) represent those of the category itself.

After analysis, these categories were characterised using the following parameters: the *interactions*, the *response modality* (i.e., input or choice), the *statement type* (i.e., format of the question asked), the *number of facts* targeted, the *number of answers* desired, and the *number of propositions* presented (i.e., if the response modality of a concrete task of this type is "Choice"). These parameters are similar to those used for the exercises. They represent a minimal and relevant set of parameters that allow us to discriminate the different categories and gameplays. For example, the SELECT category is characterised as follows: 1 to many facts can be targeted, both classic and fill-in statement types are allowed, choice is the only possible response modality, 1 to many answers can be desired, and two interactions (i.e., *Select Y from X*, and *Y (Select 1 from* X_1 *to* X_Y*)*) are possible.

However, during the characterisation phase, it became apparent that the possible interactions and the statement type changed depending on whether one or more responses were desired. Therefore, to simplify the mappings, each category allowing one or more possible responses was divided into two sub-categories: single (i.e., only one possible response) and multiple (i.e., from two to several

possible responses). As a result, our 5 categories became 9. Table 4 presents the game categories through their characterisation. Afterwards, the mappings consisted of directly comparing the values of the parameters.

Table 4: Characterisation of the gameplay categories ((S) = Single, (M) = Multiple) [14].

	Number of Facts	Statement Types	Response Modality	Number Answers	Number of Choices	Interactions
Select (s)	1	Classic Graphic Fill-in	Choice	1	2 to ∞	*Select 1 from X* *Point 1* or *Locate 1*
Select (m)	1 to ∞	Classic Graphic Fill-in	Choice	2 to ∞	2 to ∞	*Select Y from X* *Point Y* or *Locate Y* *Y (Select 1 from X_1 to X_Y)*
Move (s)	1	Classic Graphic Fill-in	Choice	1	2 to ∞	*Select 1 from X* *Point 1* or *Locate 1* *Match 1 with 1*
Move (m)	1 to ∞	Classic Graphic Fill-in	Choice	2 to ∞	2 to ∞	*Match Y with X* *Point Y* or *Locate Y* *Select Y from X* *Y (Select 1 from X_1 to X_Y)* *Order X*
Orient (s)	1	Classic Fill-in	Choice	1	2 to ∞	*Select 1 from X*
Orient (m)	1 to ∞	Classic Fill-in	Choice	2 to ∞	2 to ∞	*Y (Select 1 from X_1 to X_Y)* *Y (Select 1 from X)*
Position (s)	1	Fill-in	Input	1	0	Write 1
		Classic Graphic	Choice		2 to ∞	*Select 1 from X* *Point 1* or *Locate 1*
Position (m)	1	Graphic	Input	2 to ∞	0	*Write Y*
Direct Resp.	1	Classic	Input	1	0	*Write 1*

Task types to gameplay categories

From there, all the necessary information was available to answer our main question: Which type of task is suitable for which category of game? What are the conditions? Consequently, the final step consisted of comparing the task types and categories on the basis of their parameter values (i.e., comparing Table 3 with Table 4). During this stage we observed that 4 parameters represented mapping conditions based on their values: statement type, number of facts targeted, number of expected answers, and the response modality. Therefore, the obtained relationships are 6-uplets composed as follows: (<task type>, [<statement type1>, <statement type2>, ...], <number of facts>, <number of expected answers>, <response modality>, [<category1>, <category2>, ...]).

In conclusion, this section presented the process followed to map the task types for declarative knowledge training to the gameplay categories for the Roguelite video game genre. Next section will present the results obtained.

5 A Systematic Mapping Approach

This research resulted in two contributions: (1) an approach for mapping designers' own task types to their own game categories, and (2) mappings between our task types and our gameplay categories.

5.1 Proposed Mapping Approach

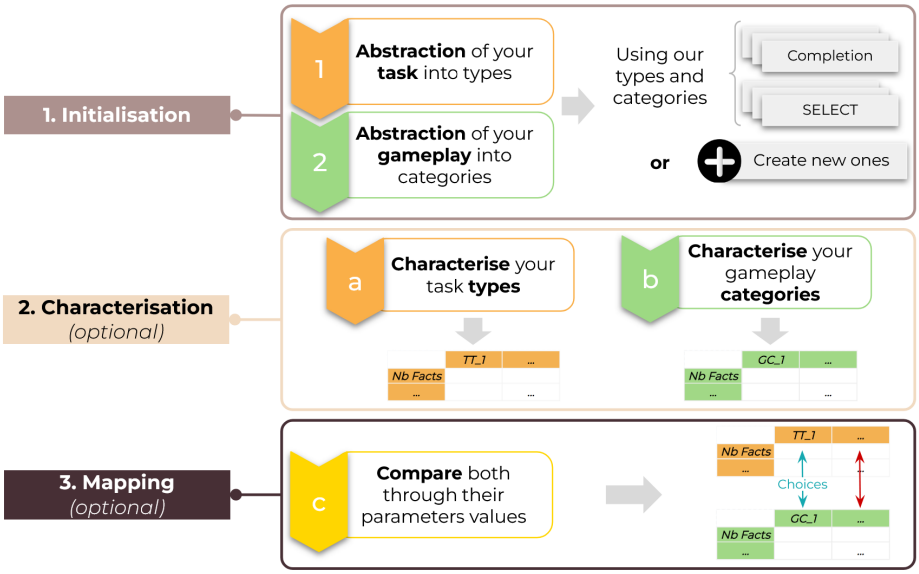


Fig. 4: Proposed Mapping Approach.

The proposed mapping approach is a two to five-steps approach, illustrated in Figure 4. It is composed of two initial steps:

1. abstraction of the concrete tasks using the types of tasks presented (e.g., a task "associate the right date with the historical event" becomes complete a fact with a missing element) or by creating new task types;
2. association of the gameplay to one of the categories presented or to a new gameplay category.

At this point, there are four possible states: new task types and categories have been created, only new task types have been created, only new game categories

have been created, or nothing has been created. According to the state, the instructions below must be followed:

1. If new task types and new gameplay categories were created:
   (a) Characterise the task types using the six parameters defined above (i.e., number of facts, types of statements, response modalities, number of desired responses, number of propositions, and interactions). In a substep, map task types and quiz exercises (cf. Table 2) to define the values of the *interactions* parameter.
   (b) Characterise the gameplay categories using the same parameters.
   (c) Finally, compare both tables (i.e., characterisation) through their values. As a reminder, the values of the *Statement Type*, *Number of Facts*, *Number of Answers*, and *Response Modality* parameters are possible conditions of the relations.
2. If only new task types were created, then realise step 1a and step 1c.
3. If only new gameplay categories were created, then realise steps 1b and 1c.
4. If no new elements have been created, the work is already done, cf. Figure 5.

Let's take as example a task type **T1** characterised as {number of facts = 1; type of statement = classic; modality = input or choice; number of desired answers = 1}, and a gameplay category **C1** = {number of facts = $[1 - \infty]$; type of statement = classic or fill-in; modality = choice; number of desired answers = $[1 - \infty]$}. In this case, only one relationship would result: *(**T1**, Classic, 1, 1, Choice, **C1**)*.

5.2 Relations Between Task Types and Gameplay Categories

The process presented in section 4 resulted in the definition of several conditional relationships between the task types and gameplay categories. Figure 5 presents the resulting relations. As examples, the task type **Order** has a unique relationship: *(**Order**, [Classic, Fill-in], [2 – ∞], Nb Facts, Choice, [*Move (m)*])*. The task type **Identification** has four relationships, including: *(**Identification**, Classic, 1, Nb Facts, Input, [*Position (s), Direct Response*])* and *(**Identification**, Classic, [2 – ∞], Nb Facts, Choice, [*Select (m), Move (m), Orient (m)*])*

5.3 Evaluation of the Relations

For the purpose of gathering feedback on the gameplay mock-ups (i.e., identifying relevant gameplays and game elements), the members (around 10 people) of our user group (from the AdapTABLES project) were asked to complete a survey presenting possible gameplays for each type of task. The experiment was also an opportunity to validate certain mappings (i.e., relationships for which the categories have existing gameplay mock-ups).

During the exploratory research [12], five domain-specific configurable tasks were identified for multiplication tables training:

1. *Completion 1* (of the type **Completion**): complete an incomplete multiplication fact that has one missing element (e.g., $3 \times ? = 15$, $15 = ? \times 5$, $3 \times 5 =$

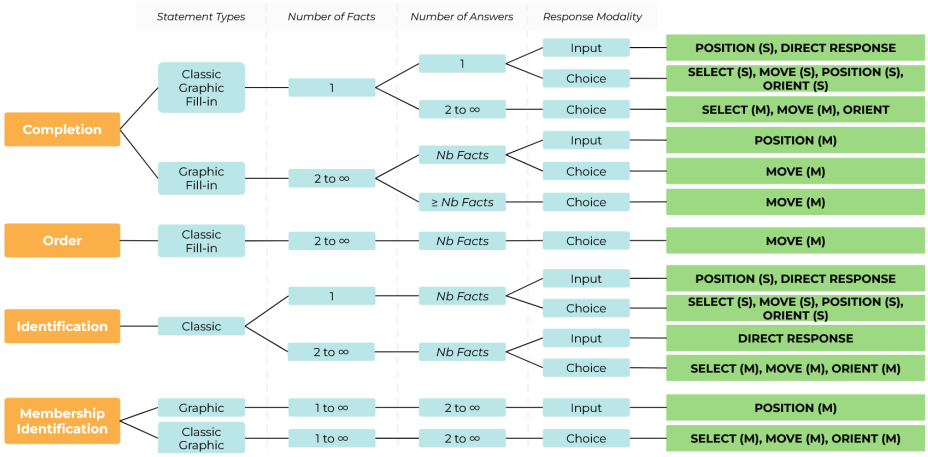


Fig. 5: Conditional relations between task types and gameplay categories.

?); It is characterised by: {number of facts = 1; type of statement = classic or fill-in; modality = input or choice; number of desired answers = 1}

2. *Completion 2* (of the type **Completion**): complete an incomplete multiplication fact that has two missing elements (e.g., ? × ? = 15 with a set of given choices [3, 6, 5, 10], ? × 5 = ? or 3 × ? = ? also with sets of given choices); It is characterised by: {number of facts = 1; type of statement = classic or fill-in; modality = choice; number of desired answers = 2}
3. *Reconstruction* (of the type **Completion**): reconstitute a multiplication fact (e.g., ? × ? = ? with a set of given choices [3, 6, 5, 10, 15]); It is characterised by: {number of facts = 1; type of statement = classic or fill-in; modality = choice; number of desired answers = 3}
4. *Identification* (of the type **Identification**): identify the accuracy or inaccuracy of one or several multiplication facts (e.g., $3 \times 5 = 15$, true or false?); It is characterised by: {number of facts = $[1 - \infty]$; type of statement = classic; modality = input or choice; number of desired answers = $[1 - \infty]$}
5. *(Non-)Membership Identification* (of the type **Membership Identification**): identify the elements that are results of a given multiplication table (e.g., [3, 5, 9, 12, 14, 21] which are results of the table 3?); It is characterised by: {number of facts = $[1 - \infty]$; type of statement = classic; modality = input or choice; number of desired answers = $[2 - \infty]$}.

The experiment was realised in the context of multiplication tables training. Therefore, none of the relations where the statement type was *Graphic* were evaluated. Furthermore, the tasks defined for the multiplication tables do not require completion targeting several facts or ordering. However, for every other relationship, gameplay mock-ups have been defined for each category and each task type. The survey consisted of presenting the experts with an image of a gameplay for a given task, a description of the gameplay, and a question on the relevance of the gameplay (a comment box was at their disposal). Let's take the

example of a gameplay consisting of selecting the right pot among several pots having propositions (i.e., SELECT with Choice) to answer a textual question of the type "$3 \times ? = 15$" (i.e., *Completion 1*). If this gameplay is validated by the survey, then so is the relationship *(**Completion**, Classic, 1, 1, Choice,* SELECT*)*.

According to the results, the **mappings were relevant**. Negative comments were about didactic issues or a lack of precision. As an example, MOVE gameplays that require objects to be placed on the correct answer were rejected because the selected answer was hidden by the object, thus impacting learners' thinking. This is a cognitive issue, unrelated to the game mechanic, that can be corrected by displaying the value above the object, or by displaying the chosen value inside the statement at the correct position with another colour. Figure 6 illustrates both solutions: the statue pushed on the left tile hides the associated '5' value, but 1) the value appears on top of the statue, 2) the value appears now in purple inside the room's statement.

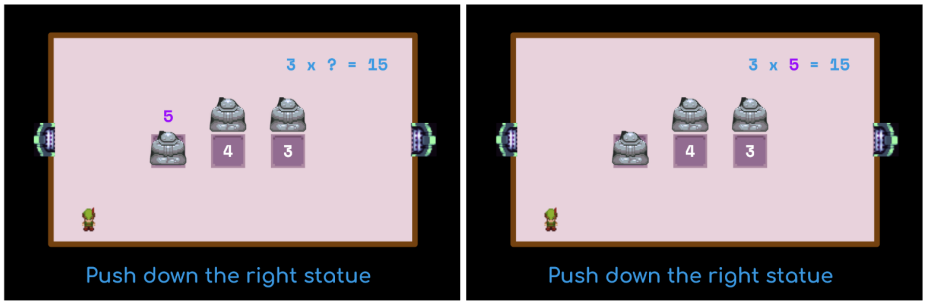


Fig. 6: Examples of possible solutions [14].

Furthermore, the gameplay mock-ups for the ORIENT category relied (at the time) on an object *lantern* where the avatar had to orient the light towards the answer. This gameplay received mixed reviews because of the lack of cognitive meaning of the object (i.e., light is emitted in every direction). In order to reach a set of satisfactory gameplays, we carried out a focus group in which we discussed the disagreements and proposed solutions to the problems observed (e.g., use of statues rather than lanterns).

6 A Formal Modelling of the Mappings

Overall, the aim of our research concerns the design of generators of Roguelite-oriented game activities for declarative knowledge training. Our approach's originality lies in the use of Model-Driven Engineering (MDE) [9] to depict and structure any data required by the generators.

Basically, the idea involves specifying the data (i.e., domain-dependent tasks, task types, game categories, gameplays, game elements, and relationships) in

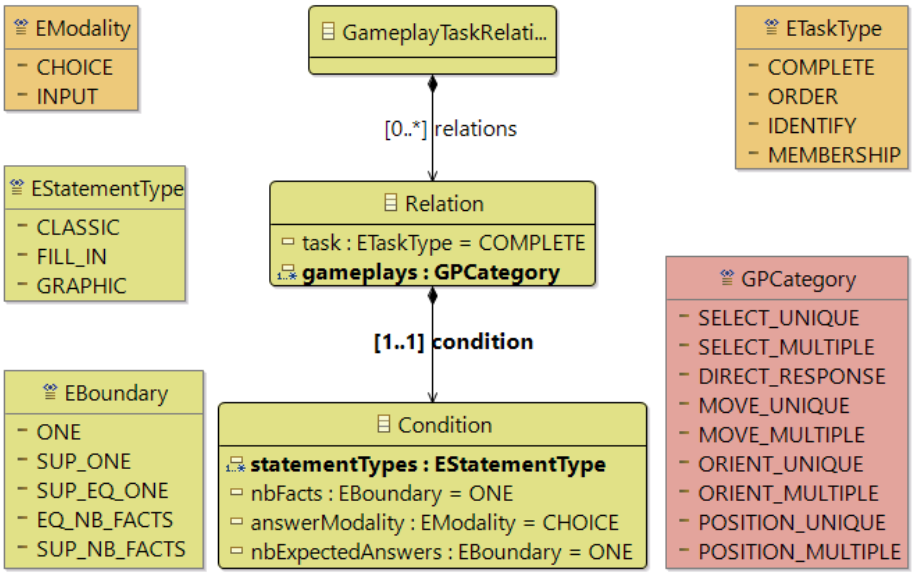


Fig. 7: Metamodel describing the conditional relation structure (unique=single).

different interconnected models that are consistent with a dedicated metamodel (i.e., model whose instances are models), that we also specify. Consequently, each activity generation could be seen as a model transformation according to MDE.

Currently, a generator has been developed for training multiplication tables. This generator uses models and metamodels that we specified using the *Eclipse Modeling Framework* (EMF) [20], and generates XMI files (i.e., EMF imposed format) accurately describing every element composing an activity (i.e., a dungeon, e.g., rooms, rooms order, game elements and their positions in each room, questioned facts). In order to have a universal, portable, and easily deployable structure, the XMI files are then translated into XML files using the *Epsilon Framework* [11]. Figure 7 presents the EMF metamodel used by the generator to depict the relations structure. As defined earlier, the relations are between a task type (i.e., attribute *task*) and a set of gameplay categories (i.e., attribute *gameplays*). These relations are bound by a condition featuring authorized statement types (i.e., attribute *statementTypes*), the number of facts targeted (i.e., attribute *nbFacts*), the number of expected answers (i.e., attribute *nbExpectedAnswers*) and the answer modality (i.e., attribute *answerModality*). *EBoundary* describes the possible values for the number of expected answers, so that the generation algorithm can interpret them. Domain-dependent tasks, as well as the specification of concrete gameplays and their implementation in game elements, are not shown in the figure. Another part of the metamodel (not displayed here) describes tasks by their attribute *type* (i.e., *ETaskType*) and gameplays by their attribute *category* (i.e., *GPCategory*).

Figure 8 presents the instantiation, through EMF, of the metamodel with the relations defined in Figure 5. At runtime, the generator loads and interprets this model to select the permitted gameplays based on the learner's context (i.e., the tasks the learner has to perform). For a given domain-dependent task, the main steps performed by the mapping algorithm are as follows:

1. get the associated task type of the targeted task;
2. collect every relation from the mapping model related to this task type;
3. restrict the collected relations to those whose associated condition is satisfied (compare the statement, number of facts, number of expected answers, and response modality values of the condition and the original task);
4. collect the gameplay categories of the remaining relations.

Therefore, by following our defined mappings (cf., Figure 5), multiplication tables training tasks (cf., Section 5.3) will be associated by the algorithm with:

- *Completion 1* is compatible with Position (s), Select (s), Move (s), Orient (s), and Direct Response;
- *Completion 2* and *Reconstruction* are compatible with Select (m), Move (m), Orient;
- *Identification* is compatible with Position (s), Select (s, m), Move (s, m), Orient (s, m), and Direct Response;
- *(Non-)Membership Identification* is compatible with Select (m), Move (m), Orient (m);

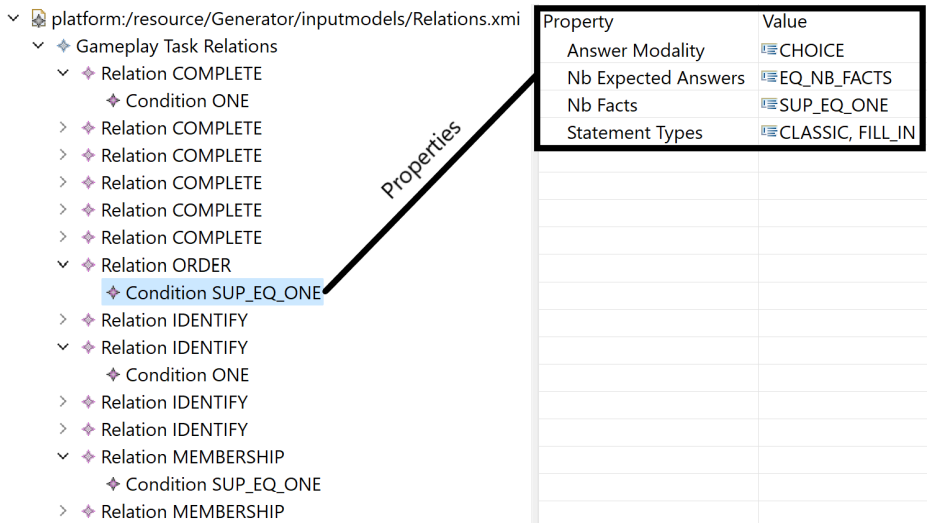
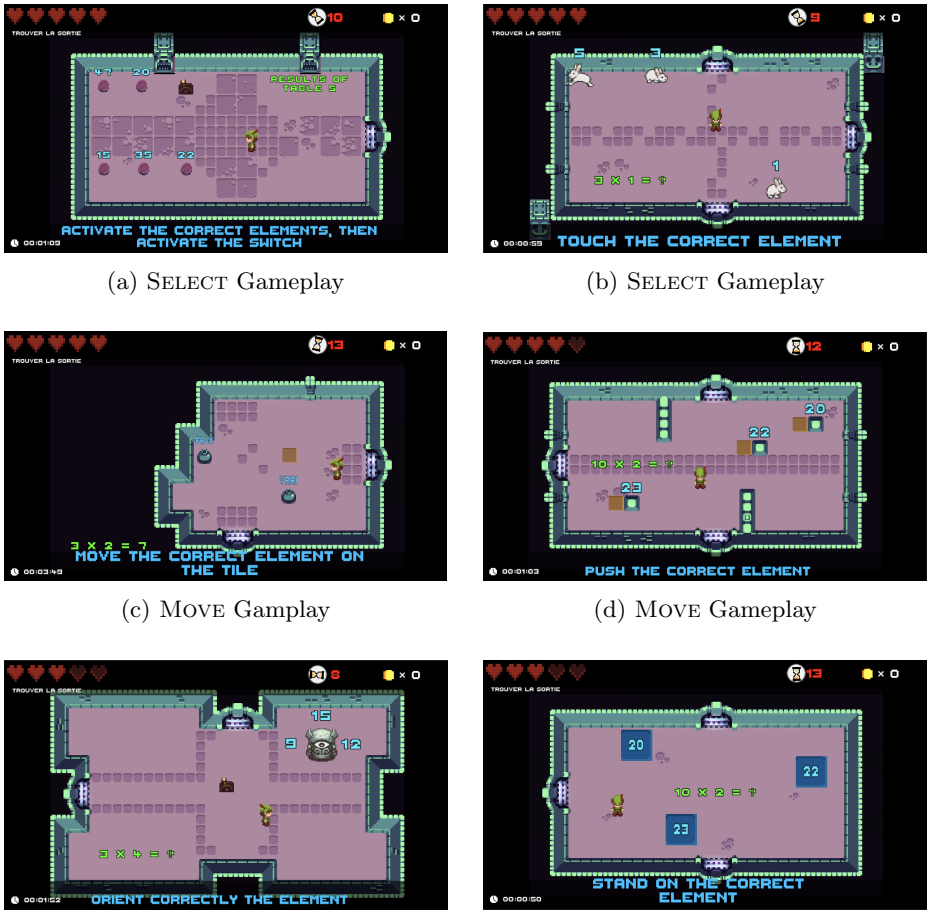


Fig. 8: Tree-based EMF model view of our model describing the defined relations.

Then, the generation algorithm handles further steps such as: selecting the gameplays that implement the allowed categories, filtering them according to other information (e.g., gameplays unlocked and available to the learner), instantiating the game elements composing the chosen gameplay, and so on.

Currently, a game prototype0 has been developed that uses this generator every time a new game level is requested. This prototype acts as an interpreter that reads a given XML file and transforms its contents into a playable level (i.e., a dungeon). Figure 9 presents screenshots of dungeon rooms and playable gameplays.



(a) Select Gameplay (b) Select Gameplay

(c) Move Gamplay (d) Move Gameplay

(e) Orient Gameplay (f) Position Gameplay

Fig. 9: Examples of playable gameplays.

7 Conclusion & Perspectives

In conclusion, this chapter has outlined a systematic approach for mapping task types onto gameplay categories in the context of declarative knowledge training. The originality of this work lies in two aspects: 1) the proposed approach is based on the use of a pivot (i.e., exercises extracted from numerical questionnaires tools); 2) it is oriented towards the automation of learning game activity design (i.e., generation) and therefore specifies fine-grained relationships. This chapter also presented a formal modelling of relationships to enable their interpretation by activity generators.

The task types and gameplay categories to be mapped were defined with the help of experts (i.e., teachers, didactic experts, game designers) and do not claim to be exhaustive. Gameplay categories are defined to offer a wide variety of activities, while task types are designed to meet the needs expressed by our experts. As a result, these task types could be refined according to other didactic domains, and the game categories proposed could be extended. Furthermore, the parameters used in the approach are subjective in the sense that they represent those that are necessary in our opinion. As a result, these parameters could be argued according to different viewpoints. In addition, not every relationship was evaluated, but only those used for multiplication table training.

Furthermore, a generator for multiplication was developed that uses the presented metamodel and model to generate activities. In addition, this generator is currently being used in a game prototype that interprets the generated XML files (i.e., detailed description activities) to provide learners-players with playable dungeons. In future work, we plan to implement a history-geography fact generator and evaluate relationships that have not yet been assessed.

References

1. Bloom, B.S.: Taxonomy of educational objectives: The classification of educational goals. Cognitive domain (1956)
2. Brame, C.J., Biel, R.: Test-Enhanced Learning: The Potential for Testing to Promote Greater Learning in Undergraduate Science Courses. CBE—Life Sciences Education **14**(2) (Jun 2015). https://doi.org/10.1187/cbe.14-11-0208
3. Chong, Y., Wong, M., Thomson Fredrik, E.: The impact of learning styles on the effectiveness of digital games in education. In: Proceedings of the Symposium on Information Technology in Education, KDU College, Patailing Java, Malaysia (2005)
4. Codish, D., Ravid, G.: Detecting playfulness in educational gamification through behavior patterns. IBM Journal of Research and Development **59**(6), 1–14 (Nov 2015). https://doi.org/10.1147/JRD.2015.2459651
5. Djaouti, D., Alvarez, J., Jessel, J.P., Methel, G., Molinier, P.: A Gameplay Definition through Videogame Classification. International Journal of Computer Games Technology pp. 1–7 (2008). https://doi.org/10.1155/2008/470350
6. Dondi, C., Moretti, M.: A methodological proposal for learning games selection and quality assessment. British Journal of Educational Technology **38**(3), 502–512 (May 2007). https://doi.org/10.1111/j.1467-8535.2007.00713.x

7. Gosper, M., McNeill, M.: Implementing game-based learning: The MAPLET framework as a guide to learner-centred design and assessment. In: Assessment in Game-Based Learning, pp. 217–233. Springer, Springer Nature, United States (2012). https://doi.org/10.1007/978-1-4614-3546-4_12
8. Hall, J.V., Wyeth, P.A., Johnson, D.: Instructional objectives to core-gameplay: A serious game design technique. In: Proceedings of the First ACM SIGCHI Annual Symposium on Computer-human Interaction in Play. pp. 121–130. ACM, Toronto Ontario Canada (Oct 2014). https://doi.org/10.1145/2658537.2658696
9. Kent, S.: Model Driven Engineering. In: Integrated Formal Methods. pp. 286–298. Springer Berlin Heidelberg (2002)
10. 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). https://doi.org/10.1080/1464536X.2011.573008
11. Kolovos, D., Rose, L., Paige, R., Garcia-Dominguez, A.: The Epsilon Book. Eclipse (2010)
12. 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). https://doi.org/10.5220/0011033100003182
13. Lemoine, B., Laforcade, P., George, S.: An analysis framework for designing declarative knowledge training games using roguelite genre. In: Proceedings of the 15th International Conference on Computer Supported Education, CSEDU 2023, Volume 2, Prague, Czech Republic, April 21-23, 2023. pp. 276–287. SCITEPRESS (2023). https://doi.org/10.5220/0011840200003470
14. Lemoine, B., Laforcade, P., George, S.: Mapping task types and gameplay categories in the context of declarative knowledge training. In: Proceedings of the 15th International Conference on Computer Supported Education, CSEDU 2023, Volume 2, Prague, Czech Republic, April 21-23, 2023. pp. 264–275. SCITEPRESS (2023). https://doi.org/10.5220/0011840100003470
15. Lim, T., Carvalho, M.B., Bellotti, F., Arnab, S., de Freitas, S., Louchart, S., Suttie, N., Berta, R., Gloria, A.D.: The LM-GM framework for Serious Games Analysis. Pittsburgh: University of Pittsburgh (2013)
16. Prensky, M.: Computer Games and Learning: Digital Game-Based Learning. Handbook of Computer Game Studies (2005)
17. Rapeepisarn, K., Wong, K.W., Fung, C.C., Khine, M.S.: The Relationship between Game Genres, Learning Techniques and Learning Styles in Educational Computer Games. In: Technologies for E-Learning and Digital Entertainment, vol. 5093, pp. 497–508. Springer Berlin Heidelberg (2008)
18. Roediger, H.L., Pyc, M.A.: Inexpensive techniques to improve education: Applying cognitive psychology to enhance educational practice. Journal of Applied Research in Memory and Cognition **1**(4), 242–248 (Dec 2012). https://doi.org/10.1016/j.jarmac.2012.09.002
19. Sherry, J.L.: Matching computer game genres to educational outcomes. In: Teaching and Learning with Technology, pp. 234–246. Routledge (2010)
20. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework. Addison-Wesley Professional, Boston, Massachusetts (2008)
21. 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). https://doi.org/10.1007/978-1-4020-9827-7_16