# Framework de conception et d'implémentation de générateurs d'activités de jeu d'entrainement aux connaissances déclaratives

## *Framework for the design and implementation of game activity generators for declarative knowledge training*

Bérénice LEMOINE[1]; Pierre LAFORCADE[1]; Sébastien GEORGE[1]

[1]Laboratoire d'Informatique de l'Université du Mans, Laval, France

**Résumé.** La génération d'activités adaptées est une technique peu abordée en Environnements Informatiques pour l'Apprentissage Humain. Cet article aborde l'aide à la conception de générateurs d'activités adaptées dans les jeux sérieux pour l'entrainement aux connaissances déclaratives. La proposition est un *framework* de conception de générateurs d'activités (i.e., architecture logicielle), orienté modèles et méta-modèles, extensible à différents domaines didactiques. Les générateurs produits permettent d'obtenir des activités d'entrainement adaptées et variées sous forme de niveaux de donjon pour des jeux de type *Roguelite*. L'article présente ce *framework* ainsi que son extension à deux domaines didactiques (i.e., table de multiplication et faits de judo) et son évaluation du point de vue ingénierie (i.e., tests automatisés, validation de modèles).
**Mots-clés :** génération, adaptation, jeux sérieux, conception, modélisation

***Abstract.*** *Activity generation is a relatively underexplored technique in Technology Enhanced Learning. This article targets the design assistance of game activity generators for declarative knowledge training. The proposal is a model-driven framework for activity generator design (i.e., software architecture) that can be extended to various didactic domains. The produced generators provide varied and adapted training activities in the form of dungeon levels for Roguelite-oriented games. The article presents this framework, its extension to two didactic domains, and its engineering evaluation, encompassing automated testing and model validation.*
***Keywords:*** *generation, adaptation, serious games, design, modelling*

## 1. INTRODUCTION

La mémorisation à court et long termes de connaissances déclaratives (e.g., lois, faits, règles) nécessite de la répétition (Kim *et al.*, 2013). Cependant, la répétition peut rapidement devenir ennuyeuse pour les apprenants (Smith, 1981). Toutefois, les jeux sérieux offrant des activités redondantes avec un challenge non adapté aux compétences/connaissances des apprenants-joueurs peuvent également conduire à un sentiment d'ennui (Streicher et Smeddinck, 2016) pouvant pousser à l'abandon des tâches, impactant alors l'apprentissage. En conséquence, pour limiter le sentiment d'ennui, les jeux sérieux visant l'entrainement de connaissances déclaratives doivent proposer des activités 1) variées et 2) adaptées aux apprenants-joueurs. La personnalisation manuelle d'activités pour chaque apprenant est une tâche chronophage et exigeante en termes d'efforts. De plus, concevoir des activités de jeu nécessite des compétences en *game design* (e.g., conception de situations de jeu) que les enseignants ne possèdent pas forcément.

D'autre part, la littérature en psychologie cognitive a montré que le processus de récupération de concepts ou de faits par le biais de tests augmente leur acquisition à long terme (Brame et Biel, 2015). Le concept de *Retrieval Practice* est une forme d'apprentissage par le test, consistant en des rappels répétés de ce qui a été appris (e.g., par l'utilisation de flashcards, de quiz) (Roediger et Pyc, 2012). Dans notre contexte, nous définissons l'entrainement comme une forme de *Retrieval Practice* consistant à poser, de manière répétée, différentes formes de questions sur des faits aux apprenants-joueurs.

La génération est une technique informatique (voir Figure 1) permettant la création automatique de contenu (e.g., niveau de jeu, histoire, dialogue) à partir d'un ensemble de données structurées et d'un ensemble de règles définies au travers d'algorithmes. Par conséquent, la génération d'un contenu décrivant une activité d'entrainement formalisée, à destination d'un interpréteur inclus dans le jeu sérieux d'entrainement, est une solution envisageable pour proposer des activités variées et adaptées. Cependant, le principe de génération n'est que très peu abordée dans le domaine des Environnements Informatiques pour l'Apprentissage Humain (EIAH) (Bezza *et al.*, 2013).

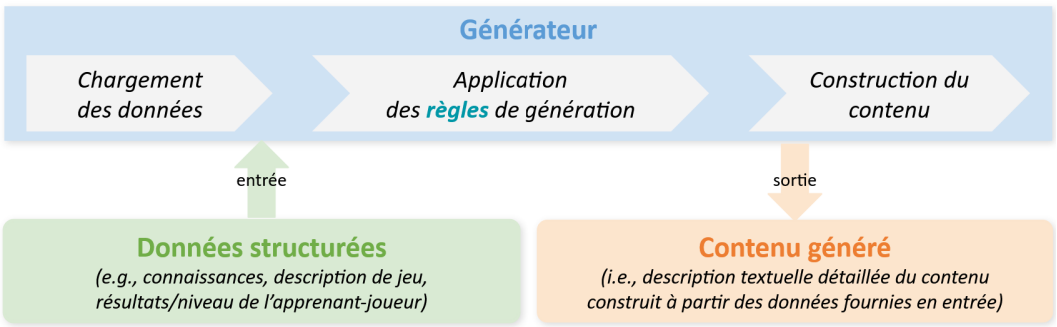


Figure 1 : Principe de génération de contenu

La structure d'une activité de jeu est entièrement dépendante du genre de jeu visé. Par exemple, une activité consistant à résoudre des énigmes et une activité consistant à explorer un monde ne se construisent pas de la même façon. Une activité d'énigme va être construite à partir d'un problème à résoudre (e.g., *Professeur Layton*) alors qu'une activité consistant à explorer va nécessiter la création d'un monde (zone d'exploration virtuelle) et d'un avatar (personnage) qui se déplacera et effectuera des actions. En conséquence, pour générer des activités de jeu d'entrainement, un genre de jeu doit être sélectionné. Les *Roguelites* sont souvent des jeux de type exploration de donjons dans lesquels les joueurs doivent parcourir des

niveaux générés, combattre des ennemis, collecter des objets et progresser. Ce genre de jeu a été analysé et il semble posséder les différentes caractéristiques nécessaires à de l'entrainement aux connaissances déclaratives : variété (génération procédurale avec de l'aléatoire), répétition (mécanique de mort permanente) et progression (rétention d'items) (Lemoine *et al*., 2024b).

Nos travaux portent sur la recherche en ingénierie pour la conception et le développement de générateurs. Ces générateurs doivent produire des activités de jeu de type *Roguelite*, variées et adaptées pour l'entrainement aux connaissances déclaratives. En considérant les jeux d'entrainement comme des logiciels, les générateurs d'activités sont des composants logiciels du point de vue informatique. Nos travaux consistent en une recherche exploratoire visant à mieux caractériser ces générateurs (objets de recherche) et à proposer des modèles, des outils et des techniques pour faciliter leur conception (au sens informatique) et leur développement. Concevoir de tels générateurs est complexe et ne peut pas être réduit à un pur problème d'ingénierie informatique (Tchounikine *et al*., 2009). En effet, de nombreux problèmes de spécifications et de mise en œuvre requièrent l'expertise de différents acteurs : les développeurs du jeu d'entrainement (choix de conception, exploitation des technologies), les experts du domaine didactique visé (faits à travailler, manière de les travailler, choix des adaptations par apprenant) et des experts en jeux vidéo (connaissances en jeu, *game design*...).

Dans ce but, nous proposons un *framework* (i.e., une infrastructure logicielle composée de modèles et d'outils) permettant d'assister la conception et l'implémentation de ces générateurs. Le travail présenté dans Lemoine *et al*. (2023) avait pour objectif de présenter le framework en se focalisant sur la dimension éducative uniquement. Cet article étend ce travail en y ajoutant les éléments en lien avec la dimension ludique. La proposition s'inscrit dans le cadre d'une approche d'Ingénierie Dirigée par les Modèles (IDM) (Kent, 2002). Le *framework* a permis de produire un générateur d'activités d'entrainement aux tables de multiplication qui est actuellement utilisé dans un prototype de jeu.

La Section 2 présente les travaux existants. La Section 3 définit notre contexte de recherche et positionne nos travaux. La Section 4 présente le *framework* dans son ensemble. La Section 5 présente une preuve de concept de l'extensibilité du *framework* au travers de son application à deux domaines didactiques. La Section 6 présente l'évaluation du *framework* en termes d'infrastructure logicielle. Enfin, la Section 7 conclut et présente les perspectives.

## 2. ÉTAT DE L'ART

La génération d'activités adaptées pour des jeux d'entrainement nécessite de s'intéresser à trois axes de recherche : la conception de jeux (plus particulièrement, la conception d'activités de jeu), l'adaptation de contenu, et la génération de contenu. Il est important de souligner que **la conception de générateurs d'activités de jeu ne nécessite pas de concevoir des jeux** sérieux. Les générateurs sont des composants logiciels indépendants, qui peuvent donc être évalués indépendamment d'un jeu. Cependant, créer des activités de jeux nécessite d'effectuer en amont des choix en termes de conception de jeu.

### 2.1. CONCEPTION DE JEUX ET DE JEUX SÉRIEUX

La conception de jeux et de jeux sérieux a fait l'objet de nombreuses méthodes, *frameworks* et approches (Amory, 2007 ; Carvalho *et al*., 2015). Parmi ces travaux, on peut citer le *framework* MDA *(Mechanics, Dynamics, Aesthetics)* (Hunicke *et al*., 2004) qui propose de décomposer les jeux en trois parties : la mécanique correspondant aux principaux com-

posants du jeu, la dynamique correspondant au comportement du jeu en cours d'exécution et l'esthétique correspondant aux réactions émotionnelles souhaitables pour le joueur. Ou encore, GOM *(Game Object Model)* version 1 et 2 (Amory, 2007), un *framework* orienté objets qui considère que le jeu sérieux est composé d'un ensemble de composants appelés *objets* qui sont décrits par des interfaces abstraites et concrètes. Les éléments éducatifs sont associés aux interfaces abstraites et les éléments de jeux sont associés aux interfaces concrètes. Cependant, la majorité des travaux existants sont orientés soit vers l'analyse de jeux existants, soit vers l'assistance à la conception générale (conception de haut niveau et spécification des besoins) du jeu à développer (Junior et Silva, 2021). À notre connaissance, aucun de ces travaux de conception de jeux et de jeux sérieux n'aborde la notion d'adaptation aux apprenants-joueurs, ni la notion de génération. Pourtant, de nombreux jeux commerciaux sont fondés sur un principe de génération (e.g., *Hadès, The Binding of Isaac, Enter the Gungeon*).

### 2.2. Adaptation et méthodes d'adaptation en EIAH

L'adaptation peut être mise en œuvre de différentes manières et peut viser une ou plusieurs cibles (e.g., les préférences de jeu, le contenu d'apprentissage, la difficulté). L'adaptation est souvent caractérisée par trois concepts : la source (à quoi adaptons-nous ?), la cible (qu'est-ce qui est adapté ?) et les « chemins » (comment est adaptée la cible à la source ?) (Vandewaetere *et al*., 2011). Les formes d'adaptation qui nous intéressent principalement sont l'individualisation et la personnalisation de contenu. L'*individualisation* peut être définie comme un moyen de répondre aux compétences ou aux capacités spécifiques des élèves, y compris les besoins spéciaux, en fournissant des progressions d'apprentissage adaptées à ses besoins (Grant et Basye, 2014). D'autre part, la *personnalisation* peut être définie comme l'utilisation de modèles dans le but d'adapter les systèmes à chaque personne (Bakkes *et al*., 2012 ; Ismail et Belkhouche, 2018) [1]. De nombreux travaux proposent d'adapter la ludification de plateformes ou contenus éducatifs (Codish et Ravid, 2015 ; Monterrat *et al*., 2017). Cependant, la ludification de contenus et la conception de jeux sérieux présentent différents objectifs. Ludifier consiste à ajouter une couche d'éléments de jeux à du contenu éducatif déjà existant et structuré. En revanche, concevoir un jeu sérieux requiert de penser conjointement la structuration du contenu éducatif et la structuration du jeu (Prensky, 2005).

Plusieurs travaux abordent la conception ou l'aide à la conception de contenus de jeux adaptés. Natkin *et al*. (2007) proposent un système de recommandation de quêtes en fonction d'un modèle utilisateur. Un ensemble de quêtes, sélectionné à partir du modèle utilisateur, est proposé au joueur. En fonction des choix du joueur et de ses traces d'utilisation, le modèle est mis-à-jour pour permettre de raffiner les propositions de quêtes par la suite. Marne *et al*. (2013) présentent un outil auteur permettant de soutenir les enseignants dans la conception de scénarios de jeux adaptés et non linéaires. En fonction de leurs réponses, les apprenants-joueurs sont confrontés à différents scénarios. En plus des activités d'apprentissage, les scénarios incluent des activités purement ludiques. Marty et Carron (2011) présentent une approche permettant de créer des modèles usager adaptés pour les jeux sérieux. Bontchev *et al*. (2021) proposent un framework incluant un modèle d'apprenant-joueur pour la personnalisation de jeux sérieux de type labyrinthe. Ce modèle est découpé en trois axes : les caractéristiques du joueur, les caractéristiques de l'apprenant et les caractéristiques de l'utilisateur.

Ces travaux ont tous un point commun : ils utilisent des modèles et des résultats des apprenants pour adapter. Cependant, les formes d'adaptation visées sont toutes différentes :

1. Ces deux termes seront traités comme des synonymes dans le cadre de cet article.

système de recommandation fondé sur un modèle utilisateur (Natkin *et al*., 2007), système de sélection d'activités à partir des réponses utilisateurs et du scénario créé par l'enseignant (Marne *et al*., 2013), modèles incluant des paramètres permettant l'adaptation aux apprenants (Bontchev *et al*., 2021 ; Marty & Carron, 2011). Les approches étant différentes, les modèles sont dépendants des contextes et donc difficilement applicables à d'autres contextes. Cependant, ces approches peuvent aider ou orienter la définition d'autres modèles similaires dans d'autres contextes.

Quelques travaux s'intéressent à assister la conception de systèmes d'apprentissage adaptés. Par exemple, Roepke *et al*. (2021) proposent une architecture modulaire, basée sur des composants, pour mettre en œuvre des pipelines personnalisés pour les jeux d'apprentissage dans le cadre de la formation à la lutte contre l'hameçonnage. Un pipeline est un processus en trois étapes : collecte de données, génération de contenu, diffusion de contenu. Ismail et Belkhouche (2018) proposent une architecture réutilisable pour la conception de systèmes logiciels d'apprentissage personnalisé, décomposée en quatre unités : l'unité apprenant (qui conserve les données relatives à l'apprenant), l'unité des connaissances (qui conservent les ressources d'apprentissage), l'unité de personnalisation (qui fait correspondre le modèle de l'apprenant aux ressources d'apprentissage) et l'unité de présentation (qui représente l'environnement du logiciel). Ces travaux proposent des lignes directrices suffisamment générales pour être suivies pour la conception de tout système d'apprentissage adapté.

### 2.3. Génération en EIAH et dans les jeux

Bien que peu traitée en EIAH, la problématique de la génération, consistant à construire automatiquement du contenu, a été abordée sous trois angles principaux : génération de contenu non adapté, génération de contenu adapté à l'apprenant et génération de contenu adapté au joueur. Diwan *et al*. (2019) proposent un modèle pour générer des parcours pédagogiques à partir de ressources open-source. Holohan *et al*. (2006) ont défini une ontologie permettant de décrire les bases de données relationnelles. Cette ontologie est utilisée pour générer automatiquement des exercices en ligne pour l'apprentissage de connaissances procédurales (i.e., les bases de données relationnelles). Carpentier et Lourdeaux (2014) proposent une approche pour générer dynamiquement des scénarios adaptés aux capacités des apprenants et aux besoins pédagogiques dans des environnements virtuels. Leur approche s'inscrit dans un cadre fondé sur trois modèles : le modèle du domaine (la description statique du monde, de ses éléments et de leurs relations), le modèle de l'activité (la structure hiérarchique de l'activité observée), le modèle de causalité (l'expression des chaînes causales pertinentes se produisant dans l'environnement). Sehaba et Hussaan (2013) proposent une architecture générale pour la génération de scénarios de jeu adaptés à l'apprenant (c'est-à-dire à ses compétences, aptitudes et besoins). Cette architecture repose sur plusieurs modèles : le modèle du domaine qui modélise les concepts du domaine et leurs relations ; le modèle de l'apprenant qui modélise les informations personnelles, la motivation, les compétences et les interactions des apprenants ; le modèle de présentation qui décrit les structures des scénarios ; le modèle du jeu sérieux qui associe les ressources du jeu à celles de la pédagogie. Les connaissances en matière d'adaptation sont représentées sous la forme d'un système basé sur des règles. Laforcade et Laghouaouta (2018) proposent une approche IDM, inspirée des travaux de Sehaba et Hussaan (2013), pour permettre la spécification des générateurs de séquences d'activités (scénarios de jeu) adaptées aux besoins individuels de l'apprenant. L'approche s'appuie sur trois perspectives (points de vue incrémentaux sur les éléments à générer) et trois dimensions (les éléments à générer, les éléments décrivant le contexte de la génération, les éléments décrivant le jeu d'apprentissage). Callies *et al*. (2015) proposent une architecture adaptative consistant à générer des plans pédagogiques adaptés ainsi qu'à

adapter les comportements des personnages non-joueurs en fonction des actions des joueurs dans des jeux de type simulation. Ce travail est basé sur un modèle de joueur composé de ses connaissances du jeu et du domaine, ainsi que sur un module d'adaptation.

Dans le contexte des jeux, certains travaux abordent la génération de contenus adaptés, comme Dormans et Bakkes (2011) qui proposent un *framework* pour la conception de niveaux de jeu adaptés, de type action-aventure, fondée sur l'utilisation de *grammaires génératives*. Des niveaux à choix conditionnels sont créés dynamiquement pour chaque type de joueurs (modèle de joueur) afin de personnaliser l'expérience de jeu. Sina *et al*. (2014) proposent *ScenarioGen* une méthode pour générer du contenu textuel sur les activités quotidiennes. Leur méthode consiste à concevoir un nouveau scénario à partir d'un profil utilisateur et d'un scénario existant. Ces travaux ont tous un point commun : ils utilisent des données structurées (e.g., modèles, ontologies) à partir desquelles du contenu est généré.

### 2.4. CONSTATS

Le point essentiel de ces recherches réside dans le fait que l'adaptation et la génération de contenu sont majoritairement fondées sur l'utilisation de données structurées telles que des modèles, des ontologies, etc. En ce qui concerne l'adaptation du contenu éducatif, avec ou sans génération, les principales données structurées utilisées sont :

— des informations sur le domaine qui décrivent les connaissances et le contenu d'apprentissage ;
— des informations sur l'apprenant comprenant ses données personnelles, sa progression, ses résultats, etc. ;
— des informations sur la structure des activités à générer ; et des informations sur les règles d'adaptation.

D'autre part, pour ce qui est de l'adaptation du contenu des jeux, avec ou sans génération, les principales données structurées utilisées sont :

— des informations sur le jeu, c'est-à-dire la description statique des éléments du jeu ;
— des informations sur la structure de l'activité de jeu à générer ;
— et des informations sur le joueur incluant ses préférences, sa progression de jeu, etc.

Lorsque ces travaux se penchent sur la génération de contenu de jeu adapté, ils introduisent généralement des informations supplémentaires concernant les relations entre les éléments de jeu et les éléments éducatifs. Cependant, ces relations sont souvent intégrées au sein de l'algorithme de génération sans être spécifiées à travers des modèles de manière explicite. En outre, les modèles ou méthodes de représentation des données proposés sont généralement spécifiques à un contexte ou à un domaine didactique particulier, ce qui les rend difficilement réutilisables pour d'autres domaines.

Un second constat important est que l'adaptation est souvent abordée soit du point de vue éducatif, soit du point de vue ludique, mais rarement dans les deux perspectives simultanément. L'ajustement des activités en fonction des apprenants est reconnu comme un moyen d'améliorer l'apprentissage. De plus, l'adaptation aux joueurs contribue à rendre les tâches plus engageantes et motivantes. Il semble donc pertinent d'envisager l'adaptation en prenant en considération à la fois la dimension éducative et ludique.

Par conséquent, la création d'activités de jeu pour l'entrainement nécessite la création de modèles pour spécifier l'ensemble des données nécessaires comme le domaine didactique visé, la structure de l'activité, les résultats et la progression des apprenants-joueurs ou encore les éléments de jeu. Afin de favoriser un apprentissage plus efficace et un engagement accru des apprenants-joueurs, la génération doit offrir des activités adaptées sur les dimensions éducative et ludique.

# 3. Contexte de Recherche

## 3.1. Projet AdapTABLES

AdapTABLES est un projet de recherche qui s'intéresse à l'acquisition longue durée des tables de multiplication. Ce projet vise à concevoir et développer un jeu sérieux dédié à l'entrainement aux tables de multiplication. L'objectif principal du projet est de stabiliser les connaissances (Dias, 2018), c'est-à-dire que les tables sont considérées comme déjà expérimentées et comprises.

De nombreux jeux sérieux proposent de travailler les tables de multiplication en ligne. Cependant, ces jeux présentent en général des questions auxquelles les joueurs doivent répondre, accompagnées de mécaniques de jeu (e.g., récompenses, scores, pression du temps). Les choix en termes de pédagogie se limitent souvent à la sélection des tables à travailler ou au niveau de difficulté (facile, moyen, difficile). Ces niveaux de difficultés ont majoritairement un impact sur le temps de réponse autorisé ou sur les tables travaillées. Cependant, la méthode d'attribution des niveaux de difficulté aux tables n'est pas explicitée. Dans le cadre du projet, une étude exploratoire (Laforcade *et al.*, 2022) a été conduite à l'aide d'experts en mathématiques (i.e., enseignants des cycles 2-3 et didacticien). Le principal objectif de cette étude était de permettre la spécification des besoins d'adaptation du point de vue des tables de multiplications (organisation d'un entrainement, tâches d'entrainement, éléments à faire varier, etc.).

## 3.2. Problématique et Positionnement

À partir des constats relevés précédemment, notre problématique est la suivante : **comment faciliter la conception de générateurs d'activités de jeu adaptées et variées, destinées à l'entrainement aux connaissances déclaratives ?** Plusieurs questions de recherche découlent de cette question générale :

1. Comment proposer une approche suffisamment générique pour considérer les connaissances déclaratives indépendamment d'un domaine didactique spécifique ?
2. Qu'est-ce qu'une activité d'entrainement de jeu adaptée et variée ? De quels éléments éducatifs et de quels éléments de jeu sont composées ces activités ? Comment associer les éléments de jeu et les éléments éducatifs de manière cohérente ?
3. Comment structurer ces éléments et leurs relations pour guider la génération d'activités cohérentes ?
4. Comment spécifier ces informations informatiquement pour développer des générateurs d'activités ?

L'intérêt de s'intéresser aux connaissances déclaratives réside dans la possibilité de proposer des modèles pouvant être réutilisés au-delà d'un domaine didactique spécifique. Comme précédemment mentionné, l'acquisition de connaissances déclaratives implique une pratique répétée (Kim *et al.*, 2013). Dans notre contexte, l'entrainement consiste en la répétition de diverses formes de questions sur des faits, qui sont posées aux apprenants-joueurs de manière répétée. Or, la structure d'une activité de jeu étant dépendante du genre de jeu visé, un genre doit être sélectionné pour permettre la génération d'activités de jeu d'entrainement. En conséquence, nous avons étudié différents genres de jeux afin d'identifier ceux capables de maintenir l'engagement des joueurs tout en proposant des *gameplays* répétitifs, mais variés (i.e., conditions nécessaires à l'entrainement). Le *Roguelite* répond à ces besoins (Lemoine *et al.*, 2024b). Ce genre se caractérise principalement par la génération procédurale de donjons au contenu pseudo-aléatoire, la mort permanente (chaque mort de l'avatar impose au joueur

de commencer une nouvelle partie), et la détention limitée d'éléments de jeu déblocables (e.g., personnages, objets, *powerups*, ...) facilitant la progression ludique dans la prochaine partie. Par conséquent, une activité de jeu d'entrainement est un *donjon*, c'est-à-dire un ensemble de salles interconnectées dans lesquelles l'avatar se déplace et où l'entrainement a lieu.

Pour réduire la sensation de répétition causée par l'entrainement, les activités générés doivent être variées et adaptées. Étant donné que l'adaptation aux deux dimensions (éducative et ludique) est rarement abordée conjointement, notre objectif est d'individualiser les activités pour les joueurs et les apprenants. Pour être plus précis, l'**adaptation** vise à prendre en considération **trois perspectives** distinctes : celle de l'**enseignant** (stratégies d'entrainement, choix pédagogiques et didactiques), celle de l'**apprenant** (son niveau de connaissance et sa progression), et celle du **joueur** (ses préférences de jeu). L'adaptation envisagée, selon Plass et Pawar (2020), porte sur deux variables cognitives (le niveau ou la progression de l'apprenant et la stratégie de l'enseignant) ainsi qu'une variable motivationnelle (l'intérêt du joueur à travers ses préférences). Ainsi, l'adaptation éducative consistera à prendre en compte divers aspects tels que les types de faits rencontrés, la manière de les questionner, leur nombre, leur ordre en fonction du niveau de l'apprenant, de ses résultats antérieurs et de l'entrainement établi par l'enseignant. D'autre part, les *Roguelites* possèdent souvent une mécanique d'achat ou d'activation permettant aux joueurs d'effectuer des choix pouvant ou non influencer la génération des niveaux de jeu. Ainsi, l'adaptation des préférences consistera à se focaliser sur la possibilité d'activer ou de désactiver des éléments de jeu, notamment des équipements débloquant différents *gameplays* qui permettront de répondre aux faits questionnés, en orientant ou en déplaçant des objets, par exemple. Ce choix de conception permet aux joueurs de désactiver des *gameplays* qui leur déplaisent.

En plus de l'adaptation, la variété des activités est nécessaire pour réduire le sentiment d'ennui produit par la répétition. Les *Roguelites* sont fondés sur un mécanisme de génération procédurale avec de l'aléatoire rendant chaque niveau de jeu différent en termes de contenus (structure des donjons, éléments et leur position dans les salles, etc.). Dans notre contexte, notre approche consiste à modéliser des éléments (éducatifs et de jeux) avec une certaine variabilité. Par exemple, les faits sont modélisés pour que les mauvais choix soient définis à chaque génération d'un donjon. Ou encore, les *gameplays* sont définis à l'aide de **capacités** pour permettre de faire varier les éléments de jeux sélectionnés à chaque génération d'un donjon. L'objectif est de permettre à l'algorithme de génération de choisir de façon pseudo-aléatoire, pour conserver la cohérence des activités, les éléments d'entrainement et de jeux d'un donjon.

Un générateur d'activité de jeu de type *Roguelite* pour l'entrainement aux connaissances déclaratives est un composant logiciel (élément constitutif d'un logiciel destiné à être incorporé en tant que pièce détachée) dont l'algorithme permet de construire des activités variées à partir de trois types d'informations fournies en entrée : des informations sur l'entrainement, des informations sur le jeu et des informations sur l'apprenant-joueur concerné. Ces composants logiciels produisent en sortie des descriptions détaillées d'activités (niveaux de donjon) adaptées aux apprenants-joueurs.

Pour aborder notre problématique, nous proposons un *framework* de conception et d'implémentation de générateurs d'activités adaptées et variées pour l'entrainement aux connaissances déclaratives à travers des jeux de type *Roguelite*. L'originalité de cette proposition et de ce positionnement est d'aborder l'adaptation en prenant en compte simultanément les dimensions de jeu et d'entrainement (apprentissage). De plus, le framework est un outil (infrastructure logicielle) disposant d'un mécanisme d'extension permettant de prendre en compte de nombreux domaines didactiques. À notre connaissance, aucune approche permet-

tant de guider la conception de générateurs au niveau algorithmique n'existe. Ce *framework* est composé d'un **cadre conceptuel** (Lemoine & Laforcade, 2023a) et d'une **infrastructure logicielle extensible** (partiellement présentée (Lemoine *et al.*, 2023)) à des domaines didactiques spécifiques. Cette infrastructure logicielle capture l'ensemble des éléments communs pour tout domaine didactique. Tandis que le mécanisme d'extension guide l'ajout des éléments spécifiques à un domaine didactique visé. L'avantage d'une approche extensible est de faciliter l'implémentation de générateurs d'activités, par des ingénieurs ou développeurs, en limitant le développement nécessaire aux informations reliées aux connaissances déclaratives du domaine didactique. Ce framework propose un algorithme, des modèles, et méta-modèles déjà existants à étendre et qui seront donc réutilisés pour spécifier un générateur d'activités.

Notre proposition s'inscrit dans le contexte d'une approche fondée sur l'Ingénierie Dirigée par les Modèles (Kent, 2002). L'IDM est fondée sur la notion de modèle (abstraction d'un système selon un point de vue) et repose sur quatre principes : la capitalisation (les modèles doivent être réutilisables), l'abstraction (les modèles doivent être indépendants des technologies), la modélisation (les modèles doivent adopter une vision productive, c'est-à-dire permettre la génération de code final du logiciel), et la séparation des préoccupations (Jézéquel *et al.*, 2012). L'IDM est un champ de recherche vaste abordant la spécification, l'exécution, la transformation et la composition de modèles. La transformation de modèles est une opération centrale en IDM qui permet la génération automatique de modèles à partir de modèles sources. Pour permettre ces transformations, les modèles doivent être conformes à des méta-modèles (modèle permettant de décrire des modèles) qui définissent la structure et les règles qu'ils doivent respecter. Les principes de l'IDM, ainsi que les différents outils permettant de soutenir son utilisation, en font une approche très intéressante dans notre contexte. La contribution présentée s'inscrit dans le cadre de la recherche en ingénierie des systèmes EIAH (Tchounikine *et al.*, 2009) contribuant à l'exploration et à l'orientation des solutions pour la génération d'activités adaptées.

## 4. Framework de conception et d'implémentation de générateurs

Cette section présente la contribution : un framework de conception et d'implémentation de générateurs d'activités de jeu d'entrainement aux connaissances déclaratives. La sous-section 4.1 présente une vue d'ensemble de la proposition. La sous-section 4.2 présente le cadre conceptuel du *framework* défini au cours de cette recherche exploratoire. Enfin, la sous-section 4.3 présente l'implémentation informatique du *framework* (modèles interprétables, algorithme de génération, règles d'extension).

### 4.1. Présentation générale et Propriétés

Le *framework* proposé est une infrastructure conceptuelle et logicielle composée d'un ensemble de modèles et d'outils pour formaliser et pour guider l'implémentation de générateurs d'activités variées et adaptées. Les générateurs produits sont des éléments logiciels pouvant être considérés comme des composants d'un jeu d'entrainement aux connaissances déclaratives de type *Roguelite*. Ces générateurs permettent de produire une nouvelle activité d'entrainement à chaque demande, c'est-à-dire une description textuelle détaillée d'un niveau de donjon, pour un apprenant-joueur donné. Les descriptions doivent ensuite être interprétées par un *game player* pour proposer un niveau de jeu jouable à l'apprenant-joueur.

Le *framework* est fondé sur une approche conceptuelle structurant l'ensemble des informations nécessaires à la génération (voir Section 4.2) et implémenté au travers d'une approche d'Ingénierie Dirigée par les Modèles (voir Section 4.3). Ce *framework* se décompose en deux parties : un ensemble de composants génériques et des règles permettant l'extension à des domaines didactiques spécifiques. Étant dans le cadre d'une approche IDM, les composants génériques regroupent un ensemble de modèles et de méta-modèles ainsi qu'un algorithme de génération d'activités adaptées et variées utilisant ces modèles. Comme nous l'expliquerons ultérieurement, certains éléments ne peuvent pas être traités ou générés indépendamment du domaine. Les règles d'extension permettent de construire les modèles, les méta-modèles, et les composants de code spécifiques au domaine et requis par le générateur. La Figure 2 présente les différents composants du *framework*.

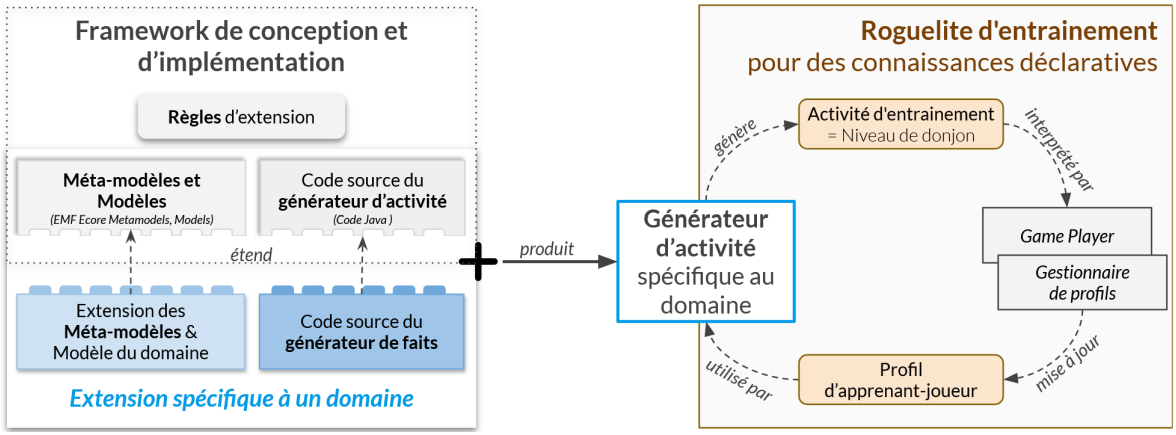


Figure 2 : Positionnement général du *framework* de conception

Dans un contexte IDM, l'algorithme de génération peut être perçu comme réalisant une transformation de modèle. Ainsi, les générateurs produits requièrent en entrée, un modèle conforme à chaque méta-modèle d'entrée du *framework* (données concrètes : connaissances à travailler, éléments de jeu disponible, résultat et progression de l'apprenant-joueur, etc.) et produisent en sortie un modèle également conforme à un méta-modèle du *framework*. Ces générateurs sont composés du code source correspondant aux méta-modèles du *framework*, leur permettant de lire et d'utiliser les données contenues dans les modèles. Ils possèdent également l'algorithme de génération d'activité générique (indépendant du domaine didactique) ainsi que les modèles et l'algorithme de génération de faits conçu lors de l'extension du *framework* en fonction du domaine didactique. La Figure 3 représente l'ensemble des composants d'un générateur dans notre contexte.

Comme évoqué dans la Section 3.2, les activités générées doivent être variées et adaptées en fonction de trois points de vue distincts : celui de l'enseignant, en tenant compte de sa vision de l'entraînement pour chaque apprenant, celui de l'apprenant, en prenant en considération son niveau et sa progression dans l'entraînement, et enfin, celui du joueur, en intégrant ses préférences de jeu. De plus, le *framework* doit viser l'entrainement aux connaissances déclaratives non spécifiques à un domaine didactique. En conséquence, le *framework* et les générateurs produits par ce dernier doivent respecter certaines propriétés. Le *framework* doit posséder les deux propriétés suivantes :

— **FP1** : possibilité d'exprimer différents domaines didactiques ;
— **FP2** : possibilité d'exprimer la vision des enseignants sur l'entrainement des apprenants individuellement.

En ce qui concerne les générateurs, il est nécessaire de garantir le respect de trois propriétés :

— **GP1** : les activités générées doivent être adaptées au niveau et résultats de l'apprenant dans son parcours d'entrainement ;
— **GP2** : les activités générées doivent être adaptées aux préférences de jeu du joueur ;
— **GP3** : les activités générées doivent être variées sur le plan éducatif et de jeu.

Ce *framework* a été élaboré dans le cadre d'une recherche exploratoire centrée sur la conception itérative d'un cas d'étude initial, avec l'implication d'un groupe d'usagers et la participation d'experts. Il s'agit donc d'une méthode inductive où les résultats obtenus, en rapport avec le projet AdapTABLES, sont généralisés et re-évalués dans le contexte des tables de multiplication, mais également sur deux autres domaines, c'est-à-dire les techniques et gestes d'arbitrages de judo et les repères d'histoire-géographie du Diplôme National du Brevet.

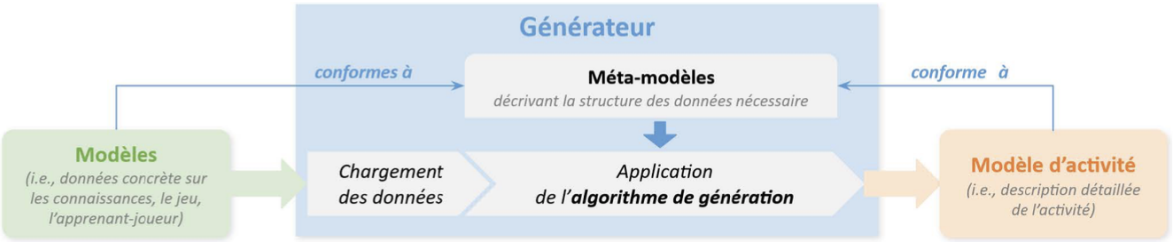


Figure 3 : Composants à gros grain d'un générateur d'activité d'entrainement de jeu

### 4.2. Cadre conceptuel : des modèles pour la génération

Comme précédemment mentionné, la génération requiert différentes informations structurées pour générer des activités variées et adaptées. Nous avons proposé un ensemble de modèles conceptuels nécessaires à la génération d'activités de jeu de type *Roguelite* pour l'entrainement aux connaissances déclaratives (Lemoine et Laforcade, 2023a ; 2023b). Ces modèles ont été définis pour être génériques (toutes connaissances déclaratives indépendamment d'un domaine spécifique). En conséquence, certains modèles contiennent des points d'extension, c'est-à-dire une portion de modèle à étendre en fonction du domaine visé. Cette section présente ces différents modèles conceptuels.

#### 4.2.1. Modèle du domaine : parcours d'entrainement et connaissances

Dans le contexte du projet, une étude exploratoire a été menée avec des experts en mathématiques, partiellement présentée (Laforcade *et al.*, 2022). Cette étude avait deux principaux objectifs : 1) définir une structure de parcours d'entrainement, 2) spécifier les adaptations à prendre en compte pour l'entrainement aux tables de multiplication. À partir des résultats obtenus avec les experts en mathématiques, un travail d'abstraction nous a permis de définir une structure d'entrainement (voir Figure 4), indépendante des mathématiques, appelée parcours d'entrainement (Lemoine *et al.*, 2024b). Cette structuration vise à **expliciter la vision de l'enseignant sur la progression de l'entrainement** (adaptation), en termes de connaissances à travailler et de paramètres à faire varier, **pour un apprenant ou un groupe d'apprenants**. Un parcours d'entrainement consiste en un ensemble d'objectifs (e.g., s'entrainer sur la table de 2) ordonnés par des relations de pré-requis. Chaque objectif vise un ensemble de faits à travailler (connaissances visées) et est décomposé en niveaux progressifs, eux-mêmes décomposés en tâches d'entrainement.

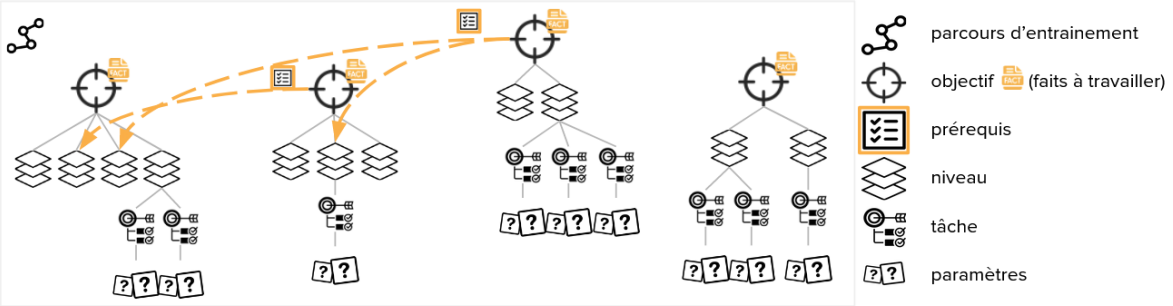


Figure 4 : Illustration de la structure des parcours d'entrainement

Dans le cadre des tâches pour l'entrainement aux tables de multiplication, cinq tâches ont été identifiées :

— Complétion 1 : compléter un fait incomplet ayant un élément manquant (e.g., $3 \times ? = 15$, $15 = ? \times 5$) ;
— Complétion 2 : compléter un fait incomplet ayant deux éléments manquants (e.g., $3 \times ? = ?$ avec un ensemble de choix donnés [3, 6, 5, 15]) ;
— Reconstruction : replacer, dans l'ordre correct, tous les éléments importants d'un fait (e.g., $? \times ? = ?$ avec un ensemble de choix donnés [3, 6, 5, 10, 15]) ;
— Identification : identifier l'exactitude ou l'inexactitude d'un ou de plusieurs faits (e.g., $3 \times 5 = 15$, vrai ou faux ?) ;
— Identification d'Appartenance : identifier les éléments qui partagent ou non une propriété donnée (e.g., [3, 5, 9, 14, 21] qui sont des résultats de la table 3 ?).

Notons que les trois premières tâches (Complétion 1, Complétion 2, Reconstruction) sont trois formes de complétion de faits ayant des éléments manquants.

D'autre part, nous avons échangé avec des enseignants d'histoire-géographie [2] pour 1) discuter de la structure d'entrainement et 2) définir des tâches d'entrainement pour les repères d'histoire-géographie du brevet. Les enseignants avec lesquels nous avons discuté ont considéré la structure d'entrainement appropriée. Les tâches d'entrainement actuellement définies sont semblables à celles des mathématiques : association (une forme de complétion de plusieurs faits), vérification de la validité d'un fait (une forme de tâche d'identification), nommer et localiser sur une carte (une forme d'identification d'appartenance). Cependant, une tâche est différente, la tâche d'ordonnancement chronologique.

En fonction du domaine didactique, une tâche possède différents paramètres permettant de construire les faits à questionner. Pour illustrer, dans une tâche de complétion en mathématiques, l'élément manquant sera un entier (opérande, table ou résultat), tandis qu'en histoire, il s'agira plutôt d'une chaîne de caractères (événement, date ou période). Ainsi, pour permettre l'obtention d'un algorithme de génération d'activités générique et comme trois des tâches sont présentes en mathématiques et en histoire-géographie, nous avons défini **quatre types de tâches** génériques :

— **Complétion** (compléter des faits ayant des éléments manquants) ;
— **Identification** (identifier si des faits sont vrais ou faux) ;
— **Identification d'Appartenance** (identifier si des faits partagent ou non une propriété commune) ;
— **Ordonnancement** (ordonner des faits en fonction d'une heuristique donnée).

Ces tâches n'ont pas pour but d'être exhaustives, mais de couvrir au moins les domaines

2. Ce second domaine est présenté ici seulement pour illustrer d'autres faits et tâches spécifiques ; dans la section dédiée à l'évaluation nous nous appuierons sur un troisième domaine.

d'application visés.

Ainsi, un premier modèle à fournir au générateur est le **modèle du domaine** (*LearningDomain*) qui comprend le parcours d'entrainement (*LearningPath*) et les connaissances à travailler. La Figure 5 présente une version simplifiée du modèle du domaine. Les faits à travailler sont regroupés en ensemble de faits (e.g., tables de multiplication) et peuvent être graphiques, c'est-à-dire avoir une représentation graphique, être associé à une image ou une carte (e.g., régions de France). Ces faits (*Fact*) sont des faits bruts tels que $3 \times 5 = 15$, puisqu'ils sont différents en fonction des domaines didactiques, la notion de fait est abstraite. De même, les niveaux (*Level*) sont également abstraits puisqu'ils peuvent posséder des paramètres spécifiques au domaine didactique (e.g., construction des tables de multiplication, $1 \times 1, 1 \times 2...$ ou $1 \times 1, 2 \times 1...$). Enfin, les tâches (*Task*) sont abstraites puisqu'elles possèdent des paramètres spécifiques au domaine didactique tel que le type d'élément recherché dans une tâche de complétion.

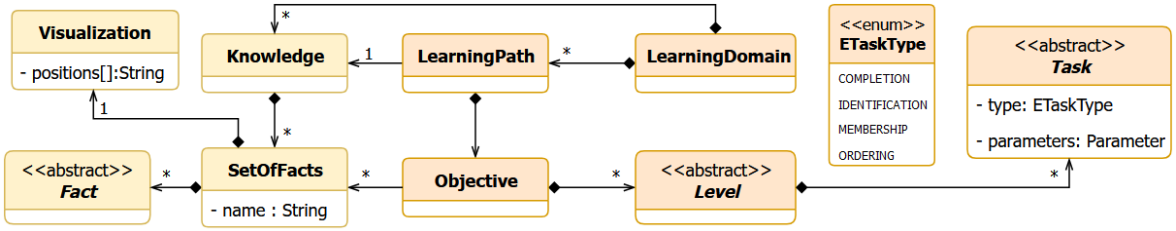


Figure 5 : Modèle conceptuel simplifié du modèle de domaine (Lemoine & Laforcade, 2023a)

Si certains paramètres des tâches sont spécifiques au domaine, d'autres sont communs à toutes les tâches ou à un type de tâche donné. Toutes les tâches décrivent le nombre de faits à questionner en même temps, le taux d'apparition de la tâche (favorisation d'une tâche plutôt qu'une autre par l'enseignant), le temps de réponse maximal attendu par question, le nombre de réponses attendues par fait, le critère d'acquisition des faits (combien de fois un fait doit être réussi consécutivement pour être considéré comme acquis ?). De plus, les tâches décrivent si ce sont des tâches graphiques (avec carte), si elles sont à validation automatique ou à validation manuelle de la réponse par l'apprenant (permet une correction et une autoréflexion sur la réponse donnée, mais fait perdre du temps dans le cadre d'un jeu). Pour les tâches d'Identification d'Appartenance, un paramètre commun est un booléen indiquant si les éléments recherchés sont ceux qui partagent la propriété ou ceux qui ne la partagent pas.

### 4.2.2. Modèle du jeu

D'après Prensky (2005), la principale raison de l'échec des jeux d'apprentissage réside dans leur « manque » de *gameplay* (gameplays trop simples ou minimalistes). Dans un jeu, le *gameplay* est représenté par l'ensemble des éléments avec lesquels les joueurs interagissent (actions que l'avatar peut effectuer et qui ont un impact sur la progression) ou qui leur fournissent des informations. En conséquence, nous avons tenté de proposer une variété de *gameplays*.

Ainsi, en parallèle des échanges avec des experts des domaines, nous avons échangé avec des concepteurs de jeux vidéos pour concevoir des *gameplays* permettant de répondre à des questions sur des connaissances déclaratives (en s'inspirant des tâches définies dans les deux domaines). À partir des échanges, plusieurs maquettes de *gameplays* ont été définies. Suite à la conception des maquettes, nous avons constaté que certains *gameplays* semblaient appartenir à une même catégorie. Par exemple, casser un pot possédant la bonne réponse ou

ouvrir un coffre possédant la bonne réponse sont deux manières de sélectionner une réponse. Ce constat est cohérent avec la classification de jeux de Djaouti *et al*. (2008) qui consiste à décrire les jeux à l'aide de briques de *gameplay* (des catégories d'actions pouvant être réalisées dans les jeux). En conséquence, nous avons défini cinq catégories de *gameplay* dans le contexte des *Roguelites* d'entrainement aux connaissances déclaratives (ces catégories ne prétendent pas être exhaustives) : *SELECT* (sélectionner les objets avec les bonnes réponses), *MOVE* (déplacer les bons objets vers les zones attendues), *ORIENT* (orienter les objets vers les bonnes réponses), *POSITION* (placer l'avatar sur la bonne réponse) et *DIRECT RESPONSE* (saisir la bonne réponse) (Lemoine *et al*., 2024a).

Ces catégories représentent différentes manières de répondre aux questions posées sur les faits. D'autres *gameplays* présentant des éléments présents dans les *Roguelites* comme les pièges à éviter, des ennemis, ou des objets à casser pour obtenir des pièces doivent également être présents pour conserver l'esprit du *Roguelite*. Cependant, si la définition statique des gameplays en termes d'éléments de jeu spécifiques permet un certain niveau de variété, elle impose deux contraintes : 1) elle est coûteuse en temps, c'est-à-dire que les gameplays doivent être décrits un par un en fonction des éléments de jeu disponibles et 2) elle est statique, c'est-à-dire que l'ajout d'un élément de jeu signifie qu'il faut spécifier de nouveaux *gameplays* pour cet élément. Ainsi, pour augmenter la variabilité des *gameplays* proposés, nous proposons de définir les *gameplays* et les éléments de jeux au travers de **capacités**. Les capacités définissent le comportement des éléments (la manière dont l'avatar du joueur peut interagir avec eux). Par exemple, un bloc peut être poussé (*pushable*), un pot peut être déplacé (*movable*), un pont peut être traversé (*crossable*), etc. Ainsi, plusieurs types d'éléments peuvent être définis avec une même capacité, par exemple, un cube et un pot peuvent être déplacés (*movable*). Cette modélisation permet de définir un *gameplay* d'éléments déplaçables mais de générer des *gameplays* jouables (description réelle du *gameplay* qui va être joué) avec différents types d'éléments. Ainsi, les *gameplays* sont décrits par des composants (*AComponent*), pouvant être simples (e.g., pot à casser) ou composites (e.g., bloc à pousser avec son détecteur associé), faisant référence à une capacité attendue. Ces composants décrivent également le type d'information attendu : proposition(s), question, propositions et question (e.g., structure à déclarer pour chaque question sur un fait), etc. Ces informations permettront de correctement instancier un *gameplay* (création des objets de jeu) en fonction des faits qui sont questionnés.

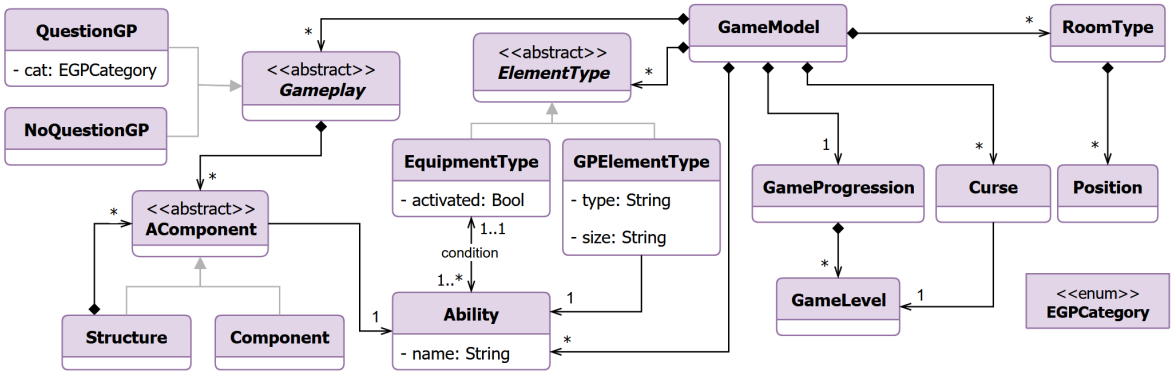


Figure 6 : Modèle conceptuel simplifié du jeu (Lemoine & Laforcade, 2023a, 2023b)

Comme précédemment précisé, les *Roguelites* sont fondés sur une mécanique d'achat ou d'activation. Dans notre contexte, l'achat d'équipement permet de déverrouiller des capacités et donc de débloquer de nouveaux *gameplays*. Ces éléments doivent donc être modélisés

pour permettre à l'algorithme de génération de ne choisir que des *gameplays* déverrouillés. D'autre part, dans le but de faire varier la forme des salles des donjons, nous avons fait le choix de modéliser des types de salles (*RoomType*) qui décrivent les positions possibles pour des éléments et les accès qu'ils autorisent. Enfin, un jeu possède une progression qui doit être spécifiée également. Comme pour la plupart des *Roguelites*, la progression est fondée sur une approche en termes de niveaux de difficultés. Un niveau de jeu influe sur la taille du donjon (le nombre de salles avec et sans question) et les pièges possibles (difficulté et nombre). De plus, de nombreux *Roguelite* propose une mécanique de malédiction qui apparaissent lorsque le joueur a atteint un certain niveau de difficulté. Par exemple, à partir du niveau 8 le niveau est dans le noir. Nous avons également fait le choix de conserver cette mécanique de malédiction. Cependant, certaines malédictions pourraient poser un problème du point de vue de l'apprentissage. Par exemple, une malédiction imposant de terminer un niveau en un temps donné pourrait aller à l'encontre de l'objectif pédagogique de l'enseignant. L'ajout de malédictions doit donc faire l'objet de réflexions avec les enseignants.

Ainsi, un second modèle à fournir au générateur est le modèle du jeu qui décrit l'ensemble des éléments de jeu disponibles et la progression à suivre du jeu. La Figure 6 présente une version simplifiée du modèle conceptuel du jeu.

### 4.2.3. Modèle de l'activité

Une activité d'entrainement aux connaissances déclaratives pour des *Roguelites* est un donjon, c'est-à-dire un ensemble de salles interconnectées. Un donjon a une entrée, une sortie et chaque salle a des accès vers d'autres salles (ses voisines). Une salle contient un *gameplay* de question (e.g., compléter le fait en déplaçant le bon pot) ou un *gameplay* de piège (e.g., éviter les pics pour passer à la salle suivante). En fonction des *gameplays* les salles vont posséder différents éléments positionnés. Un élément positionné est un élément de jeu qui possède la capacité spécifiée dans la description du *gameplay* sélectionné, et qui a été positionné dans le type de salle sélectionné. Ces éléments vont avoir différents paramètres. Lorsqu'ils représentent une partie d'une question sur un fait (e.g., proposition, question), les éléments pourront avoir une ou des valeurs à afficher, une information sur le type d'affichage (texte ou image), un paramètre de vérification (e.g., booléen décrivant si la proposition est correcte ou incorrecte), etc.

De plus, une salle est composée des faits questionnés qui sont interrogés dans cette salle. Les faits questionnés sont des questions à propos de faits. Comme précédemment mentionné, un fait brut a une structure différente en fonction du domaine didactique visé. Prenons l'exemple de deux tâches T1 et T2. T1 consiste à choisir parmi un ensemble de propositions la réponse correspondant au résultat de la multiplication pour chaque fait. À partir des paramètres de T1, les faits interrogés sont construits pour donner, par exemple, une question telle que $2 \times 6 = ?$ avec un ensemble de propositions tel que $\{8, 12, 14\}$. T2 consiste à choisir dans un ensemble de propositions celles qui sont des résultats possibles d'une table donnée. À partir des paramètres de T2, les faits questionnés construits donneraient des questions telles que « Quels sont des résultats de la table de 3 ? » ainsi qu'un ensemble de propositions tel que $\{3, 5, 7, 9, 12\}$. Ainsi, pour offrir un algorithme de génération d'activités générique (indépendant de tout domaine didactique), il est nécessaire de posséder un objet générique dont on peut manipuler les données pour construire automatiquement des *gameplays* concrets.

Dans ce but, nous avons proposé une modélisation générique de faits questionnés (2023b). Les faits questionnés peuvent être vus comme des objets possédant une question (texte avec ou sans images), un ensemble de propositions (si la modalité de réponses est du choix parmi des propositions) associées à une valeur de véracité (correcte ou incorrecte), un ensemble de bonnes réponses (si la modalité de réponse est de la saisie) et un nombre de réponses atten-

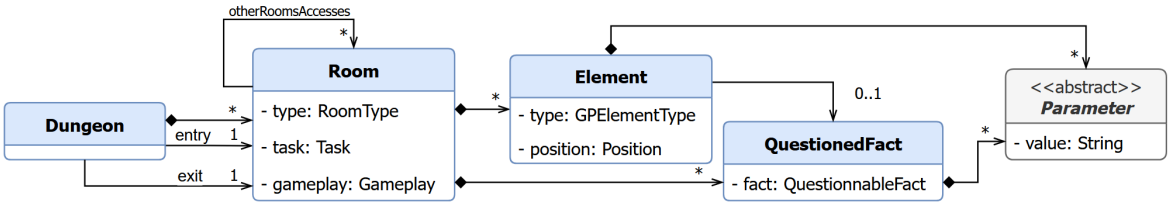


Figure 7 : Modèle conceptuel simplifié d'activités générées (Lemoine & Laforcade, 2023a, 2023b)

dues. En fonction des faits bruts et des paramètres des tâches, les paramètres des faits questionnés sont instanciés ou non. Cette modélisation permet de générer des *gameplays* concrets indépendamment du domaine didactique visé en associant les valeurs des faits questionnés aux objets de jeu correspondant aux composants décrits dans les *gameplays*. La Figure 7 présente une version simplifiée du modèle conceptuel de l'activité à générer.

### 4.2.4. Modèle de l'apprenant-joueur

Afin de permettre la génération d'activités adaptées à l'apprenant-joueur, il est nécessaire de connaitre la progression de l'apprenant, la progression du joueur et ses préférences. Ainsi, pour connaitre la progression d'un apprenant dans son parcours d'entrainement, défini par l'enseignant, ses résultats aux faits questionnés pour chaque tâche d'un couple objectif/niveau doivent être enregistrés. Cet enregistrement permet de calculer le nombre de faits qui ont été vus et travaillés ainsi que le pourcentage de succès pour chaque tâche et chaque niveau.

Dans les modèles précédents, nous avons parlé de faits bruts et de faits questionnés (questions à propos des faits bruts). Les faits questionnés possèdent, en cas de modalité de type « choix », les bonnes propositions et les mauvaises propositions. Cependant, d'après nos discussions avec les experts, il semble plus intéressant d'un point de vue didactique de constamment faire varier les réponses incorrectes. Or, du point de vue de l'apprenant, il est nécessaire de pouvoir comparer, pour une question donnée, les résultats qu'il a obtenus. Cependant, si les résultats sont enregistrés à partir des faits questionnés, la comparaison ne sera pas pertinente puisque les propositions incorrectes des faits questionnés varient à chaque génération. Deux faits questionnés pour un même fait brut seront donc considérés comme distincts. D'autre part, enregistrer les résultats à partir des faits bruts nous fait perdre l'information de la forme de question posée. En conséquence, pour comparer les résultats des apprenants sur un fait donné, il est nécessaire d'avoir une base commune qui ne varie pas (conservation du format de la question, sans les éléments qui varient).

Dans ce but, nous proposons un processus de transformation en deux étapes. Tout d'abord, les faits questionnables sont construits sur la base des faits bruts présents dans le modèle du domaine. Un fait questionnable représente une question sur un fait sans proposition incorrecte. Ces faits sont utilisés pour conserver les résultats des apprenants dans le modèle d'apprenant-joueur (e.g., les temps de réponse, les réponses données). Deuxièmement, les faits questionnés sont construits sur la base de faits questionnables (ce sont des faits questionnables avec des propositions incorrectes). La Figure 8 illustre ce processus.

Les faits questionnables sont construits à partir des tâches spécifiques aux domaines. Ainsi, le concept de fait questionnable est abstrait puisque ces faits possèdent une forme dépendante du domaine. En conséquence, notons que la génération des faits questionnables et questionnés est forcément dépendante du domaine.

De plus, pour générer un donjon correspondant à la progression de jeu (défini dans le

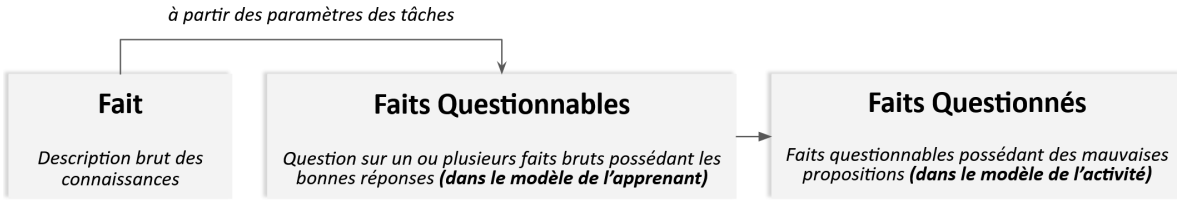


Figure 8 : Processus de transformation des faits bruts en faits questionnés

modèle de jeu), le niveau de jeu actuel du joueur doit être connu. Par exemple, si le dernier donjon réussi était de niveau 8 alors le prochain donjon généré devrait correspondre au niveau 9. Le niveau de jeu impact le donjon en termes de nombre de salles avec et sans questions, de malédictions potentielles et de forme (labyrinthique ou linéaire).

Pour rappel, notre approche consiste à adapter les *gameplays* en fonction des préférences du joueur. En conséquence, les équipements achetés et activés ou non, doivent également être spécifiés. Ainsi, le troisième modèle à fournir au générateur est le modèle de l'apprenant-joueur qui décrit sa progression dans l'entrainement, sa progression dans le jeu et ses préférences. La Figure 9 présente une version simplifiée du modèle conceptuel de l'apprenant-joueur.

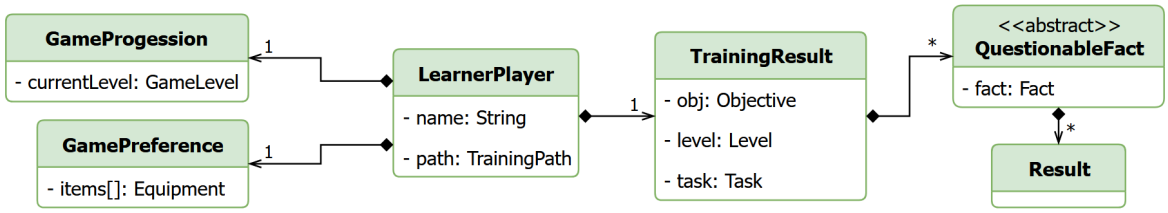


Figure 9 : Modèle conceptuel simplifié de l'apprenant-joueur (2023a)

### 4.2.5. Modèle de relations entre les éléments de jeu et d'entrainement

Précédemment, nous avons mentionné comment associer les éléments des faits questionnés avec des éléments de jeu correspondant à un *gameplay* (proposition de modélisation générique des faits questionnés). Cependant, certains *gameplays* ne sont pas compatibles avec certaines tâches. Par exemple, un *gameplay* consistant à orienter des statues n'a que peu de sens avec une tâche consistant à ordonner des évènements chronologiquement. Ainsi, pour permettre au générateur d'associer correctement des tâches et des *gameplays*, des relations entre ces éléments doivent être définies.

L'entrainement aux connaissances déclaratives est souvent réalisé au moyen des questionnaires et des quiz. En outre, les questionnaires numériques, par rapport aux questionnaires papier, permettent des interactions avec l'utilisateur qui sont plus proches des interactions présentes dans les jeux (clic, déplacement). En conséquence, nous avons proposé une approche permettant d'associer les éléments d'entrainement et de jeu fondée sur l'utilisation des formats de questionnaires numériques comme pivot (Lemoine *et al.*, 2024a). Cette méthode systématique permet la spécification de relations interprétables par la machine entre nos types de tâche (Complétion, Identification, Identification d'Appartenance, Ordonnancement) et nos catégories de *gameplay* (*SELECT*, *MOVE*, *ORIENT*, *POSITION*, *DIRECT RESPONSE*). Cette méthode permet d'obtenir des relations conditionnelles entre une catégorie de *gameplay* et une tâche. Les conditions sont un ensemble de paramètres tels que le

type de question (texte, image, texte à trous), le nombre de réponses attendues, nombre de faits questionnés et la modalité de réponse.

Dans la littérature, les relations entre les éléments éducatifs et de jeu sont rarement explicités à travers des modèles. Au contraire, elles sont souvent directement implémentées dans l'algorithme. Or, proposer un modèle interprétable par la machine permet deux choses : 1) de la modularité, c'est-à-dire la possibilité de modifier les relations sans modifier le code source, et 2) de l'extensibilité, c'est-à-dire l'ajout d'un type de tâche, le modèle peut être étendu en ajoutant les relations entre le nouveau type et les catégories de *gameplays* sans ajout de code (et inversement). En conséquence, le dernier modèle nécessaire au générateur est le modèle des relations. La Figure 10 présente le modèle conceptuel simplifié du modèle des relations entre les éléments de jeu et d'entrainement.

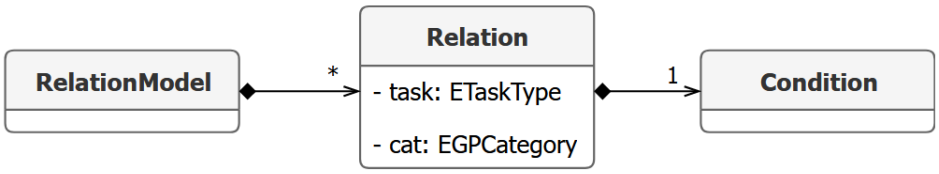


Figure 10 : Modèle conceptuel simplifié des relations (entrainement ↔ jeu) (2023a)

## 4.3. Cadre informatique : une approche d'Ingénierie Dirigée par les Modèles

Pour permettre de guider l'implémentation de générateur d'activités, notre *framework* propose un ensemble de composants logiciels déjà existants limitant le développement de code lors de la création d'extension pour différents domaines didactique. Tout d'abord, cette section présente les méta-modèles correspondants aux modèles conceptuels précédents qui ont été traduits afin d'être instanciables et interprétables pour la génération. Ensuite, les différentes étapes de l'algorithme de génération *(générique)* d'une activité sont présentées. Enfin, l'ensemble des éléments à étendre pour construire un générateur spécifique à un domaine didactique spécifique est détaillé.

### 4.3.1. Méta-modèles et modèles pour la génération

Les méta-modèles et modèles sont conçus à l'aide du *framework* de modélisation d'Eclipse EMF (*Eclipse Modeling Framework*). Ces méta-modèles inter-connectés sont une représentation informatisée des modèles conceptuels présentés dans la section précédente (voir Annexe). Le méta-modèle des connaissances décrit la structure des connaissances du domaine et est à étendre en fonction du domaine didactique visé. Le méta-modèle du domaine décrit la structure des parcours d'entrainement définis par les enseignants en termes d'objectifs, niveaux, tâches, etc. Ce méta-modèle est aussi à étendre au niveau des tâches en fonction du domaine. Le méta-modèle du jeu décrit la structure de l'ensemble des types d'éléments de jeu (*gameplays*, équipements, malédictions, progression de jeu, type de salles, etc.). Le méta-modèle de l'apprenant-joueur décrit la structure de la progression d'un apprenant-joueur dans son entrainement et dans le jeu. Le méta-modèle des relations décrit la structure des conditions sous lesquelles les types de tâches génériques et les catégories de gameplays sont compatibles. Enfin, le méta-modèle de l'activité décrit la structure d'un donjon.

Les « instances » des méta-modèles, c'est-à-dire les modèles conformes aux méta-modèles, sont représentés sous formes de fichier XMI (*XML Metadata Interchange*) et décrivent des informations concrètes sur un domaine donné ou un jeu. La Figure 11 illustre le principe

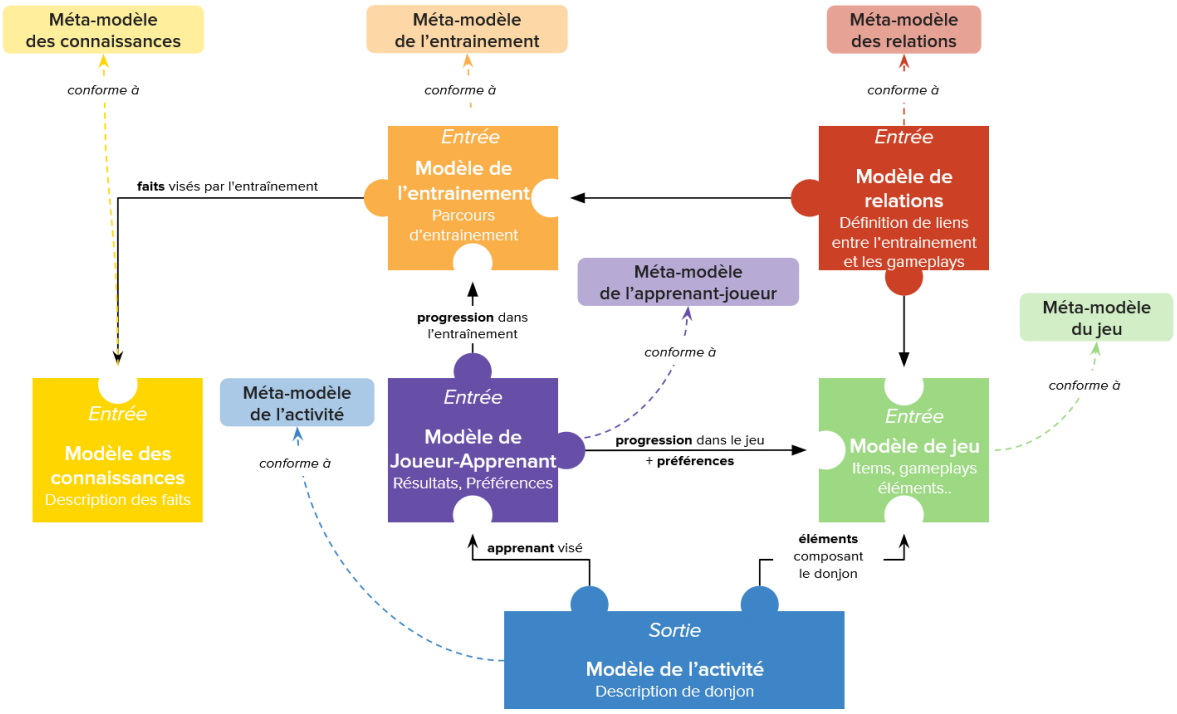


Figure 11 : Illustration des modèles interconnectés conformes aux méta-modèles impliqués dans la génération d'activités

des modèles conformes aux différents méta-modèles dans notre contexte. Les modèles des connaissances permettent de décrire les connaissances brutes à travailler (e.g., les Figures 14 et 17). Les modèles du domaine permettent de décrire des parcours d'entrainement (e.g., Figure 4). Ces instances sont dépendantes du domaine didactique. L'instance du modèle de jeu décrit l'ensemble des éléments de jeu concret disponibles pour le générateur (nous en avons défini un par défaut). Le méta-modèle de l'apprenant-joueur doit être instancié pour chaque apprenant-joueur. Les modèles d'apprenant-joueur doivent être automatiquement mis à jour après chaque niveau réalisé pour permettre à l'algorithme de génération de prendre en compte les nouveaux résultats. Cette mise à jour doit être effectuée par le jeu, en tant que composant indépendant du framework. Ensuite, nous avons « instancié » le méta-modèle des relations en fonction des résultats de la méthode d'association que nous avons précédemment proposée (Lemoine *et al.*, 2024a). Ce modèle ne doit normalement ni être ré-instancié, ni être changé, sauf en cas d'ajout de nouvelle tâche générique ou de désaccord avec notre approche. Enfin, le méta-modèle de l'activité (donjon) est « instancié » par l'algorithme de génération à partir des informations présentes dans les modèles d'entrée fournis (tous les autres).

### 4.3.2. Algorithme de génération d'activité

L'algorithme de génération d'une activité est implémenté en Java et s'effectue en plusieurs étapes incrémentales (similaire à Laforcade et Laghouaouta (2018) et Sehaba et Hussaan (2013)). Plus précisément, l'algorithme se décompose en quatre parties découpées en neuf étapes : 1) sélection de l'ensemble des éléments du donjon (étapes une à six), 2) création de la structure (septième étape), 3) instanciation du donjon (huitième étape) et 4) transformation du donjon (neuvième étape). La Figure 12 décrit les neuf étapes de l'algorithme. Tout d'abord, le nombre de salles avec et sans question est calculé en fonction du niveau actuel du joueur dans le jeu et de la progression de jeu définie. Ensuite, un couple objec-

tif/niveau éligible du parcours d'entrainement de l'apprenant est sélectionné aléatoirement. Notons que tous les choix aléatoires réalisés par l'algorithme pourraient faire l'objet d'une heuristique donnée (e.g., stratégies donnés par l'enseignant pour le choix de l'objectif niveau sélectionné (El-Kechaï *et al.*, 2015)). En fonction du taux d'apparition des tâches du niveau sélectionné et du nombre de salles avec question, les tâches sont choisies et ordonnées pour les salles avec question du donjon. Pour continuer, les faits questionnables du couple objectif/niveau sont générés si ce dernier n'avait jamais été sélectionné auparavant. Ensuite, le nombre nécessaire de faits questionnés pour le donjon est généré en fonction des tâches préalablement sélectionnées. Notons, que les méthodes de génération de faits appelées par l'algorithme générique sont spécifiques au domaine didactique. Par la suite, un *gameplay* de piège ou de question ainsi qu'un ensemble d'éléments de jeu à instancier compatibles avec ce *gameplay* sont sélectionnés pour chaque salle. La dernière étape de sélection consiste à choisir les malédictions du donjon (e.g., niveau dans le noir, donjon labyrinthique) en fonction du niveau de jeu atteint par l'apprenant, de la progression de jeu définie et avec une probabilité d'une sur deux.

Lorsque l'ensemble des éléments est choisi, l'algorithme génère la structure du donjon (linéaire ou labyrinthique). La génération des donjons repose sur le principe du *Grid-Based Dungeon Generator*, c'est-à-dire que l'espace est divisé en cellules dans lesquelles des salles peuvent être placées. Cependant, notre approche diffère puisque nous gérons deux types de salles ($petite = 1$ cellule et $grande = 4$ cellules en carré), créant un besoin de gestion des chevauchements de salles. Pour la génération de donjon linéaire, l'algorithme est fondé sur un principe de *backtrack* afin d'éviter les cul-de-sacs dus aux types de salle n'ayant pas tous les accès possibles (nord, sud, est, ouest, nord-est...) rendant la génération impossible. Ce problème ne survient pas pour les donjons labyrinthiques puisque chaque salle est éligible à chaque itération.

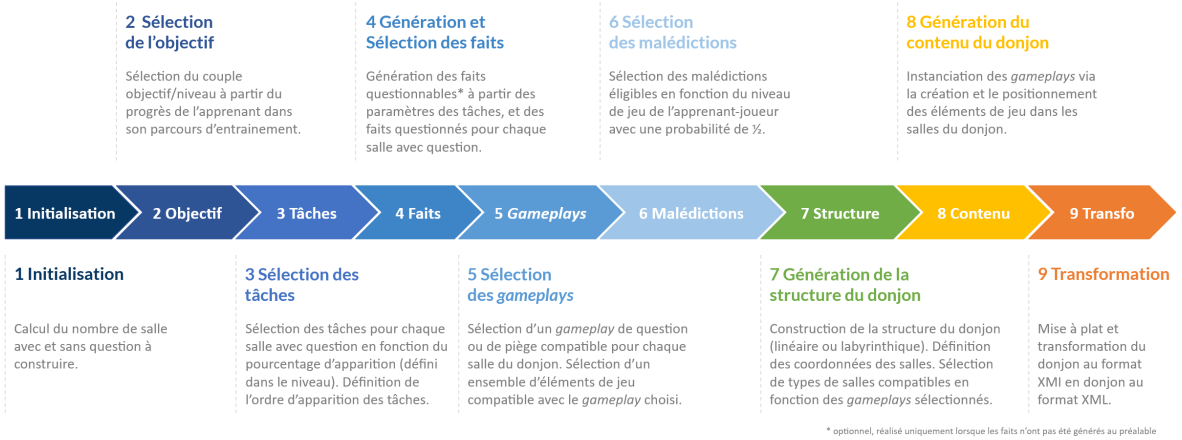


Figure 12 : Étapes de l'algorithme de génération d'activité

L'étape suivante consiste à générer le contenu du donjon en termes d'éléments de jeu. L'objectif est d'instancier correctement les éléments sélectionnés pour chaque salle du donjon en fonction des *gameplays* et des faits questionnés. Dans ce but, les valeurs des éléments de jeu (e.g., textes à afficher, propositions portées par les objets) sont définies afin de permettre le questionnement d'un fait spécifique en associant les paramètres des faits questionnés avec ceux décrits dans les *gameplays*.

Enfin, la dernière étape est une étape opérationnelle qui consiste à transformer le modèle XMI du donjon en fichier XML. Les intérêts principaux de cette transformation sont de permettre de supprimer les références vers d'autres modèles ou fichiers XMI (ce qui permet

d'aplanir le modèle) et de rendre le système plus portable. Cette transformation est réalisée à l'aide du langage *Epsilon Transformation Language* (ETL).

#### 4.3.3. Règles d'extension

Ce *framework* doit être étendu au niveau des méta-modèles, des modèles et du code pour permettre la conception d'un générateur spécifique à un domaine. Au niveau des méta-modèles présentés, les points d'extension sont représentés par des classes abstraites (n'ayant pas déjà des sous-classes concrètes). Trois méta-modèles sont visés par cette extension : le méta-modèle des connaissances, le méta-modèle de l'entrainement et le méta-modèle de l'apprenant-joueur. Comme précédemment mentionné, les niveaux et les tâches d'entrainement peuvent posséder des paramètres spécifiques au domaine visé. Par exemple, la manière de construire les tables (opérande $\times$ table ou table $\times$ opérande) ne dépendent que du domaine des mathématiques. Ou bien, dans une tâche de complétion (e.g., retrouver la date historique, ou le résultat d'une multiplication), l'élément à trouver dépend du domaine (e.g., résultat, opérande ou table pour les multiplications, ou bien évènement ou date pour l'histoire). En conséquence, ces éléments doivent être modélisés en fonction du domaine. De plus, les faits bruts et questionnables ont des formes différentes en fonction du domaine didactique. Par exemple, un fait d'une table de multiplication peut être représenté comme une classe avec trois entiers $x$ (opérande), $y$ (opérande) et $res$ (résultat). En revanche, une date historique serait plutôt représentée par une classe avec une chaine de caractère (évènement) et une date ou période (entiers). Les faits ne peuvent donc pas être ni modélisés ni générés de manière générique.

En conséquence, les algorithmes de génération de faits questionnables et questionnés sont dépendants du domaine et doivent être implémentés au cours de l'extension du *framework*. L'algorithme de génération de faits du *framework* suit un patron de conception appelée *template-method* qui permet d'avoir une partie principale indépendante du domaine et des extensions dépendantes du domaine. La partie principale capture le squelette du code de génération, c'est-à-dire, l'ensemble des invariants relatifs à la génération. En revanche, les extensions se concentrent sur les caractéristiques orientées vers les tâches (e.g., la génération de faits questionnables ou questionnés en fonction de paramètres et de la progression actuelle d'un apprenant spécifique). Cette approche permet de soutenir la mise en œuvre de la génération de faits.

Enfin, pour pouvoir générer des activités, un modèle conforme à chaque méta-modèle doit être réalisé : modèle des connaissances décrivant les faits bruts, modèle du jeu décrivant les éléments de jeu à disposition du générateur, modèle des relations permettant au générateur de conserver la cohérence des activités, modèle du domaine décrivant le parcours d'entrainement et modèle d'apprenant-joueur. Le modèle de l'apprenant-joueur ne nécessite que les informations minimales : identifiant, parcours d'entrainement, et création des progressions (jeu et entrainement) vides, puisque ce modèle doit être mis-à-jour en fonction de résultats de l'apprenant-joueur dans le jeu. De plus, certains modèles sont définis par défaut dans le *framework* comme le modèle de jeu et le modèle de relation et sont donc réutilisables.
Pour conclure, étendre le *framework* pour concevoir un générateur nécessite de réaliser trois grandes étapes :

1. Étendre les méta-modèles en fonction du domaine visé au niveau :
   (a) des faits bruts (*AbstractFact*) ;
   (b) des faits questionnables (*AQuestionableFact*) ;
   (c) des niveaux (*Level*) ;
   (d) des tâches (*CompletionTask*, *IdentificationTask*, *MembershipTask*, *OrderingTask*).

2. Spécifier les modèles conformes aux méta-modèles (fichier XMI) :
   (a) un modèle des connaissances ;
   (b) un modèle du domaine (un parcours d'entrainement) ;
   (c) un modèle d'apprenant (avec les informations minimales) ;
   (d) un modèle de jeu (*optionnel*, possibilité d'utiliser celui par défaut) ;
   (e) un modèle des relations (*optionnel*, possibilité d'utiliser celui par défaut).
3. Implémenter les générateurs de faits pour chaque tâche spécifique au domaine didactique en suivant le *template* fourni.

## 5. APPLICATION DU FRAMEWORK

Le *framework* proposé est une architecture produisant des générateurs. Comme défini, ce *framework* doit respecter des propriétés (voir Section 4.1). En conséquence, l'évaluation du *framework* passe dans un premier temps par la validation du respect des propriétés définies. Le *framework* doit posséder deux propriétés : FP1) permettre d'exprimer différents domaines didactiques et FP2) permettre d'exprimer la vision des enseignants sur l'entrainement des apprenants individuellement. Afin d'évaluer FP1, au moins deux extensions du framework doivent être conçues pour évaluer cette propriété. Nous avons actuellement développé une extension pour le domaine des tables de multiplication. Une seconde extension pour le domaine des faits de judo (prises et gestes d'arbitrage) a également été réalisée ainsi qu'une troisième extension pour le domaine des repères d'histoire-géographie du brevet des collèges (non présentée dans cet article). Afin d'évaluer FP2, au moins deux parcours d'entrainement conçus par deux enseignants différents doivent être définis pour évaluer cette propriété.

### 5.1. UN GÉNÉRATEUR POUR L'ENTRAINEMENT AUX TABLES DE MULTIPLICATION

Dans le cadre de l'étude exploratoire menée au sein du projet AdapTABLES (voir Section 3.1), nous avons défini avec des experts en mathématiques cinq tâches d'entrainement (voir Section 4.1) pour les tables de multiplication. De plus, nous avons également défini des paramètres relatifs aux niveaux de difficulté pour la construction des faits questionnables tels que la forme de construction des tables « opérande $\times$ table ou table $\times$ opérande », la position du symbole égal « à gauche ou à droite », l'intervalle minimal et maximal des multiplications interrogées (un enseignant peut vouloir travailler la table de trois uniquement de 1 à 5 : $3 \times 1$, $3 \times 2$, $3 \times 3$, $3 \times 4$, $3 \times 5$). À partir de ce travail, nous avons construit une extension du framework pour les tables de multiplication.

La Figure 13 présente l'extension du méta-modèle en question. Nous avons modélisé un type de fait brut qui représente les multiplications (*MTFact*) composé de trois entiers : la table, l'opérande et le résultat. Pour chaque tâche, nous avons créé un type de fait questionnable spécifique. En règle générale, un ou plusieurs faits questionnables sont construits à partir d'un unique fait brut, cela dépend des paramètres de constructions des tables. Cependant, dans le cas des faits pour la tâche d'identification d'appartenance (identifier les résultats d'une table), un fait questionnable est construit à partir d'un ensemble de faits bruts, en fonction des réglages de la tâche (nombre de choix attendus).

En suivant les règles d'extension, nous avons spécifié le modèle des connaissances (voir Figure 14) et avons développé les générateurs de faits pour chacune des cinq tâches. Actuellement, nous avons deux parcours d'entrainement de deux enseignantes, une de CE1 et une enseignante de CE2 pour des niveaux moyens modélisés (voir Figure 15). D'autre parcours

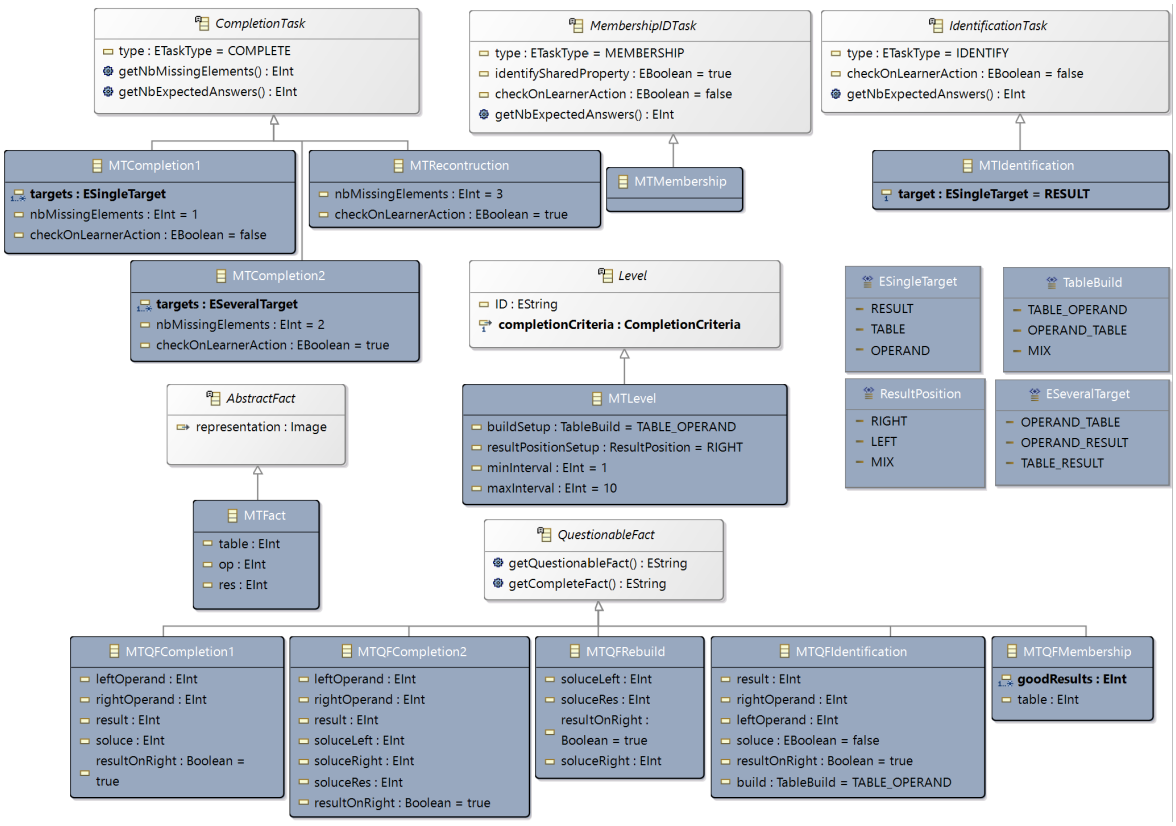


Figure 13 : Extension (en bleu) des méta-modèles du *framework* pour les tables de multiplication

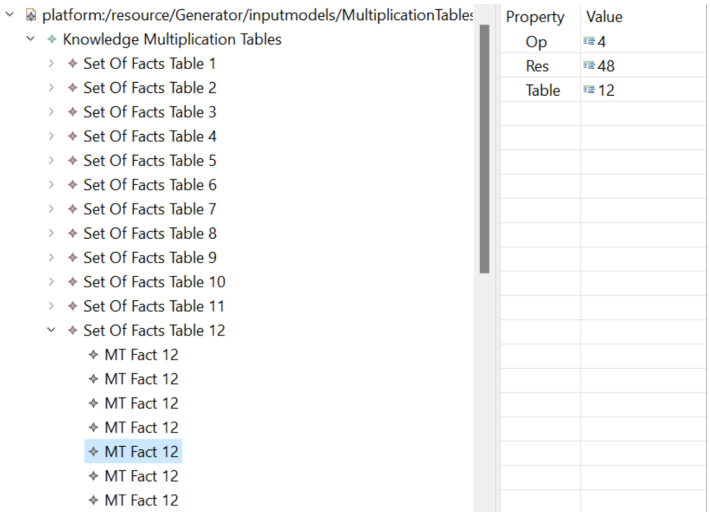


Figure 14 : Modèle des connaissances pour les tables de multiplication

d'entrainement sont en cours de développement, notamment avec ces deux enseignantes qui ont prévu de définir chacune, deux autres parcours pour deux autres niveaux d'apprenant.

## 5.2. Un générateur pour l'entrainement aux faits de judo

Le judo est un art martial japonais fondé en 1882 par Jigorō Kanō. La progression se fait à travers l'apprentissage et la maîtrise de techniques spécifiques et propres à chaque ni-

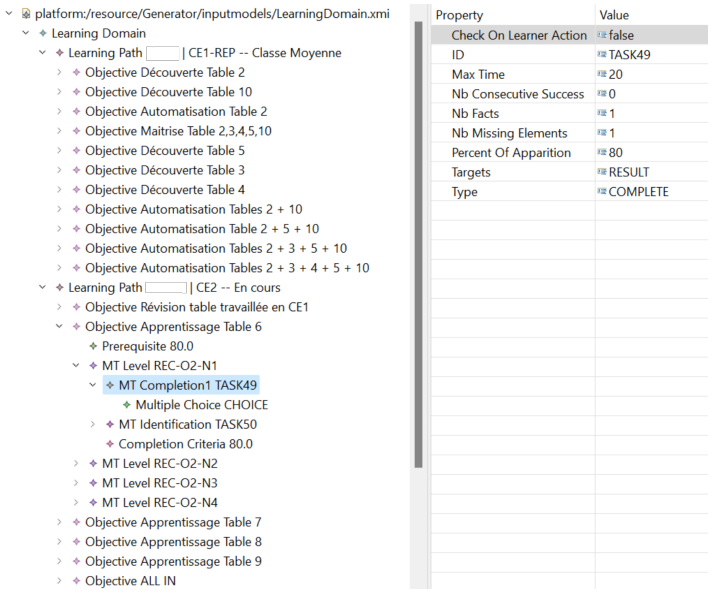


Figure 15 : Modélisation de parcours d'entrainement défini par des enseignantes en mathématiques

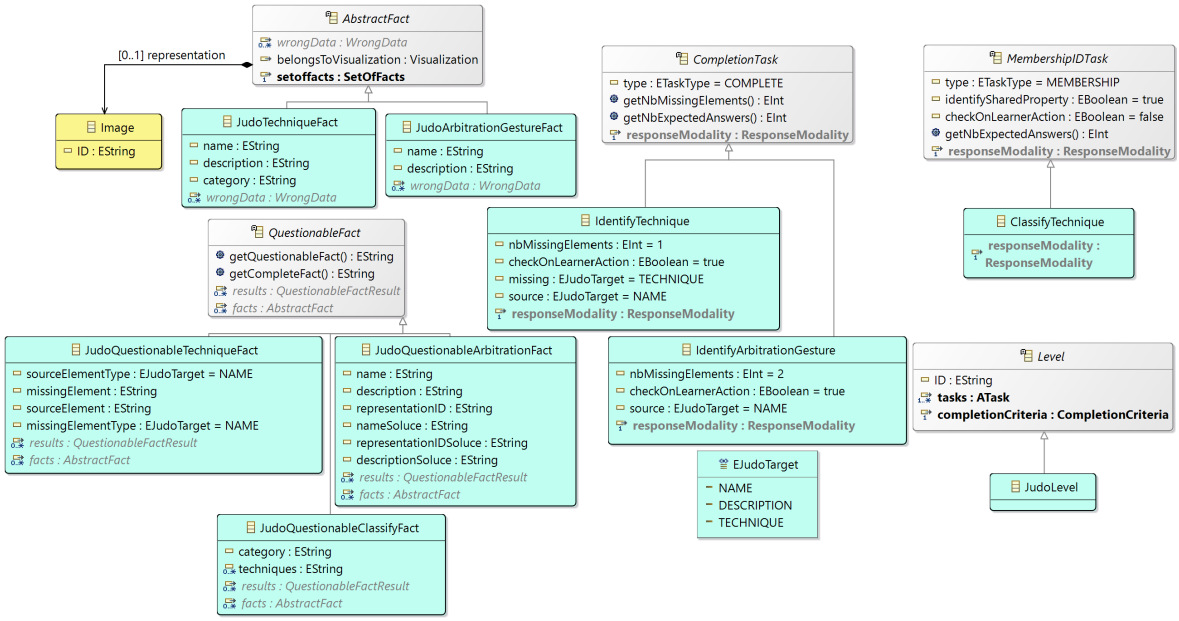


Figure 16 : Extension (couleur cyan) des méta-modèles du *framework* pour le judo

veau. Les judokas évoluent à travers un système de ceintures de couleurs (blanche, jaune, orange, verte, bleu, marron, noire) où chaque ceinture représente un niveau de compétences et de *connaissances* acquises. Les compétences à acquérir ne peuvent évidemment pas être acquises à travers notre approche, il s'agit de savoir-faire. Cependant, les connaissances à acquérir sont des connaissances déclaratives : nommer ou identifier des techniques et connaitre les gestes d'arbitrages (nécessaire à tout combattant). En conséquence, nous proposons une extension du *framework* permettant de s'entrainer aux connaissances demandées au judo (cf. Figure 16).

Nous avons identifié trois tâches : 1) Identification de techniques (*IdentifyTechnique*) consistant à associer deux éléments parmi {nom de la technique, image représentative, des-

cription [3] }, 2) Identification des gestes d'arbitrage (*IdentifyArbitrationGesture*) consistant à reconstituer un fait composé de l'annonce de l'arbitre, la signification de son annonce et le geste associé (e.g., « HAJIME, début du combat, *main-le-long-du-corps* »), et 3) Classification de technique (*ClassifyTechnique*) consistant à identifier les techniques appartenant à une catégorie donnée. Notons qu'au judo les techniques sont classées en fonction de leur catégorie : projection de hanche, projection de jambe, immobilisation, étranglement, clé, sacrifice, etc.

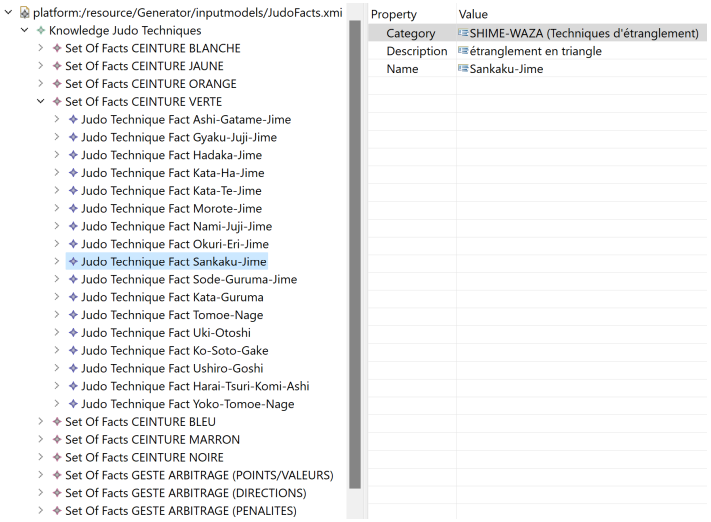


Figure 17 : Modèle des connaissances pour les faits de judo

Comme pour les tables de multiplication, nous avons spécifié le modèle des connaissances (voir Figure 17) en suivant les règles d'extension et avons développé les générateurs de faits pour chacune des trois tâches. Le parcours d'entrainement étant en général le même pour l'ensemble des apprenants, nous avons défini un unique parcours d'entrainement consistant en un objectif par ceinture. Chaque objectif n'a qu'un seul niveau et a pour pré-requis le niveau de la ceinture inférieure (e.g., objectif de la ceinture jaune a pour pré-requis le niveau de l'objectif de la ceinture blanche). Enfin, chaque niveau possède trois tâches : une tâche d'identification de technique, une de geste d'arbitrage et une de classification. Évidemment, d'autres parcours peuvent être créés, ce parcours nous permet d'effectuer des tests et nous sert principalement de preuve de concept.

### 5.3. Un interpréteur d'activités/de niveaux de jeu

**En dehors** du cadre du *framework*, un interpréteur ou *player* de niveau de jeu (d'activités/donjons) a été développé. Cet interpréteur permet : 1) de retranscrire les XML générés en donjon jouables, 2) de visualiser la structure du donjon (carte) et 3) et de jouer. La Figure 18 présente des captures d'écran de salles de donjon de judo et de mathématiques générées et interprétées par le *player*.

## 6. Évaluation du framework

Comme défini, les générateurs produits par le *framework* doivent respecter des propriétés (voir Section 4.1) :

3. Une description est souvent donnée par les enseignants pour décrire les techniques « Ippon-seoi-nage : projection d'épaule par un côté »


(a) Complétion un fait


(b) Reconstitution de fait


(c) Identification de résultats


(d) Identification de technique


(e) Identification de geste d'arbitrage


(f) Classification de technique

Figure 18 : Captures d'écran de salles de donjon interprétées par le *player*

— GP1) les activités générées doivent être adaptées à l'apprenant ;
— GP2) les activités générées doivent être adaptées aux préférences du joueur ;
— GP3) les activités générées doivent être variées en termes d'éléments éducatifs et d'éléments de jeu.

L'évaluation de ces propriétés peut être réalisée par des tests systèmes automatisés (JUnit) consistant à vérifier le respect d'un ensemble d'intentions données.

### 6.1. Validation des activités générées : tests systèmes

Les générateurs construits sont spécifiques à un domaine didactique. En conséquence, vérifier que les activités produites sont adaptées et variées, consiste à vérifier à l'aide de tests systèmes que les activités respectent les propriétés attendues (voir Section 4.1). Les tests systèmes utilisent un générateur spécifique à un domaine didactique donné. Dans notre contexte, les tests systèmes ont été réalisés à partir du générateur des mathématiques. Cependant, l'algorithme étant générique (seule la génération de fait est spécifique au domaine), les résultats peuvent être généralisés (e.g., l'algorithme de sélection d'objectif ne change pas

d'un domaine à un autre).

### 6.1.1. Adaptation des activités à l'apprenant

L'évaluation de l'adaptation aux apprenants consiste à vérifier que les activités générées sont conformes aux prédictions déductibles d'une analyse du parcours d'entrainement associé à l'apprenant. La conformité de l'activité repose sur deux niveaux d'analyse : A) sélection du couple objectif/niveau à travailler et B) tâches présentes dans le donjon en fonction du niveau sélectionné, de la progression de l'apprenant et du pourcentage d'apparition souhaité.

A) SÉLECTION COUPLE OBJECTIF/NIVEAU. L'algorithme de génération doit choisir un couple objectif/niveau éligible (plusieurs couples objectif/niveau peuvent être éligibles en même temps) en fonction de la description du parcours d'entrainement et de la progression de l'apprenant dans ce parcours. Pour évaluer l'algorithme de sélection d'un couple objectif/niveau, nous avons défini un parcours d'entrainement et un modèle d'apprenant fictif incluant les différents cas limites à tester. À partir de ces modèles, nous avons créé une méthode de tests pour chacune des intentions suivantes :

- le couple objectif/niveau sélectionné est éligible ;
- pour 150 générations, tous les couples objectif/niveau éligibles sont apparus au moins une fois ;
- pour 150 générations, aucun des couples objectif/niveau non éligibles n'est apparu ;
- les niveaux dont le pourcentage de faits rencontrés et le pourcentage de succès sont atteints sont considérés comme inéligibles (et donc n'apparaissent jamais dans les donjons).

B) RÉPARTITION DES TÂCHES. L'algorithme de génération doit répartir les tâches en fonction : 1) de leur taux d'apparition, 2) du nombre de salles du donjon dépendant de la progression du joueur dans le jeu (niveau de jeu actuel du joueur). Par exemple, une tâche ayant un taux de 20% devra apparaitre dans 2 salles d'un donjon de taille 10. Notons que lorsque la taille d'un donjon et le taux d'apparition d'une tâche sont petits (e.g., cinq salles et 5% d'apparition), cette tâche peut ne pas apparaitre dans le donjon. En conséquence, des tests doivent être effectués sur deux niveaux de jeu différents (tailles de donjon différentes). Lorsqu'une tâche est terminée, l'algorithme répartit son pourcentage d'apparition proportionnellement aux pourcentages d'apparition des tâches restantes : soit T1[50%], T2[20%], T3[20%], T4[10%] où T1 est achevée, alors les pourcentages des autres tâches deviennent T2[40%], T3[40%], T4[20%].

Pour évaluer la répartition des tâches, nous avons défini deux parcours d'entrainement de test (voir Figure 19). Le premier est composé de 5 tâches avec un taux d'apparition identiques (20%), c'est-à-dire qu'aucune tâche ne prédomine sur les autres. Le second est composé de 6 tâches avec des taux d'apparition différents, c'est-à-dire que certaines tâches sont prédominantes. Quatre situations critiques (cas limites) sont à vérifier : 1) l'apprenant n'a pas commencé l'entrainement, toutes les tâches ont un taux de succès et de faits rencontrés à 0% (toutes les tâches apparaissent proportionnellement à leur pourcentage d'apparition dans le donjon) ; 2) toutes les tâches ont été démarrées, mais aucune n'a été terminée (les deux pourcentages sont supérieurs à 0, toutes les tâches apparaissent proportionnellement à leur pourcentage d'apparition dans le donjon) ; 3) une seule tâche a été terminée (les deux pourcentages de cette tâche ont atteint 100% et la tâche n'apparait plus dans le donjon) ; 4) toutes les tâches, sauf une, ont été terminées (seule la tâche non terminée apparait dans le donjon). Pour chacun des cas limite et chacun des parcours d'entrainement, nous avons spécifié deux modèles d'apprenant-joueur : un modèle dans lequel le joueur est au niveau de jeu 1

(cinq salles avec questions) et un modèle dans lequel le joueur est au niveau de jeu 16 (20 salles avec questions). Ensuite, pour chaque modèle d'apprenant-joueur, une méthode de test vérifiant que le donjon généré possède le bon nombre de salles par tâche a été implémentée.

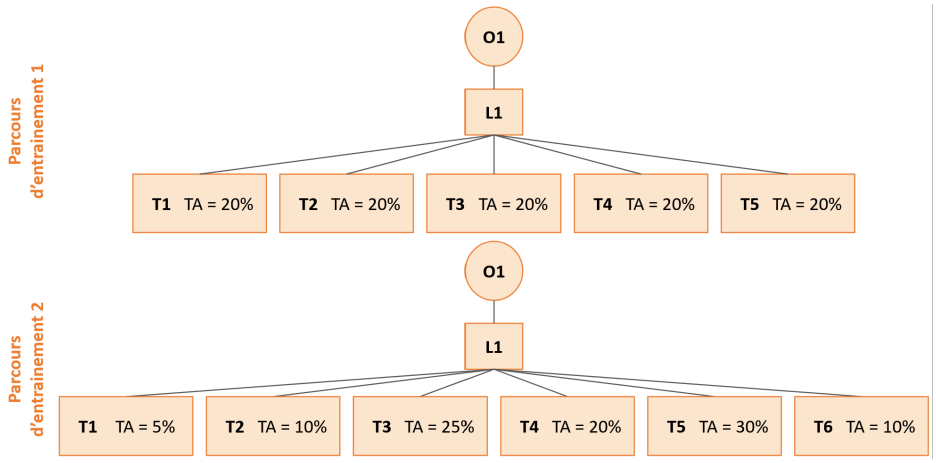


Figure 19 : Parcours d'entrainement fictifs permettant de tester l'algorithme de répartition des tâches

### 6.1.2. Adaptation des activités au joueur

Tout comme l'évaluation de l'adaptation à l'apprenant, l'adaptation au joueur consiste à vérifier que les activités générées sont conformes aux prédictions déductibles d'une analyse des préférences de jeu associées au joueur. Les préférences de jeu du joueur, dans notre contexte, sont décrites au travers d'équipement achetables et activables. Ces équipements permettent de débloquer des capacités et donc de déverrouiller de nouveaux *gameplays*. Un joueur peut activer ou désactiver l'équipement, bloquant alors l'apparition de cette capacité et des gameplays associés en fonction de ses préférences de jeu. Bien sûr, des *gameplays* par défaut, ou gameplays de base, sont définis et non désactivables, sinon le jeu ne serait pas jouable. En conséquence, nous avons implémenté des méthodes de tests représentant les quatre cas limites :

— l'apprenant-joueur n'a effectué aucun achat, alors seuls les *gameplays* par défaut sont présents dans les donjons ;
— l'apprenant-joueur a acheté tous les équipements possibles, mais n'en a activé aucun, alors seuls les *gameplays* par défauts sont présents dans les donjons ;
— l'apprenant-joueur a acheté et activé tous les équipements possibles, alors tous les *gameplays* apparaissent au moins une fois dans un des donjons générés ;
— l'apprenant-joueur a acheté et activé certains équipements, les *gameplays* associés aux équipements achetés et activés apparaissent au moins une fois dans un donjon à un moment donné et les autres n'apparaissent jamais.

### 6.1.3. Variété des activités

La variété des donjons repose à la fois sur le contenu d'entrainement et sur le contenu de jeu. Nous nous intéressons à la variété des activités au niveau : A) de la structure/répartition des donjons et B) des éléments dans les donjons.

A) STRUCTURE/RÉPARTITION DES DONJONS. Dans les *Roguelites*, deux donjons n'ont jamais la même forme (salles, objets). Tout est organisé différemment à chaque fois. L'interpréteur de donjon (voir Section 5.3) permet de visualiser les donjons sans avoir à les explorer. Cet interpréteur permet de visualiser une carte dans laquelle chaque salle est colorée en fonction de la tâche associée et où les accès entre chaque salle sont définis par un petit trait (permet de différencier les donjons labyrinthiques des donjons linéaires). Par conséquent, une première évaluation s'appuie sur une évaluation manuelle consistant à générer, pour un même niveau, plusieurs donjons et à comparer graphiquement leur carte. La Figure 20 présente quatre cartes de donjons générés pour un même objectif ou niveau donné et un même apprenant joueur (avec une progression n'ayant pas changé). On peut facilement remarquer que ces donjons présentent différentes structures et sont répartis de façon non identique.

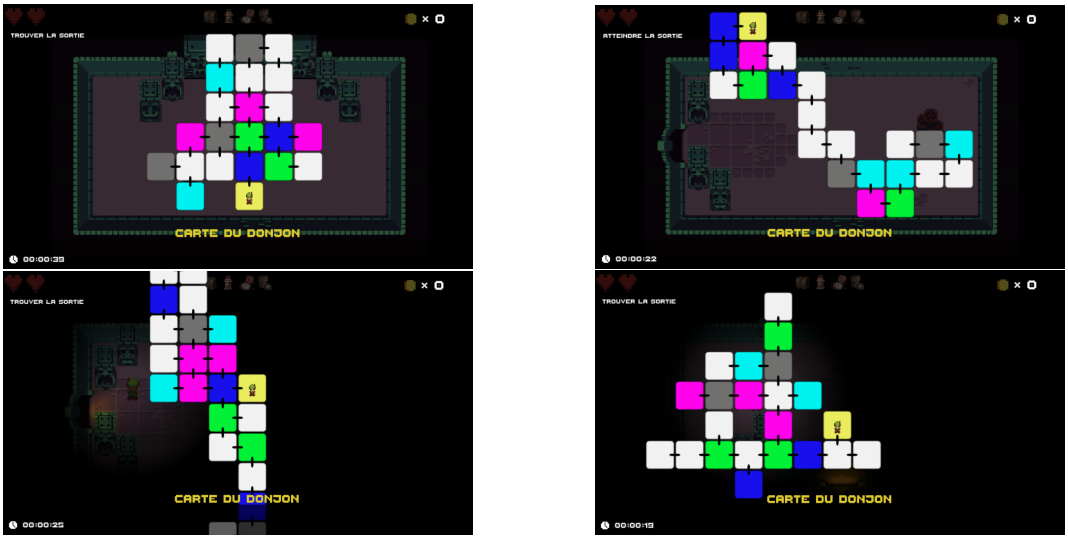


Figure 20 : Cartes de quatre donjons générés pour un même couple objectif/niveau et un même apprenant-joueur

B) ÉLÉMENTS DES DONJONS. La variété des éléments des donjons peut être considérée à différents niveaux : 1) variété des *gameplays* sélectionnés pour chaque tâche ; 2) variété des éléments pour chaque *gameplay* ; 3) variété des types de salle choisis (*RoomType*) ; 4) variété de la position des éléments dans les salles ; 5) variété des mauvais choix des faits questionnés. La variété des types des salles et de la position des éléments est dépendante de la modélisation réalisée dans le modèle de jeu. De plus, la variété des mauvais choix des faits questionnés dépend du domaine didactique visé puisque la méthode créée pour le générer fait partie de l'extension. La variété des *gameplays* sélectionnés pour chaque tâche dépend de deux choses : 1) la variété de *gameplays* décrit dans le modèle de jeu et 2) les achats effectués par le joueur. Pour chacune des capacités (capacités par défaut et capacités verrouillables), différents *gameplays* et variantes de *gameplays* ont été définis (e.g., une variante de "pousser des éléments vers la droite" est de "pousser des éléments vers la gauche") afin que chaque tâche apparaisse avec au moins deux *gameplays* différents ou deux variantes de *gameplays*. Cet aspect (la sélection d'au moins deux modes de jeu différents pour chaque tâche sur 100 générations de donjons) a été évalué pour un joueur n'ayant rien acheté et pour un joueur ayant tout acheté et tout activé. Enfin, la variété des éléments pour chaque *gameplay* dépend majoritairement de la variété des éléments de jeu décrit dans le modèle de jeu. Nous n'évaluons pas formellement ce critère pour le moment, cependant, l'approche semble

prometteuse puisque nous trouvons des variantes d'un même gameplay en termes d'éléments de jeu dans les donjons. Par exemple, nous avons une capacité nommée *CATCHABLE* pour laquelle nous avons deux éléments de jeu associés : le lapin et la vache. Le *gameplay* associé à la capacité *CATCHABLE* apparait dans les donjons avec les deux éléments de jeu différents (voir Figure 21).

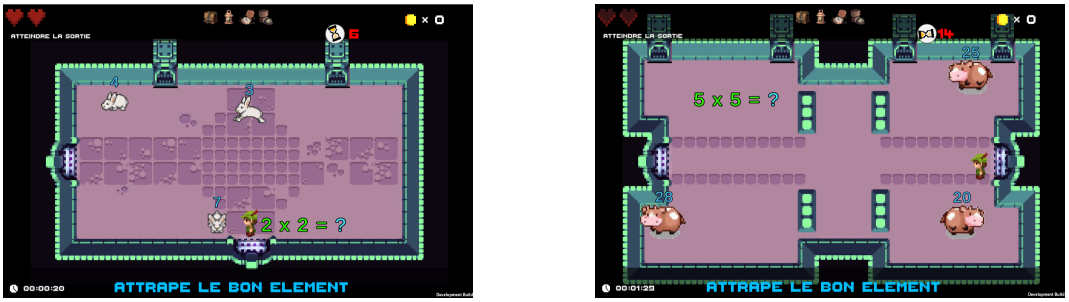


Figure 21 : Deux variantes d'un même *gameplay*

## 6.2. Vérification des modèles : validation de contraintes statiques

Une autre étape de vérification du *framework* consiste à valider les modèles qu'il utilise. Si de nombreuses informations sont capturées par les méta-modèles, ces derniers ne permettent pas de capturer l'ensemble des propriétés sémantiques voulues pour un domaine donné. Par exemple, dans notre modèle du domaine actuel (voir Figure 24 en annexe) rien ne nous empêche de définir qu'un objectif a pour pré-requis son propre niveau. Ce qui sémantiquement n'a aucun sens. La validation de contraintes statiques sur les modèles est une méthode semi-formelle de vérification automatique de modèles consistant à définir un ensemble de contraintes et à valider ou non les modèles en fonction de ces contraintes. Pour garantir la bonne sémantique des modèles (génériques) fournis en entrée des générateurs, nous avons défini un ensemble de contraintes sémantiques sur les modèles à l'aide du langage *Epsilon Validation Language* (EVL).

```
/*
Checks if every SetOfFacts of an Objective, belongs to the
corresponding Knowledge of the objective's path.
*/
constraint objectiveFactsBelongsToPathKnowledge {
    check {
        return self.objectives.forAll(obj |
        obj.setoffacts.forAll(facts |
        self.knowledge.knowledgefacts.contains(facts)));
    }

    message {
        return "SetOfFacts does not belong to the correct knowledge";
    }
}
```

Figure 22 : Exemple de règle de validation de modèles écrite en EVL vérifiant si les faits associés à un objectif appartiennent au modèle de connaissances associé au parcours concerné

Par exemple, les règles définies pour la validation du modèle du domaine sont :

- les faits d'un objectif appartiennent aux connaissances associées au parcours d'entrainement (la Figure 22 illustre cette règle en EVL) ;
- le niveau (*requiredLevel*) d'un pré-requis d'un objectif O, n'est pas un niveau de l'objectif O ;

— les pourcentages des succès et de faits rencontrés (*successPercent* et *encounters-Percent*) des pré-requis et des critères de complétion ont des valeurs entre 0 et 100) ;
— les pourcentages de succès et de faits rencontrés des pré-requis doivent être inférieurs ou égaux aux pourcentages du critère de complétion (sinon ce n'est pas atteignable).

## 7. Conclusion & Perspectives

Cet article présente un *framework* guidant la conception et l'implémentation de générateurs d'activités d'entrainement de jeu de type Roguelite aux connaissances déclaratives. Le *framework* est une architecture logicielle extensible fondée sur une approche d'Ingénierie Dirigée par les Modèles, c'est-à-dire utilisant des modèles et méta-modèles interprétables par la machine. Ce *framework* permet de faciliter l'implémentation de générateurs d'activités, par des ingénieurs ou développeurs, en limitant le développement nécessaire aux informations reliées aux connaissances déclaratives du domaine didactique. Ainsi, ce *framework* propose un ensemble d'algorithmes, modèles, et méta-modèles déjà existants à étendre qui seront donc réutilisés pour spécifier un générateur d'activités.

Pour évaluer, la généricité et l'extensibilité du *framework*, trois générateurs ont été produits à partir de celui-ci : un générateur pour l'entrainement aux tables de multiplication, un second pour les techniques et les gestes d'arbitrages de judo, et un troisième pour les repères d'histoire-géographie au programme du brevet des collèges. Les propriétés d'adaptation à l'apprenant-joueur et de variété des activités ont été évaluées à travers des tests automatisés.

À court terme, nous souhaitons automatiser les tests au niveau des variantes d'éléments pour un *gameplay* donné. Pour rappel, des tests permettant de vérifier la variété de *gameplays* proposés pour une tâche d'entrainement donnée ont été implémentés. Ces tests complémentaires consisteraient à vérifier que, pour un même *gameplay* sélectionné, différents éléments de jeu ayant les capacités référencées dans ce gameplay sont choisis par l'algorithme de génération.

Également, à court-terme, nous préparons l'expérimentation directe du framework par des experts et ingénieurs maitrisant l'IDM. Ils auront à étendre le framework à un domaine que nous leur fournirions : spécifications non informatiques des faits à entrainer, les tâches d'entrainement souhaitées, etc. Il s'agira d'évaluer la *documentation* guidant cette *extension* ainsi que la facilité/difficulté de l'extensibilité du framework (*effort* de modélisation et de codage). L'IDM étant un cadre théorique et pratique non maitrisé par l'ensemble des ingénieurs, une perspective possible, à plus long terme, serait d'instrumenter les activités de l'expert en proposant un environnement logiciel interactif pour assister la création d'extensions pour différents domaines didactiques.

À moyen terme, nous aimerions également prendre en compte d'autres composantes pédagogiques de l'entrainement qui pourraient faire l'objet d'adaptations à l'apprenant. En effet, actuellement, soit la génération ne précise pas certaines informations qui sont laissées à l'appréciation du jeu (e.g., le message lorsque la réponse donnée est correcte ou mauvaise), soit la génération réalise des choix au hasard (e.g., le choix du couple objectif/niveau à entrainer parmi ceux éligibles). Ces choix arbitraires, ou laissés au hasard, peuvent faire l'objet de nouvelles adaptations. Par exemple, les *feedbacks* pourraient être personnalisés en fonction de préférences de l'élève ou bien de l'enseignant, ils pourraient également mieux prendre en compte le contexte de réponse (temps de réponse, erreurs passées pour le même fait, etc.). De plus, le choix pour chaque niveau de jeu du couple objectif et niveau d'entrainement visés pourrait s'appuyer sur des stratégies pédagogiques précisées par l'enseignant pour chaque élève (e.g., maitriser d'abord un objectif avant de passer à un autre).

À plus long terme, nous souhaiterions également nous intéresser à la génération d'activités de jeu pour d'autres types de connaissances : les connaissances procédurales. Ces connaissances plus complexes nécessitent d'entrainer la réalisation d'actions spécifiques dans un contexte didactique donné. Une génération adaptée et variée nécessiterait de s'intéresser au besoin de l'entrainement de ces connaissances, mais également à d'autres genres de jeu pouvant répondre à ces besoins.

## RÉFÉRENCES


Amory, A. (2007). Game object model version II : a theoretical framework for educational game development. *Education Tech Research Dev*, *55*(1), 51-77. https://doi.org/10.1007/s11423-006-9001-x

Bakkes, S., Tan, C. T., et Pisan, Y. (2012). Personalised gaming : a motivation and overview of literature. *Proceedings of The 8th Australasian Conference on Interactive Entertainment : Playing the System*, 1-10. https://doi.org/10.1145/2336727.2336731

Bezza, A., Balla, A., et Marir, F. (2013). An approach for personalizing learning content in e-learning systems : A review. *2013 Second International Conference on E-Learning and E-Technologies in Education (ICEEE)*, 218-223. https://doi.org/10.1109/ICeLeTE.2013.6644377

Bontchev, B. P., Terzieva, V., et Paunova-Hubenova, E. (2021). Personalization of serious games for learning. *ITSE*, *18*(1), 50-68. https://doi.org/10.1108/ITSE-05-2020-0069

Brame, C. J., et Biel, R. (2015). Test-enhanced learning : the potential for testing to promote greater learning in undergraduate science courses. *LSE*, *14*(2). https://doi.org/10.1187/cbe.14-11-0208

Callies, S., Sola, N., Beaudry, E., et Basque, J. (2015). An empirical evaluation of a serious simulation game architecture for automatic adaptation. *R. Munkvold & L. Kolas, Proceedings of the 9th European Conference on Games Based Learning (ECGBL 2015)*, 107-116.

Carpentier, K., et Lourdeaux, D. (2014). Generation of learning situations according to the learner's profile within a virtual environment. Dans J. Filipe et A. Fred (dir.), *Agents and Artificial Intelligence* (p. 245-260, T. 449). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-44440-5_15

Carvalho, M. B., Bellotti, F., Berta, R., De Gloria, A., Sedano, C. I., Hauge, J. B., Hu, J., et Rauterberg, M. (2015). An activity theory-based model for serious games analysis and conceptual design. *Computers & Education*, *87*, 166-181. https://doi.org/10.1016/j.compedu.2015.03.023

Codish, D., et Ravid, G. (2015). Adaptive approach for gamification optimization. *2014 IEEE/ ACM 7th International Conference on Utility and Cloud Computing, UCC 2014*, 609-610. https://doi.org/10.1109/UCC.2014.94

Dias, T. (2018). *Enseigner les mathématiques à l'école*. Magnard.

Diwan, C., Srinivasa, S., et Ram, P. (2019). Automatic generation of coherent learning pathways for open educational resources. Dans M. Scheffel, J. Broisin, V. Pammer-Schindler, A. Ioannou et J. Schneider (dir.), *Transforming Learning with Meaningful Technologies* (p. 321-334, T. 11722). Springer International Publishing. https://doi.org/10.1007/978-3-030-29736-7_24

Djaouti, D., Alvarez, J., Jessel, J.-P., Methel, G., et Molinier, P. (2008). A gameplay definition through videogame classification. *International Journal of Computer Games Technology*, 1-7. https://doi.org/10.1155/2008/470350

Dormans, J., et Bakkes, S. (2011). Generating missions and spaces for adaptable play experiences. *IEEE Trans. Comput. Intell. AI Games*, *3*(3), 216-228. https://doi.org/10.1109/TCIAIG.2011.2149523

El-Kechaï, N., Melero, J., et Labat, J.-M. (2015). Adaptation de serious games selon la stratégie choisie par l'enseignant : approche fondée sur la Competence-based Knowledge Space Theory. *7ème Conférence sur les Environnements Informatiques pour l'Apprentissage Humain (EIAH 2015)*, 294-305.

Grant, P., et Basye, D. (2014). *Personalized learning : a guide for engaging students with technology*. International Society for Technology in Education.

Holohan, E., Melia, M., McMullen, D., et Pahl, C. (2006). The generation of e-learning exercise problems from subject ontologies. *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, 967-969. https://doi.org/10.1109/ICALT.2006.1652605

Hunicke, R., LeBlanc, M., et Zubek, R. (2004). MDA : a formal approach to game design and game research. *Proceedings of the AAAI Workshop on Challenges in Game AI*, *4*.

Ismail, H., et Belkhouche, B. (2018). A reusable software architecture for personalized learning systems. *2018 International Conference on Innovations in Information Technology (IIT)*, 105-110. https://doi.org/10.1109/INNOVATIONS.2018.8605997

Jézéquel, J.-M., Combemale, B., et Vojtisek, D. (2012, février). *Ingénierie dirigée par les modèles : des concepts à la pratique...* Ellipses.

Junior, R., et Silva, F. (2021). Redefining the MDA framework—the pursuit of a game design ontology. *Information*, *12*(10). https://doi.org/10.3390/info12100395

Kent, S. (2002). Model driven engineering. *Integrated Formal Methods*, 286-298.

Kim, J. W., Ritter, F. E., et Koubek, R. J. (2013). An integrated theory for improved skill acquisition and retention in the three stages of learning. *Theoretical Issues in Ergonomics Science*, *14*(1), 22-37. https://doi.org/10.1080/1464536X.2011.573008

Laforcade, P., et Laghouaouta, Y. (2018). Generation of adapted learning game scenarios : a model-driven engineering approach. Dans B. M. McLaren, R. Reilly, S. Zvacek et J. Uhomoibhi (dir.), *Computer Supported Education - 10th International Conference, CSEDU 2018, Funchal, Madeira, Portugal, March 15-17, 2018, Revised Selected Papers* (p. 95-116, T. 1022). Springer. https://doi.org/10.1007/978-3-030-21151-6_6

Laforcade, P., Mottier, E., Jolivet, S., et Lemoine, B. (2022). Expressing adaptations to take into account in generator-based exercisers : an exploratory study about multiplication facts. *14th International Conference on Computer Supported Education*. https://doi.org/10.5220/0011033100003182

Lemoine, B., et Laforcade, P. (2023a). Generator of personalised training games activities : a conceptual design approach. Dans P. Dondio, M. Rocha, A. Brennan, A. Schönbohm, F. de Rosa, A. Koskinen et F. Bellotti (dir.), *Games and Learning Alliance - 12th International Conference, GALA 2023, Dublin, Ireland, November 29 - December 1, 2023, Proceedings* (p. 321-331, T. 14475). Springer. https://doi.org/10.1007/978-3-031-49065-1_31

Lemoine, B., et Laforcade, P. (2023b). Mapping facts to concrete game elements for generation purposes : a conceptual approach. Dans P. Dondio, M. Rocha, A. Brennan, A. Schönbohm, F. de Rosa, A. Koskinen et F. Bellotti (dir.), *Games and Learning Alliance - 12th International Conference, GALA 2023, Dublin, Ireland, November 29 - December 1, 2023, Proceedings* (p. 342-352, T. 14475). Springer. https://doi.org/10.1007/978-3-031-49065-1_33

Lemoine, B., Laforcade, P., et George, S. (2023). Un Framework de conception pour des générateurs d'activités de jeu variées et adaptées. *11ème Conférence Sur Les Environnements Informatiques Pour l'Apprentissage Humain*, 88-99.

Lemoine, B., Laforcade, P., et George, S. (2024a). An approach for mapping declarative knowledge training task types to gameplay categories. Dans B. M. McLaren, J. Uhomoibhi, J. Jovanovic et I.-A. Chounta (dir.), *Computer Supported Education* (p. 47-68). Springer Nature Switzerland.

Lemoine, B., Laforcade, P., et George, S. (2024b). Designing declarative knowledge training games : an analysis framework based on the roguelite genre. Dans B. M. McLaren, J. Uhomoibhi, J. Jovanovic et I.-A. Chounta (dir.), *Computer Supported Education* (p. 69-92). Springer Nature Switzerland.

Marne, B., Carron, T., Labat, J.-M., et Marfisi-Schottman, I. (2013). MoPPLiq : a model for pedagogical adaptation of serious game scenarios. *IEEE 13th International Conference on Advanced Learning Technologies, ICALT 2013, Beijing, China, July 15-18, 2013*, 291-293. https://doi.org/10.1109/ICALT.2013.90

Marty, J.-C., et Carron, T. (2011). Hints for improving motivation in game-based learning environments. Dans *Handbook of Research on Improving Learning and Motivation through Educational Games : Multidisciplinary Approaches* (p. 530-549). IGI Global.

Monterrat, B., Yessad, A., Bouchet, F., Lavoué, É., et Luengo, V. (2017). MAGAM : a multi-aspect generic adaptation model for learning environments. Dans É. Lavoué, H. Drachsler, K. Verbert, J. Broisin et M. Pérez-Sanagustín (dir.), *Data Driven Approaches in Digital Education* (p. 139-152, T. 10474). Springer International Publishing. https://doi.org/10.1007/978-3-319-66610-5_11

Natkin, S., Yan, C., Jumpertz, S., et Marquet, B. (2007). Creating multiplayer ubiquitous games using an adaptive narration model based on a user's model. Dans A. Baba (dir.), *Proceedings of the 2007 DiGRA International Conference : Situated Play, DiGRA 2007, Tokyo, Japan, September 24-28, 2007*. Digital Games Research Association.

Plass, J. L., et Pawar, S. (2020). Toward a taxonomy of adaptivity for learning. *Journal of Research on Technology in Education*, *52*(3), 275-300. https://doi.org/10.1080/15391523.2020.1719943

Prensky, M. (2005). *Computer games and learning : digital game-based learning*. Handbook of Computer Game Studies.

Roediger, H. L., et Pyc, M. A. (2012). Inexpensive techniques to improve education : applying cognitive psychology to enhance educational practice. *Journal of Applied Research in Memory and Cognition*, *1*(4), 242-248. https://doi.org/10.1016/j.jarmac.2012.09.002

Roepke, R., Drury, V., Schroeder, U., et Meyer, U. (2021). A modular architecture for personalized learning content in anti-phishing learning games. *Software Engineering (Satellite Events)*.

Sehaba, K., et Hussaan, A. M. (2013). GOALS : generator of adaptive learning scenarios. *IJLT*, *8*(3), 224. https://doi.org/10.1504/IJLT.2013.057061

Sina, S., Rosenfeld, A., et Kraus, S. (2014). Generating content for scenario-based serious-games using crowdsourcing. *Proceedings of the National Conference on Artificial Intelligence*, *1*, 522-529.

Smith, R. P. (1981). Boredom : a review. *Human factors*, *23*(3), 329-340.

Streicher, A., et Smeddinck, J. D. (2016). Personalized and adaptive serious games. Dans R. Dörner, S. Göbel, M. Kickmeier-Rust, M. Masuch et K. Zweig (dir.), *Entertain-*

*ment Computing and Serious Games* (p. 332-377, T. 9970). Springer International Publishing. https://doi.org/10.1007/978-3-319-46152-6_14

Tchounikine, P., Mørch, A. I., et Bannon, L. J. (2009). A computer science perspective on technology-enhanced learning research. Dans *Technology-Enhanced Learning* (p. 275-288). Springer Netherlands. https://doi.org/10.1007/978-1-4020-9827-7_16

Vandewaetere, M., Desmet, P., et Clarebout, G. (2011). The contribution of learner characteristics in the development of computer-based adaptive learning environments. *Computers in Human Behavior*, *27*(1), 118-130. https://doi.org/10.1016/j.chb.2010.07.038

## ANNEXE : MÉTA-MODÈLES POUR LA GÉNÉRATION

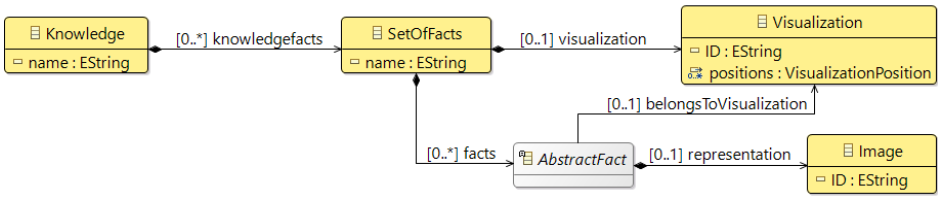


Figure 23 : Méta-modèle des connaissances

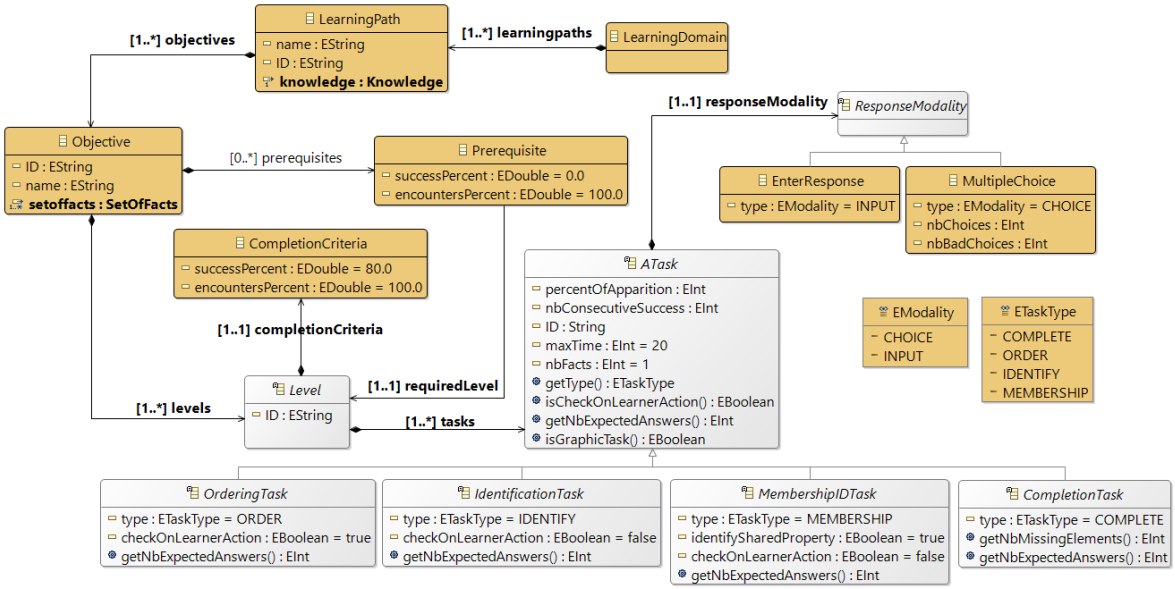


Figure 24 : Méta-modèle de l'entrainement

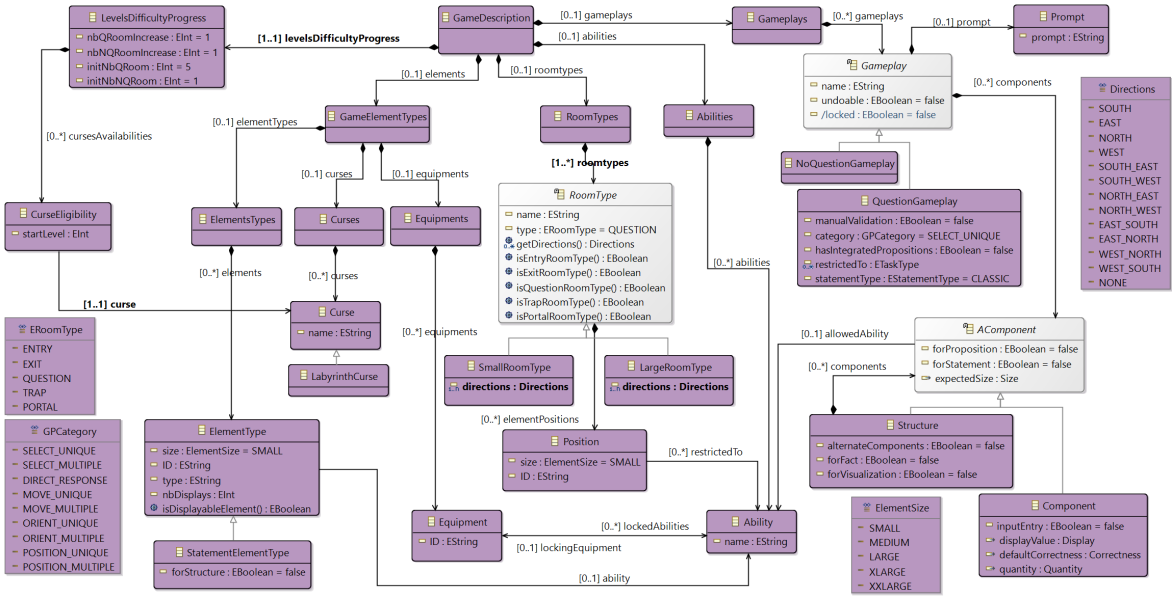


Figure 25 : Méta-modèle du jeu

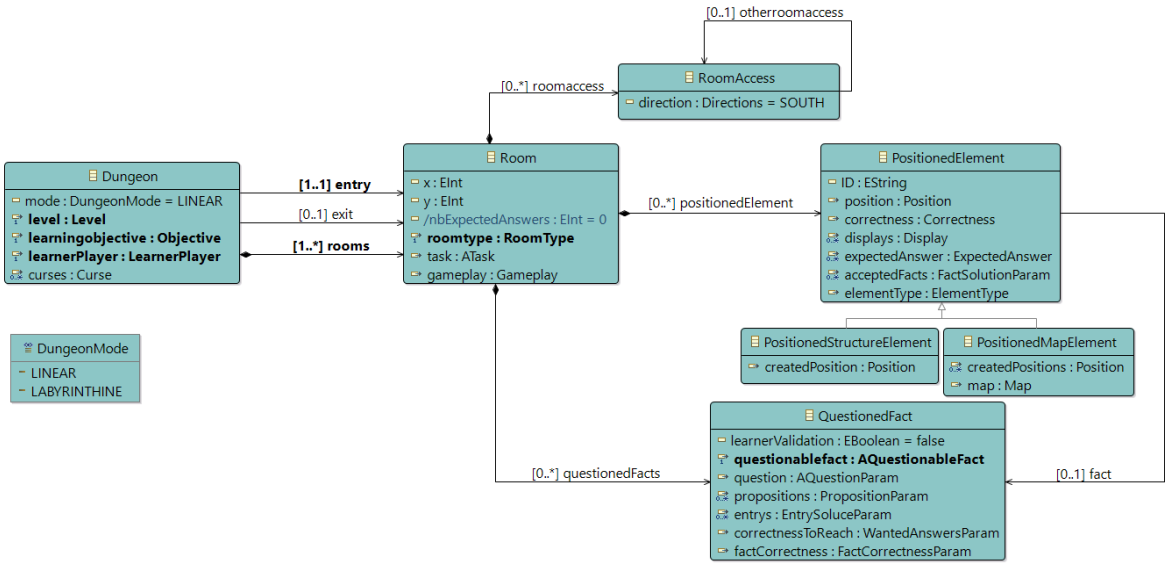


Figure 26 : Méta-modèle de l'activité

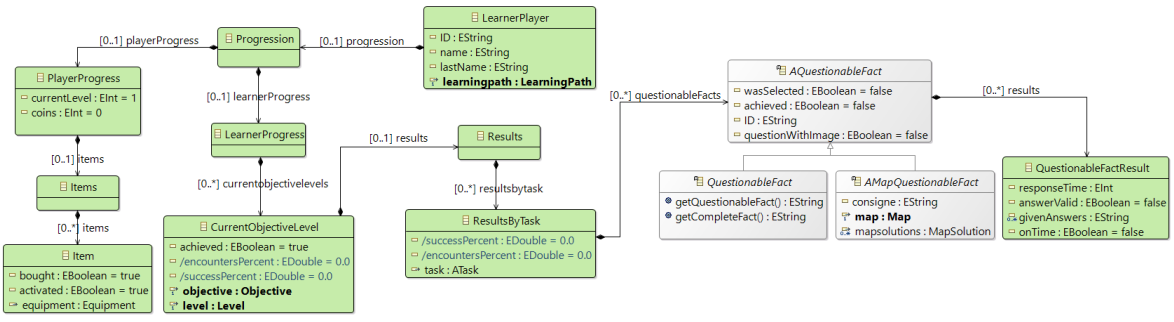


Figure 27 : Méta-modèle de l'apprenant-joueur

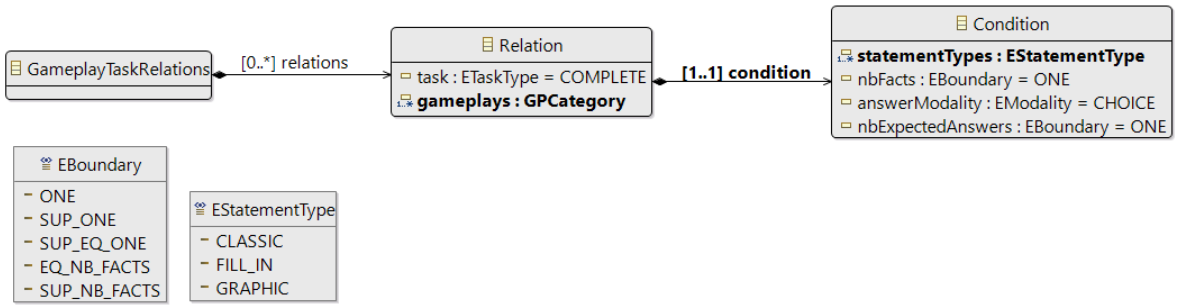


Figure 28 : Méta-modèle des relations