Une approche de modélisation pour générer des *gameplays* orientés tâches d'entrainement

Bérénice Lemoine[0000−0002−7608−3223] (3e année)

LIUM, Le Mans Université, 72085 Le Mans, Cedex 9, France
berenice.lemoine@univ-lemans.fr

Abstract. Cet article traite du problème de l'association des éléments éducatifs et ludiques dans le contexte de la conception et de la génération d'activités d'entrainement aux connaissances déclaratives. La proposition est une approche générique de modélisation des faits questionnés, ainsi qu'une modélisation des *gameplays* variables pour guider l'association des éléments au niveau algorithmique.

Keywords: Génération · Jeux Sérieux · Modélisation.

1 Introduction

Cet article concerne la génération d'activités de jeu (i.e., exploration de donjons) pour l'entraînement aux connaissances déclaratives (i.e., faits, lois, concepts). Ces connaissances nécessitent de la répétition pour leur rétention et leur généralisation [2]. Pour réduire l'ennui causé par la répétition [6], les activités de jeu doivent être variées. Dans notre contexte, l'*entrainement* aux connaissances déclaratives consiste à fournir aux apprenants-joueurs diverses formes de questions sur les faits, de manière répétée.

La conception d'activités de jeux d'entraînement nécessite d'associer les éléments d'entraînement et de jeu [5]. Le besoin de spécifier ces relations revêt une importance particulière dans le contexte de la génération d'activités de jeu puisque l'algorithme doit connaître les liens entre ces éléments afin de construire automatiquement des activités cohérentes. Différents travaux abordent la problématique de l'association des éléments ludiques et éducatifs. Certains identifient des relations entre concepts de haut-niveau permettant de guider la conception des jeux sérieux mais pas leur développement [5]. D'autres travaux guident la spécification de relations de haut-niveau (e.g., relations entre concepts tels que collaboration, orientation, exploration) pour l'analyse ou la conception de jeux [1]. À notre connaissance, aucune contribution ne propose de guider l'alignement du contenu éducatif et de jeu au niveau des éléments concrets.

Dans cet article, nous proposons une modélisation générique de faits questionnés (indépendante d'un domaine didactique), une modélisation de description de *gameplays* (i.e., éléments "fun" de jeu pouvant être contrôlés, décidés et accomplis par les joueurs [5]) variables, ainsi qu'un algorithme permettant la génération des *gameplays* concrets d'entrainement (éléments de jeu structurés décrivant une tâche d'entrainement à réaliser) à partir des faits à questionner.

Le travail synthétisé dans cet article est une partie des contributions des travaux de thèse. Cette dernière s'inscrit en recherche en ingénierie des EIAH [7]. Elle consiste à proposer une architecture logicielle générique de conception de générateurs d'activités de jeu pour l'entrainement aux connaissances déclaratives.

2 Contexte de recherche

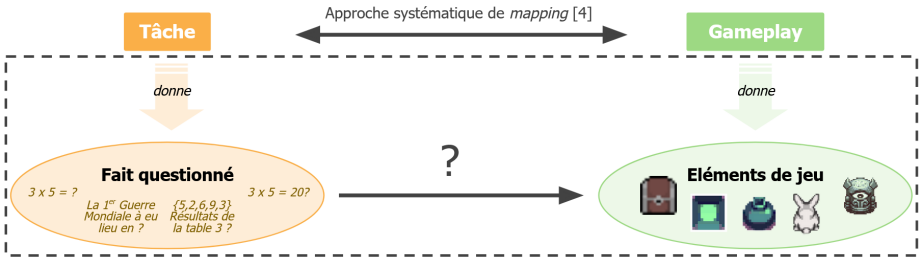

Fig. 1. Problème de recherche.

Dans des travaux précédents [4], nous avons défini quatre types de tâches d'entrainement pour les connaissances déclaratives : Completion (i.e., compléter un fait avec des éléments manquants, p. ex., 3 ? = 15) ; Ordonnancement (i.e., ordonner des faits en utilisant une heuristique, p. ex., du plus au moins proche du soleil : *fMars; Saturne; Terreg*) ; Identi cation (i.e., attester de la validité ou invalidité des faits, p. ex., 3 5 = 12 vrai ou faux ?) ; Identi - cation d'appartenance (i.e., identifier les éléments partageant une propriété donnée, p. ex., résultats table de 3 parmi : *f3; 5; 12; 4; 9g*). Dans notre contexte d'exploration de donjons, cinq catégories de *gameplays* possibles ont aussi été définies : *Select* (i.e., choisir des objets avec les réponses), *Move* (i.e., déplacer les objets corrects vers des zones), *Orient* (i.e., orienter les objets vers les réponses), *Position* (i.e., placer l'avatar pour répondre) et *Direct Response* (i.e., saisir les réponses). Pour construire une activité, l'algorithme doit d'abord être en capacité de choisir un *gameplay* (appartenant à une catégorie) compatible pour chaque tâche. Pour ce faire, nous avons proposé une méthode systématique fondée sur l'utilisation des formats de questionnaires numériques comme pivot [4]. À partir des tâches d'entrainement, des faits questionnés (i.e., questions sur les faits) sont générés. Ces faits doivent ensuite être implémentés en éléments de jeu permettant de les questionner. Ce problème de conception requiert de répondre aux questions suivantes, voir Figure 1 : 1) Comment modéliser les *gameplays* en termes d'éléments de jeu variables ? 2) Comment traduire des faits questionnés en *gameplays* les interrogeant ? La réponse à ces questions contribue à mieux appréhender l'alignement entre les éléments éducatifs et ludiques à un niveau algorithmique.

3 Modélisation des faits questionnés génériques

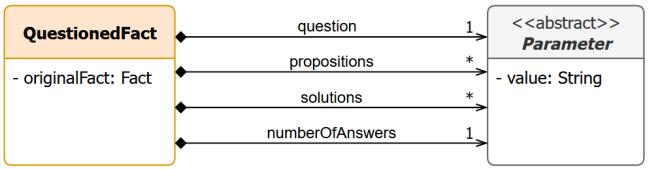

Fig. 2. Modélisation des faits questionnés [3].

Les faits questionnés ont tous des formes différentes en fonction de la tâche ou du domaine didactique. Cependant, les concepts qui les composent sont les mêmes. Notre idée est donc de considérer les faits questionnés comme un élément avec des paramètres (ces paramètres sont instanciés si nécessaires). Soit deux tâches T1 et T2 avec T1 consistant à compléter une multiplication dont le résultat est manquant par choix et T2 consistant à sélectionner les résultats d'une table. T1 donnera des questions telles que 2 6 = ? avec un ensemble de propositions $f8; 12; 14g$. T2 donnera des questions telles que "Donner les résultats de la table de 3" parmi $f3; 5; 7; 9; 12g$. Les faits questionnés ont les paramètres suivants : une question, un ensemble de propositions, un ensemble de solutions et le nombre de réponses attendues. La Figure 2 présente cette modélisation.

4 Modélisation de *gameplays* et éléments de jeu variables

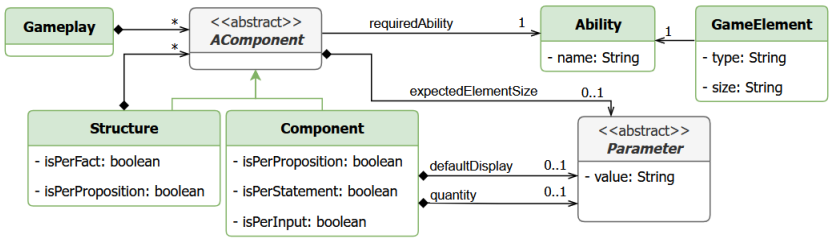

Fig. 3. Modélisation conceptuelle des *gameplays* et éléments de jeu [3].

La définition des *gameplays* en termes d'éléments de jeu spécifiques permet un certain niveau de variété, cependant elle est limitée car : 1) coûteuse en temps, c.-à-d., que les *gameplays* doivent être décrits un par un en fonction des éléments de jeu disponibles ; et 2) statique, c.-à-d., ajouter un élément de jeu impose de spécifier de nouveaux *gameplays* pour cet élément. Notre idée pour pallier ces

contraintes consiste à décrire les éléments en termes de capacites (*ability*, i.e., comportement des éléments), par exemple : un bloc peut être poussé (i.e., *pushable*), un pont peut être traversé (i.e., *crossable*), un pot et un cube peuvent être déplacés (i.e., *movable*), etc. Étant dans le contexte de l'entrainement aux connaissances déclaratives, chaque composant (i.e., *AComponent*) décrivant un *gameplay* possède une intention, c.-à-d., qu'il peut servir à afficher une proposition ou un ensemble de propositions, à afficher la question, etc. (e.g., *isPerProposition*, *isPerFact*). Ainsi, un *gameplay* est décrit par un ensemble de composants (i.e., simple ou composé, p. ex., un pot est un élément simple nommé *Component*, un bloc associé à un détecteur est un élément composé nommé *Structure*) ayant une capacité attendue et une intention. D'autres paramètres plus opérationnels, comme une taille ou une quantité d'éléments, peuvent être spécifiés. La Figure 3 présente cette modélisation.

5 Algorithme de génération

L'algorithme de génération est fondé sur les modèles décrits et consiste à associer les valeurs des paramètres instanciés des faits questionnés génériques aux éléments de jeu en fonction de leur intention. L'algorithme consiste pour un gameplay G et un fait F à parcourir les composants de G puis pour chaque composant[1], à construire des éléments positionnés en fonction des valeurs les paramètres instanciés de F et des propriétés du type de composant. Par exemple, pour un composant simple ayant *isPerProposition* à vrai, un élément positionné sera créé pour chaque élément de la liste des *propositions* de F.

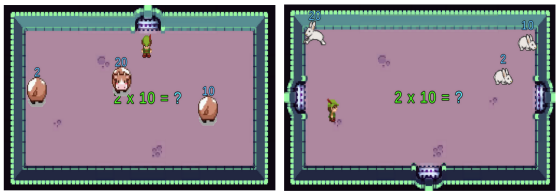

Fig. 4. Exemple de générations à partir du même *gameplay*.

Prenons un fait questionné "2 10 = ?" ($F1$) avec pour propositions=[2; 10; 20], et un *gameplay* ayant deux éléments, un de capacité *catchable* pour les propositions, l'autre de capacité *displayable* pour la question. Nous avons trois éléments de jeu, un lapin de capacité *catchable*, une vache de capacité *catchable* et un affichage de capacité *displayable*. L'algorithme va créer un élément de jeu de type affichage avec la valeur "2 10 = ?" et trois éléments, un par proposition, soit de type vache, soit de type lapin (cf. Figure 4). D'autre part, pour un même fait questionné, plusieurs *gameplays* peuvent être éligibles. Dans ce cas, le même

[1] Pour les *Structure*, un appel récursif est effectué sur l'ensemble de ses composants.