Comment représenter les connaissances dans un générateur semi-automatique d'exercices d'auto-évaluation ?

Baptiste Cablé, Marie Lefevre, and Nathalie Guin

Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
{baptiste.cable, marie.lefevre, nathalie.guin}@liris.cnrs.fr

Résumé Le générateur semi-automatique d'exercices interactifs d'auto-évaluation du projet CLAIRE permet à un auteur de créer un modèle d'exercices dont il maîtrise totalement le contenu. Ce modèle est ensuite instancié de manière automatique pour donner lieu à différentes versions de l'exercice correspondant toutes aux attentes de l'auteur. Chaque exercice dispose de connaissances permettant le diagnostic automatique de la réponse de l'apprenant. Les types d'exercices proposés (textes à trous, appariement, etc.) sont indépendants du domaine. Si l'auteur souhaite que ses exercices comportent des connaissances du domaine, il lui revient donc de les décrire dans le modèle d'exercices. Cet article discute de l'ensemble des connaissances manipulées dans le générateur d'exercices : les connaissances du domaine mais également les connaissances sur les types d'exercices et les connaissances liées au diagnostic de la réponse de l'apprenant. Pour chacune de ces familles de connaissances, nous étudions leur rôle, leur utilisation, s'il est intéressant de les expliciter, comment les représenter et comment faciliter leur création et leur usage par un auteur non-informaticien. Chacun de ces points amène à un ensemble de questions ouvertes.

Keywords: Génération semi-automatique d'exercices, connaissances du domaine, représentation des connaissances, ingénierie dynamique des connaissances, outil auteur.

1 Introduction

Le projet CLAIRE [1] a pour objectif la réalisation d'une plateforme d'édition sémantique et RichMedia [2] pour le monde de l'éducation. Une des facettes de ce projet est la réalisation d'un outil auteur pour la génération d'exercices d'auto-évaluation [1]. Le contexte d'utilisation des exercices est le suivant : à la fin d'un chapitre de cours en ligne, un exercice d'auto-évaluation est mis à disposition

1. Community Learning through Adaptive and Interactive multichannel Resources for Education. http ://projet-claire.fr

2. Un contenu RichMedia intègre au minimum un média qui a été enrichi (image, vidéo). Il est souvent interactif et peut intégrer un aspect temporel.

de l'apprenant qui travaille en autonomie. Il le résout et, le diagnostic étant automatique, il sait instantanément quelles sont ses erreurs. Il peut ainsi voir si l'essentiel des connaissances du chapitre est maîtrisé. Si nécessaire, il retourne travailler la partie de cours concernée, puis il peut à nouveau s'auto-évaluer. Il est important que l'exercice proposé lors de la seconde évaluation soit différent sinon il n'évaluerait plus la maîtrise des connaissances mais seulement la capacité à avoir retenu les réponses vues lors de la précédente résolution. Le générateur d'exercices de CLAIRE donne donc une nouvelle version de l'exercice à chaque fois qu'un apprenant veut s'évaluer.

Côté auteur, il n'est pas nécessaire de créer ces versions une à une. L'auteur crée un modèle d'exercices qui décrit l'exercice souhaité mais en laissant de la liberté pour obtenir des variantes lors de la génération. Afin de ne pas restreindre cet outil auteur à des domaines d'application précis, les types d'exercices sont indépendants du domaine (appariements, QCM, textes à trous, etc.). Seul le choix du contenu effectué par l'auteur oriente le modèle d'exercices vers un domaine.

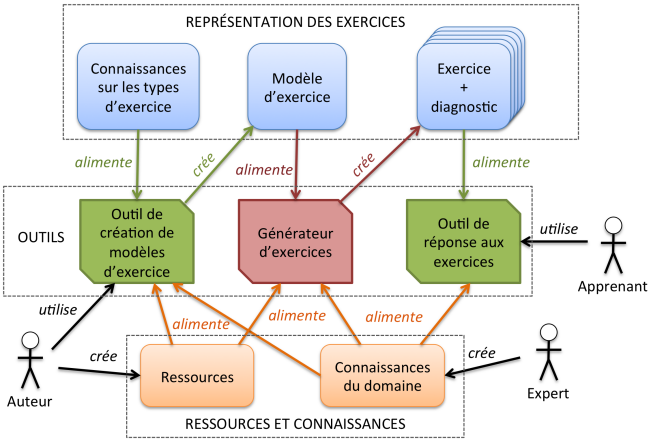

Figure 1. Architecture de l'outil auteur du projet CLAIRE pour la génération d'exercices d'auto-évaluation et de leur diagnostic [1].

Cet outil auteur est composé d'un générateur semi-automatique[3] d'exercices qui se base sur des modèles d'exercices créés par les enseignants selon leurs besoins [1]. L'architecture de cet outil est présentée sur la figure 1. Les *connaissances sur les types d'exercices* permettent à l'auteur de créer un mo-

3. Un générateur manuel demande à l'auteur de créer chaque instance d'exercice manuellement. Un générateur automatique utilise un modèle (auquel l'auteur ne peut pas accéder) pour générer un grand nombre d'exercices. À mi-chemin entre les deux, un générateur semi-automatique laisse beaucoup plus de liberté à l'auteur qui définit un modèle d'exercices qui est ensuite utilisé pour générer un grand nombre d'exercices.

dèle d'exercices au travers d'une interface que l'on appelle *outil de création de modèle d'exercices*. Le *modèle d'exercices* contient un ensemble d'informations et de contraintes suffisant pour créer différents exercices évaluant une même compétence. Cette création d'exercices à partir d'un modèle est effectuée par un *générateur d'exercices* sans intervention humaine. En d'autres mots, le générateur effectue une instanciation aléatoire du modèle d'exercices. L'instance d'*exercice* contient également les informations permettant le diagnostic de la réponse de l'apprenant. L'interface apprenant se charge de lui présenter l'exercice, de récupérer sa réponse et d'en afficher un diagnostic.

L'outil auteur s'appuie également sur des ressources et des connaissances du domaine représentées dans la partie inférieure du schéma. Les *ressources* sont des documents qui constituent la matière première permettant de construire les exercices. Il s'agit par exemple de textes, d'images et de questions prédéfinies. Un modèle d'exercices dispose de contraintes sur les ressources à utiliser pour construire l'exercice [4]. Les *connaissances du domaine* sont des connaissances indépendantes du type d'exercices et relatives à un domaine. Dans le domaine des mathématiques, il s'agit par exemple de la simplification d'écriture des racines carrées ou des valeurs remarquables des fonctions trigonométriques.

Notre outil auteur est indépendant du domaine et donc du sujet dont il est question dans l'exercice. Pour chaque nouveau domaine où un auteur souhaite l'utiliser, il doit implicitement décrire dans le modèle d'exercices les connaissances de ce domaine qu'il souhaite inclure dans l'exercice, et ce même si un autre auteur les a déjà décrites ailleurs. Il en va de même pour les connaissances relatives au diagnostic. Nous souhaitons faciliter la tâche d'un auteur lorsqu'il a besoin de connaissances souvent utilisées. Pour cela, nous voulons identifier et étudier toutes les connaissances manipulées au sein de l'outil de manière à déterminer lesquelles il serait intéressant de rendre accessibles aux utilisateurs (auteurs ou experts du domaine). Une fois ces connaissances identifiées, il faut déterminer comment les représenter, comment les créer et comment les manipuler de manière accessible à un utilisateur non informaticien. Cet article amène une discussion sur ces différents points.

Il s'organise de la manière suivante : nous recensons l'ensemble des connaissances manipulées dans notre outil auteur dans la section 2. La section 3 concerne la représentation des différentes connaissances manipulées dans notre outil. La section 4 amène des questions sur les méthodes de création de ces connaissances puis la section 5 en amène d'autres à propos de l'aide à l'utilisation de ces connaissances.

2 Typologie des connaissances pour la génération et le diagnostic d'exercices

Nous nous intéressons à l'ensemble des connaissances nécessaires pour la génération des exercices :

4. L'auteur peut lister ces ressources ou définir un ensemble de contraintes permettant de les retrouver lors de la création de l'exercice.

- les connaissances relatives aux types d'exercices ;
- les connaissances relatives aux ressources.
- les connaissances relatives au domaine ;
- les connaissances relatives au diagnostic ;

2.1 Connaissances relatives aux types d'exercices

La figure 1 représente l'architecture théorique de notre outil auteur mais dans la pratique, les connaissances sur les types d'exercices se trouvent à différents endroits, parfois contenues dans le code et parfois externes. Elles se retrouvent concrètement à 4 endroits :

- dans la brique *connaissances sur les types d'exercices* ;
- dans le générateur ;
- dans l'interface auteur ;
- dans l'interface apprenant.

La brique *connaissances sur les types d'exercices* de la figure 1 désigne uniquement la structure que doit respecter un modèle d'exercices. Il s'agit concrètement d'un XML Schema. Mais cela ne donne aucune indication sur la manière de générer un exercice (d'un type donné) à partir d'un modèle d'exercices (de ce type).

Les connaissances sur le mécanisme de génération sont intégrées dans le code du générateur. Elles sont définies par le concepteur et sont dépendantes de l'implémentation du générateur.

D'autres connaissances relatives aux types d'exercices se retrouvent au niveau de l'outil de création de modèles d'exercices. Dans cette interface auteur, la présentation et les fonctionnalités offertes varient selon le type d'exercices sur lequel l'auteur travaille. Par exemple, l'interface de création d'un texte à trous est différente de celle permettant de créer un exercice d'appariement. Ces connaissances pourraient être formulées séparément du code et l'outil de création de modèles adapterait sa forme en fonction du type d'exercices travaillé. Ainsi, pour créer un nouveau type d'exercices, un concepteur n'aurait pas à créer toute une interface mais seulement à créer ces connaissances sur « comment un auteur crée un exercice de ce type ». De plus, il semble que ces connaissances soient assez proches du contenu de la brique *connaissances sur le type d'exercices*. En étoffant légèrement ces dernières, on aurait des connaissances permettant de présenter une interface cohérente à l'auteur et de créer des modèles d'exercices. L'interface en elle-même serait donc générique et mise en forme selon les connaissances.

Des connaissances relatives aux types d'exercices sont également présentes dans l'*outil de réponse aux exercices*. Elles indiquent comment mettre en forme un exercice pour l'apprenant et comment présenter la correction de sa réponse. Il s'agit de connaissances très techniques auxquelles un auteur n'est pas supposé accéder. Comme pour le générateur, le concepteur doit créer ces connaissances pour chaque nouveau type d'exercices. À l'inverse de l'interface auteur où les

fonctionnalités sont assez récurrentes quel que soit le type d'exercices[5], il ne semble pas spécialement pertinent de chercher à créer une interface apprenant générique qui s'adapte en fonction des connaissances sur le type d'exercices. En effet, chaque type d'exercices dispose de fonctionnalités particulières et il est difficile de toutes les prévoir dans une interface générique.

En conclusion, les connaissances permettant de construire un nouveau type d'exercices seront augmentées au fur et à mesure de l'utilisation de l'outil et selon les besoins. Il faut donc les extraire du code et prévoir une interface auteur générique se basant sur ces connaissances. Nous pouvons pour cela nous inspirer de l'approche AKEPI [2] visant à faciliter l'acquisition des connaissances permettant de personnaliser un EIAH. Par contre, les connaissances procédurales définissant le mécanisme de génération et de correction peuvent rester dans le code car les diverses fonctionnalités, selon les types d'exercices, ne peuvent pas être prévues et elles requièrent dans tous les cas l'intervention d'un développeur.

2.2 Connaissances relatives aux ressources

Les ressources sont des documents et les connaissances qui les décrivent sont des métadonnées. Ces dernières apportent un complément d'information sur la ressource (par exemple le type d'une roche sur une ressource de type image), une information de synthèse sur la ressource (nombre de mots d'un texte) ou viennent tout simplement annoter la ressource pour faciliter sa recherche et sa réutilisation. La structure et l'utilisation des ressources et des connaissances les décrivant sont clairement définies dans notre outil. Nous ne les traitons donc pas dans cet article.

2.3 Connaissances relatives au domaine

Les types d'exercices initialement prévus dans CLAIRE sont indépendants du domaine et seuls le contenu et les contraintes précisés par l'auteur lient le modèle d'exercices à un domaine. À partir du type d'exercices *appariement*, l'auteur peut par exemple créer un appariement entre des fractions et leur forme simplifiée. Dans cet exemple, les connaissances du domaine sont des connaissances en algèbre et plus précisément, la simplification de fraction. Dans l'absolu, il y a deux façons de créer un tel modèle d'exercices. L'auteur peut créer manuellement les paires (ou faire appel à de telles ressources) entre une fraction non simplifiée et une fraction simplifiée. Il prévoit alors un grand nombre de paires et choisit de n'utiliser que quelques unes d'entre elles pour chaque instance d'exercice. Cette première possibilité est illustrée par un exemple en figure 2.

La deuxième manière pour créer le modèle d'un tel exercice est de faire appel à des connaissances du domaine des mathématiques. L'auteur utiliserait, par exemple, des connaissances permettant de générer des fractions simplifiées

5. Editer un modèle de QCM ou un modèle de texte à trous consiste toujours à éditer un modèle, c'est à dire une liste de ressources, des contraintes, etc. En clair, cela n'implique pas (ou peu) de fonctionnalités originales dépendantes du type d'exercices.

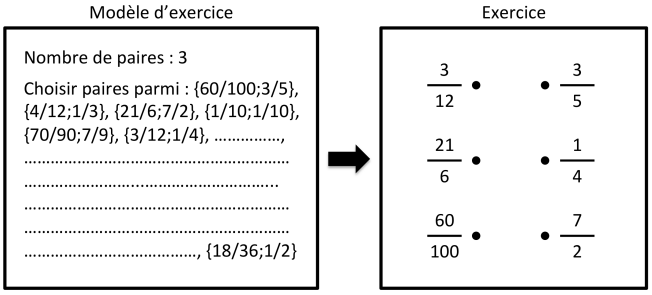

Figure 2. Exemple de modèle d'exercices sans connaissances du domaine.

(numérateur et dénominateur premiers entre eux) et de multiplier numérateurs et dénominateur de ces fractions par des entiers. Cela est illustré par l'exemple de la figure 3. Cet exemple fait aussi appel à la connaissance des nombres premiers.

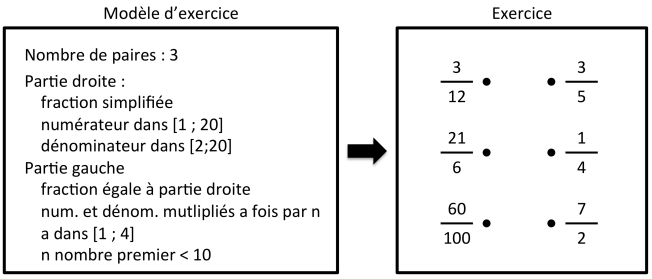

Figure 3. Exemple de modèle d'exercices avec connaissances du domaine.

Cette seconde approche nous semble plus judicieuse que la première pour plusieurs raisons. Premièrement, peu d'auteurs prendront le temps de créer manuellement un grand nombre de paires de fractions dont l'une est la version simplifiée de l'autre. Deuxièmement, même si ces paires existaient (elles peuvent avoir été créées par un autre auteur), cela rendrait le modèle d'exercices délicat à paramétrer car il s'appuierait sur un ensemble de ressources figées, et donc moins facile à faire correspondre aux préférences de l'auteur. Par exemple, comment s'assurer que les facteurs communs ne soient pas trop grands ? La seconde approche permet à l'auteur de paramétrer un modèle correspondant plus précisément à ses attentes pédagogiques. De plus, le nombre de variantes pouvant être générées de la sorte est bien plus important que lorsque l'on se limite à des ressources existantes. Enfin, l'approche à partir de connaissances du domaine offre cet avantage que les connaissances de mathématiques exploitées pour ce type d'exercices peuvent également être exploitées pour d'autres types d'exercices.

Les connaissances relatives au domaine auraient pu être mêlées au code de l'outil ou incluses dans les modèles d'exercices comme dans l'approche GEPPETOp [3] qui a inspiré nos travaux. Plus précisément, dans GEPPETOp, certaines connaissances sont codées en dur (connaissances mathématiques) et il est également possible de faire appel à des bases de connaissances externes comme un dictionnaire ou des connaissances de conjugaison mais cet appel aux connaissances externes est lui aussi défini en dur par le concepteur. Par contre, il n'est pas possible pour l'auteur de créer des connaissances du domaine et celles-ci sont fixées au départ.

Les domaines où le générateur de CLAIRE sera utilisé étant variés et non définis à l'avance, il est important que les connaissances du domaine puissent être créées en fonction des besoins. Il serait techniquement possible de ne pas employer de connaissances du domaine. Elles seraient alors implicitement disséminées au travers des différentes ressources et des modèles d'exercices, mais comme cela vient d'être expliqué, il s'agirait d'une perte de temps et de souplesse pour les auteurs lors de la création d'exercices.

En résumé, exprimer les connaissances du domaine de manière explicite permet une factorisation de connaissances utilisées par plusieurs auteurs dans un domaine donné. Cela leur permet d'une part de gagner du temps mais aussi d'aller vers des modèles d'exercices où le contenu est plus spécialisé tout en gardant un outil indépendant du domaine. Pour le moment, la création de ces connaissances est à la charge de l'expert du domaine et aucune décision n'a été arrêtée quant à leur forme et à leur utilisation.

2.4 Connaissances relatives au diagnostic

Notre outil auteur manipule également des connaissances relatives au diagnostic de la réponse de l'apprenant. Ce sont les connaissances qui permettent de juger si un apprenant a bien répondu ou non à l'exercice. Il s'agit, d'une part, de connaissances procédurales décrivant comment traiter et comparer la réponse de l'étudiant avec la solution de l'exercice. D'autre part, il s'agit de connaissances pouvant être renseignées par un auteur ou un expert du domaine telles qu'un complément d'information sur la correction ou les misconceptions (erreurs classiques) du domaine. Les premières décrivent le mécanisme pour exploiter les secondes.

2.5 Lien entre les différentes connaissances

Au moins une partie des connaissances de diagnostic est très liée au domaine : il s'agit des misconceptions. En effet, cette connaissance est utilisée pour le diagnostic mais peut aussi être employée comme connaissance du domaine pour générer les exercices. Par exemple, une misconception peut être utilisée pour créer une proposition trompeuse dans un QCM. Il conviendrait donc, tout en gardant une distinction entre connaissances relatives au diagnostic et connaissances du domaine, de choisir une méthode de représentation commune.

Le type d'exercices est, *a priori*, indépendant du domaine. C'est bien le cas pour les types d'exercices fournis avec l'outil auteur, mais cet outil est conçu de manière à ce que des développeurs puissent créer de nouveaux types d'exercices répondant à des besoins particuliers. Ainsi, certains types peuvent être intimement liés au domaine, comme par exemple « diagonalisation de matrice » qui est un type uniquement utilisable dans des contextes comprenant des matrices. Dans ce cas, il y a confusion entre les connaissances du domaine et les connaissances relatives aux types d'exercices car les secondes contiennent des connaissances orientées domaine. Doit-on donc d'abord formaliser ces connaissances sous forme de connaissances du domaine puis créer un nouveau type d'exercices exploitant ces connaissances du domaine ou doit-on directement créer un nouveau type d'exercices et y inscrire « en dur » les connaissances relatives au domaine ? La première solution offre l'avantage de ne pas dupliquer de connaissances et de maintenir une frontière claire entre les deux. La seconde permet au type d'exercices de fonctionner même si les connaissances du domaine venaient à être altérées, ou tout simplement dans le cas où elles n'existeraient pas car il serait trop difficile de les décrire avec le formalisme prévu pour les connaissances du domaine.

2.6 Notre besoin

Après l'étude des différentes connaissances mises en jeu dans l'outil auteur pour exercices du projet CLAIRE, nous constatons que les connaissances sur les types d'exercices se situent à trois niveaux : interface apprenant, générateur et interface auteur. Pour les deux premiers, il semble judicieux de laisser ces connaissances mêlées au code. Pour l'interface auteur, nous préfèrerions avoir une interface générique qui s'adapte au type d'exercices. Les connaissances sur le type d'exercices (pour la création de modèles) sont déjà explicitées et il ne reste qu'à les enrichir.

Les connaissances de diagnostic contiennent pour le moment uniquement des connaissances du domaine ou des connaissances relatives aux types d'exercices. Nous les traiterons donc, pour le moment, comme nous traitons les connaissances de ces deux catégories.

Les connaissances du domaine doivent être isolées du code, éditables et accessibles. Il faut donc trouver un moyen de les représenter et de les manipuler sans restreindre le spectre des domaines d'application. Les domaines peuvent en effet concerner toute matière scolaire (y compris post-bac) mais éventuellement d'autres champs de formation. Dans la suite de cet article, nous nous intéresserons à cette question : comment les représenter et faciliter leur acquisition pour un auteur non informaticien ?

3 Représenter les connaissances

3.1 Méthodes classiques en ingénierie des connaissances

La littérature en ingénierie des connaissances est vaste et nombre de méthodes sont destinées à la représentation des connaissances du domaine et aux

raisonnements permettant de les exploiter. Parmi celles que nous avons étudiées se trouvent les ontologies et les systèmes à base de règles.

Une ontologie (au sens informatique) est une description formelle des connaissances d'un domaine [4,5]. Elle en définit les concepts, relations et autres distinctions pertinents pour modéliser un domaine. Elle est très souvent définie par un ou plusieurs experts. L'aspect sémantique est un des intérêts principaux des ontologies mais il ne nous intéresse pas réellement dans notre approche où l'on souhaite, pour chaque domaine, ne modéliser que des fragments de connaissance pouvant même être indépendants les uns des autres. Ce qui nous intéresse est essentiellement le contenu des connaissances. Les ontologies ne nous semblent donc pas constituer une solution pour représenter les connaissances du domaine de notre outil auteur pour exercices.

Un système à base de règles représente classiquement les connaissances sous forme de faits et de règles basés sur la logique formelle [6]. Un moteur d'inférences permet de déclencher les règles pour déduire de nouveaux faits jusqu'à obtenir la réponse au problème. Cette solution pourrait être pertinente car on y retrouve à la fois des connaissances de type « fait » et des connaissances sur des mécanismes. Leur utilisation dans PERSUA2 [7] nous a permis de voir qu'un enseignant non informaticien était capable de définir des règles simples (si, alors, sinon). Par contre, le cœur d'un système à base de règles se situe dans les interactions entre faits et règles pour déduire un résultat. Dans notre approche, les connaissances de type « mécanisme » peuvent être indépendantes de toute autre connaissance du domaine et il n'y a donc pas réellement d'inférence. On perd donc une grande partie de l'intérêt des systèmes à base de règles.

Ces deux approches ne semblent donc pas totalement adaptées à notre contexte. Dans la section suivante, nous précisons le type de connaissances de domaine que nous souhaitons manipuler et représenter.

3.2 Pour les exercices du projet CLAIRE

Le générateur d'exercices de CLAIRE étant essentiellement destiné à l'enseignement post-bac, il est potentiellement supposé pouvoir couvrir l'enseignement de n'importe quelle discipline. Il est, bien entendu, impossible de prévoir un outil capable de représenter et manipuler n'importe quelle connaissance de n'importe quel domaine. Aussi, nous faisons des choix sur les types de connaissances que nous voulons manipuler. Ces choix traduisent un ensemble de fonctionnalités orientées édition d'exercices que nous souhaitons offrir aux auteurs.

Nous avons, pour le moment, identifié les types de connaissances du domaine suivants :

- **constante** : une constante du domaine. Exemple : pi = 3,141592654.
- **classe d'objets** : type de donnée pouvant être instancié et manipulé. Il est décrit avec un ensemble de fonctionnalités qui lui sont propres. Exemple : la fraction qui est composée de deux entiers et qui dispose de fonctionnalités telles que la simplification.
- **énumération** : une liste (ordonnée ou non) de mots, de valeurs ou d'objets. Exemples : les mots-clés du langage C, la liste des acides aminés.

- **tableau** : tableau à une ou plusieurs dimensions où des clés sont associées à des valeurs. Exemple : les valeurs remarquables des fonctions trigonométriques et les angles associés.
- **fonction** : fonction à la fois au sens mathématique et au sens informatique du terme. Elle prend zéro, une ou plusieurs valeurs (ou objets) en entrée et renvoie une ou plusieurs valeurs (ou objets) en sortie. Exemples : génération d'un nombre pair, test d'un entier pour savoir s'il est un nombre premier, étude des informations sur une tumeur pour savoir si elle est cancéreuse, détermination du temps d'un verbe, etc.

Ces différents types de connaissances permettent de faciliter la tâche de l'auteur du modèle d'exercices en mettant à sa disposition beaucoup de connaissances sans pour autant engager une grande complexité sémantique. La représentation des connaissances de type constante, énumération et tableau ne pose pas de réelle difficulté. L'aspect structure de données de la classe d'objets n'est pas très complexe en soi mais il faut prendre en compte son interopérabilité avec le reste des connaissances. Les fonctionnalités des objets ainsi que les connaissances de type fonction posent un problème beaucoup plus important au niveau de leur représentation. Il faut qu'elles permettent de décrire des mécanismes complexes mais il faut également qu'elles disposent d'une représentation accessible car un des objectifs futurs est de permettre à un auteur non informaticien de pouvoir éditer ces connaissances.

Pour permettre à ces auteurs de définir et de faire évoluer ces connaissances du domaine, il faut donc trouver une représentation des connaissances adéquate, et ceci soulève plusieurs questions. Par exemple, faut-il représenter de deux manières différentes les fonctions, selon qu'elles soient uniquement l'application d'une formule ou bien qu'elles contiennent un aspect algorithmique plus élaboré ? Faut-il privilégier une représentation sous forme de règles, sous forme de pseudo-code ou sous forme graphique ? Comment intégrer la gestion des types de données (classes d'objets) sans trop compliquer l'accessibilité à un utilisateur non expert ?

4 Faciliter la création des connaissances du domaine

La personne en charge de la création des connaissances est, pour le moment, un expert. Il est appelé expert dans le sens où il maîtrise le domaine mais surtout dans le sens où il dispose des compétences techniques pour la création de connaissances du domaine. Malheureusement, il est peu réaliste qu'une personne se consacre à cette tâche sans en bénéficier par la suite. En effet, les principaux bénéficiaires des connaissances du domaine sont les auteurs des modèles d'exercices et il semble donc plus pertinent que ce soit eux qui créent ces connaissances du domaine.

Le premier problème auquel est confronté un auteur non informaticien lors de la création de connaissances du domaine est la maîtrise technique de leur représentation. Un premier effort pour lui faciliter cette tâche de création est

donc de travailler sur la forme que devront prendre les connaissances pour être accessibles.

La deuxième difficulté n'est plus vraiment technique mais plutôt conceptuelle. Prenons un exemple. Si je suis un auteur d'exercices et que je m'aperçois que la connaissance du domaine dont j'aurais besoin (les enzymes de la digestion) n'existe pas, dois-je faire mon texte à trous en listant manuellement ces enzymes ou dois-je plutôt créer cette connaissance ? Dans le second cas, de quel type de connaissances s'agit-il ? Une énumération, des constantes ou un tableau ? Comment choisir la représentation et ce qu'il faut mettre dedans ? La solution la plus simple n'est-elle pas de tout lister dans mon modèle d'exercices ? Est-ce que si je définis une connaissance du domaine quelqu'un s'en resservira plus tard ?

Effectivement, un auteur peut se retrouver confronté aux questions de quand faut-il créer une connaissance du domaine, de quel type de connaissance utiliser et de quel contenu y mettre précisément. Nous envisageons d'apporter de l'assistance aux auteurs à deux niveaux. Premièrement, nous pourrions étudier les connaissances du domaine saisies implicitement par les auteurs dans leurs modèles d'exercices. Quand certaines connaissances reviennent de manière récurrente, nous pourrions proposer à l'auteur de formaliser automatiquement ces connaissances. Il n'aurait plus qu'à les nommer et à valider. Cela résoudrait en partie le problème du « quand ? ». Deuxièmement, nous pourrions assister l'auteur pendant qu'il formalise une connaissance : l'assister par des explications, par du remplissage automatique (appel aux autres connaissances), par de la correction de syntaxe (fonctions) et en vérifiant que le type de connaissances choisi soit cohérent.

Ces différentes assistances à l'auteur nous amèneraient à construire un outil assez complexe au cœur de notre outil auteur d'exercices : un outil auteur de connaissances du domaine.

5 Faciliter l'utilisation des connaissances du domaine

Même quand nous aurons choisi comment représenter les connaissances et comment faciliter leur création par les auteurs, plusieurs questions resteront en suspens concernant l'utilisation de ces connaissances.

L'objectif étant de faire gagner du temps à l'auteur, il faut que l'utilisation des connaissances du domaine existantes soit rapide et intuitive. Il faut qu'un auteur puisse facilement trouver les connaissances dont il a besoin et savoir si elles existent ou non. Il faut donc résoudre des problèmes d'ingénierie liés au partage des connaissances. Il faut également pouvoir s'assurer que la connaissance est fiable et répond au besoin de l'auteur. Une partie de solution consisterait à travailler sur la présentation de connaissances à l'auteur. Il faudrait pouvoir, pour chaque connaissance, en donner une représentation simplifiée agrémentée d'exemples d'utilisation.

La seconde partie de la solution pourrait consister à observer l'auteur durant son activité. Cela pourrait se faire en utilisant le paradigme du raisonnement à partir de l'expérience tracée [8]. Tout d'abord, lorsqu'un auteur semble s'orien-

ter vers un domaine dans son modèle d'exercices, cela permettrait de filtrer et mettre en avant automatiquement les connaissances qui sont susceptibles de l'intéresser au vu de ce qui a été précédemment fait par d'autres auteurs. Ensuite, nous pourrions chercher à identifier les connaissances qu'il est implicitement en train de créer au travers de ressources ou de modèles d'exercices et lui proposer d'utiliser des connaissances déjà formalisées à ce sujet.

Outre les problèmes d'utilisation des connaissances du domaine du point de vue de l'auteur, se posent des problèmes liés à leur utilisation au sens large. Comment s'assurer de l'utilisabilité des connaissances dans les différents types d'exercices ? Comment gérer l'interopérabilité et l'interdépendance des connaissances du domaine ?

Ces questions et celles des sections précédentes sont celles auxquelles nous aurons à répondre dans la suite de nos travaux sur les connaissances du domaine dans le cadre d'un générateur d'exercices d'auto-évaluation.

Références

1. Cablé, B., Guin, N., Lefevre, M. : Un outil auteur pour une génération semi-automatique d'exercices d'auto-évaluation. In : Environnements Informatiques pour l'Apprentissage Humain, Toulouse (2013)
2. Lefevre, M., Jean-Daubias, S., Guin, N. : Supporting Acquisition of Knowledge to Personalize Interactive Learning Environment through a Meta-Model. In : ICCE. (2009) 439–446
3. Lefevre, M. : GEPPETO : une approche générique permettant d'adapter les activités des apprenants aux intentions pédagogiques de chaque enseignant. In : RJC EIAH 2010. (2010) 33–38
4. McGuinness, D.L., Van Harmelen, F., et al. : Owl web ontology language overview. W3C recommendation **10**(2004-03) (2004) 10
5. Gruber, T.R. : Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum.-Comput. Stud. **43**(5-6) (December 1995) 907–928
6. Hayes-Roth, F. : Rule-based systems. Communications of the ACM **28**(9) (1985) 921–932
7. Lefevre, M., Guin, N., Jean-Daubias, S. : Personnaliser des activités pédagogiques de manière unifiée : une solution à la diversité des dispositifs. STICEF **19** (2012)
8. Champin, P.A., Cordier, A., Lavoué, E., Lefevre, M., Skaf-Molli, H. : User assistance for collaborative knowledge construction. In : Proceedings of the 21st international conference companion on World Wide Web. (2012) 1065–1074