Projet IDM
Modélisation, Vérification et Génération de Jeux

Omar NAIM
Lucien AUPETIT
Issam HABIBI
Badr SAJID

Département Sciences du Numérique - Deuxième année
2020-2021

Contents

1 Compte rendu

- Dans la première partie du projet nous avons proposé un métamodèle pour le jeu grâce à l'outil Xtext. Après avoir reçu le retour sur notre travail, durant cette deuxième partie nous avons amélioré nos métamodèles selon les consignes données: nous avons modifié certaines multiplicités, et enrichi nos classes. En effet, nous nous sommes fixé un métamodèle pour se répartir le travail, cependant plus nous avancions dans nos transformations ATL, plus nous avons trouvé que notre métamodèle était insuffisant pour pouvoir faire la transformation vers le réseau de Pétri, ensuite nous faisions divers améliorations pour pouvoir réussir une transformation ATL.

- Nous avons aussi écrit le programme correspondant au jeu d'Enigme avec le langage java. Ce programme présente une varieté de ce jeu qui se base sur un menu textuel intéractif avec l'utilisateur. Ensuite, nous avons défini une transformation modèle à texte avec l'outil Acceleo pour générer le code produisant le prototype qui correspond à un modèle de jeu . On a aussi défini les propriétés ltl à définir sur ce type de jeux.

2 Travail réalisé

2.1 Explication de la conception choisie

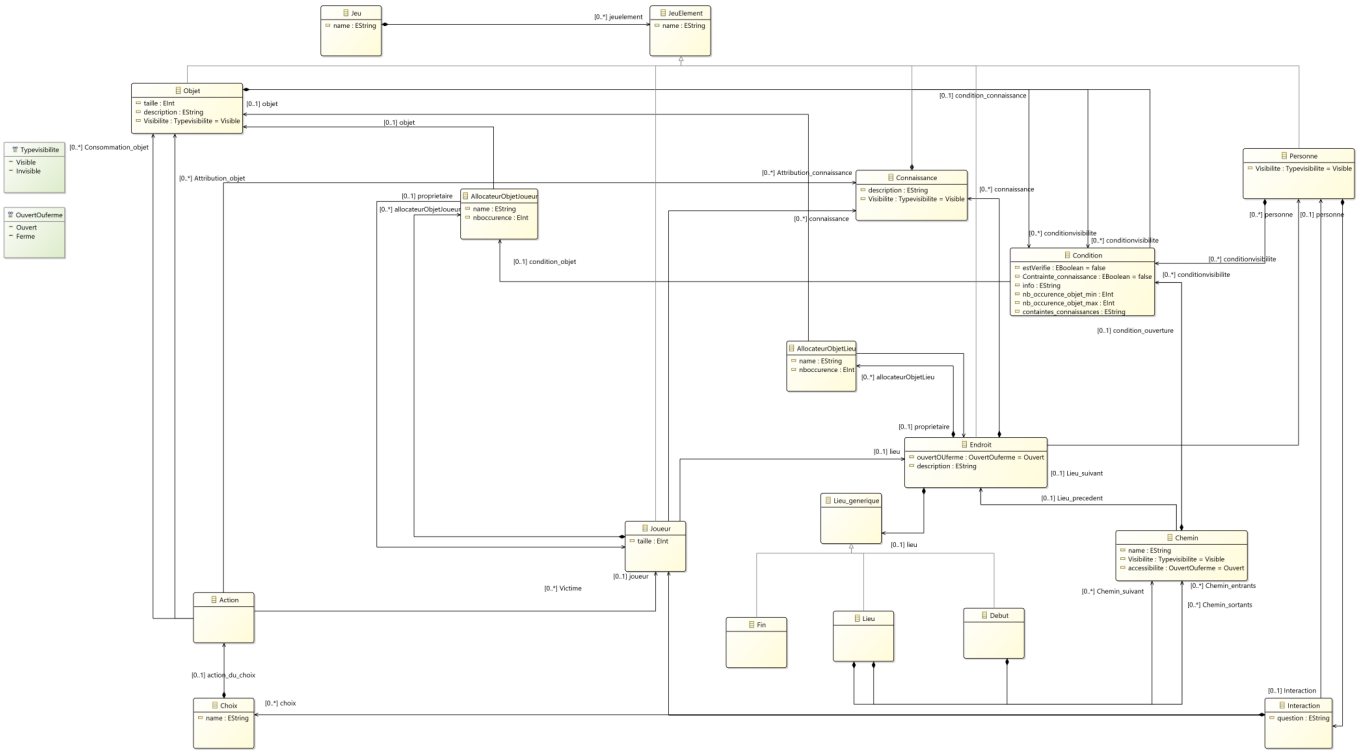

Figure 1: Métamodèle du jeu d'exploration

Pour modéliser le jeu d'exploration demandé, nous nous sommes un peu inspiré du SimplePDL, par suite nous avons fait recourt à des JeuElements, où nous avons représenté le joueur par une classe qui se trouve dans un endroit donné, qui a une taille d'inventaire fixe, et qui possède un ensemble de connaissances et d'objets qui sont visibles ou invisibles, ensuite un endroit par une classe qui possède un ensemble de connaissances et d'objets, de plus deux endroits differents sont reliés par des chemins. Et pour exprimer le fait qu'un explorateur ou qu'un endroit peut posséder plusieurs occurrences du même objet, nous avons définit des classes AllocateurObjetLieu et AllocateurObjetJoueur qui indiquent l'occurence que possède un endroit ou un lieu d'un objet donné. Ensuite les conditions sont des combinaisons logiques des connaissances et objets possédés par l'explorateur avec des multiplicités données, pour cela une condition possède des attributs allocateurs d'objets et connaissance qui permettent de régler cela. Ensuite pour gérer les interactions entre le joueur principale et les personnes non joueurs nous avons représenté cela par une classe interaction qui relie

le personnage non joueur et le joueur principal. Et finalement un choix par un nom et une action du choix qui s'applique sur une victime qui est le joueur principal et qui consomme une certain occurence d'objets ou de connaissances via l'action appliquée.

2.2 Contraintes OCL

Dans cette partie là, nous avons défini certaines contraintes que doit respecter notre modèle, comme le fait de l'existence et l'unicité du lieu de départ, que le joueur ne peut porter plus d'objets que peut supporter son inventaire, de plus de l'existence d'un point d'arrivée, de l'unicité du choix présenté lors d'une interaction et le fait que l'explorateur est l'unique joueur et qu'il se trouve seulement en un seul lieu à la fois.

2.3 Programme : Java & Transformation modèle à texte : Acceleo

```
**** DEBUT DU JEU ! ****


Vous êtes dans le lieu Énigme, Choisissez votre action
1- Afficher les détails sur les connaissances
2- Afficher les détails sur les objets obtenus
3- Afficher des détails sur le lieu courant
4- Interagir avec Sphinx
5- Prendre un objet
6- Déposer un objet
7- Prendre un chemin
0- Quitter

Indiquez votre choix : 1

Explorateur ne possède aucune connaissance !!

Vous êtes dans le lieu Énigme, Choisissez votre action
1- Afficher les détails sur les connaissances
2- Afficher les détails sur les objets obtenus
3- Afficher des détails sur le lieu courant
4- Interagir avec Sphinx
5- Prendre un objet
6- Déposer un objet
7- Prendre un chemin
0- Quitter

Indiquez votre choix : 5
0- Collecter l'objet Objet0
1- Collecter l'objet Objet1
2- Collecter l'objet Objet2
3- Collecter l'objet Objet3
4- Collecter l'objet Objet4
5- Ne rien collecter
Indiquez votre choix : 1
```

Figure 2: Le jeu en java

Nous avons décidé de programmer le jeu en langage de programmation Java. Le jeu se base sur un menu textuelle proposé au joueur, qui permet 7 possibilités: L'affichage des détails sur les connaissances que le joueur possède, l'affichage des détails sur les objets possédés par le joueur, l'affichage des détails sur le lieu courant, la possibilité de mener une interaction avec le personnage d'un lieu (Sphinx dans notre exemple concret), la collection d'un objet ou d'une conaissance et le changement du chemin.
Le joueur pourra le long du jeu afficher au fur et à mesure les informations proposés au menu. De plus, si le joueur décide de défier le Sphinx, une question avec choix multiple lui sera posée par le Sphinx, dont un seul choix est correct. Ainsi, si le joueur répond correctement à la question, il aura accès à la connaissance réussite et par suite débloquera la possibilité d'aller vers le lieu Succès. Sinon, il perdera une tentative parmis les 3 qu'il possède et ne pourra redéfier le Sphinx que s'il possède encore des tentatives, sinon il retournera collecter des objets et des connaissances. La fin du jeu est conditionnée par le fait d'arriver à Echec ou Succès.

En ce qui concerne la transformation modèle à texte via l'outil Acceleo, nous avons défini les règles générales qui permettent de générer le code Java pour n'importe quel jeu conforme à notre modèle proposé. Ceci nous a permis de générer le même code déja écrit précédemment pour le jeu d'Enigme.

2.4 Transformation modèle à modèle : ATL

Dans cette partie, nous avons défini les transformations vers le modèle PetriNet pour chaque élément du jeu. Ainsi nous pourrons vérifier par la suite que le jeu admet bien une solution qui menne à la dernière place : La fin. Le modèle généré est sous le metamodèle PetriNet mais une transformation modèle à texte nous donne le fichier transformé en Tina qui nous permet de générer un graphe avec places et transitions et de vérifier les propriétés LTL.

- Pour détailler un peu la map est déssinée à l'aide de place pour les lieux et chemins et une transition entre chaque endroit. Dans un lieu on a le choix entre plusieurs actions dont : intéragir avec une personne, déposer un objet ou le retirer etc. Toutes ces possibilités correspondent à des arcs qui partent de la place lieu.
- Le joueur est aussi une place liée par transition à un inventaire qui possède une taille maximale, représentée par son nombre de jetons. De plus, l'inventaire est lié avec chaque place qui correspond à l'objet possédé, par des transitions pour prendre en compte la taille des objets.
- Les connaissances étant illimitées, on a une transition du joueur vers une place qui correspond à la possession de la connaissance. En effet, chaque objet ou connaissance est soit visible ou invisible, pour cela, on rajoute donc une place de visibilite qui sera liée à la transition de récupération de la connaissance.
- Les conditions correspondent à des objets ou des connaissances que l'on est censé avoir ou non. On utilise pour cela des readarcs à partir des places "objet possédé" et "connaissance possédée" vers des transitions sur des actions.
- Et finalement, les actions sont soit une consommation d'objet soit une attribution d'objet/connaissance. Cela correspondra donc a un arc du choix fait vers une transition de récuperation de connaissance ou d'objet. Si c'est une consommation on ajoute un arc partant de la place de l'objet possédé vers une place de consommation puit.

2.5 Transformation modèle à texte : Génération des propriétés LTL

Dans notre modélisation, nous avons considéré qu'un jeu se termine que si l'on arrive à l'un des lieux finaux (par exemple Succès ou Echec dans le cas du jeu d'énigme). On a aussi défini d'autre propriétés : Le joueur ne peut pas être présent à 2 lieux différents au même moment, par exemple dans le jeu d'Enigme, on est soit dans le lieu Enigme, soit dans le lieu Echec ou bien dans Succès.
La vérification de ces propriétés appliquées à notre trasformation en réseau de petri du jeu d'énigme nous a donné des résultats correctes : le tout passe, sauf la dernière propriété qui dit que le jeu n'admet pas de fin.

3 Difficultés rencontrées

L'une des plus grande difficultés rencontrées fut la transformation du métamodèle du Jeu vers PetriNet, ce qui a induit à divers modifications sur le métamodèle qui demeurait à chaque fois insuffisant pour pouvoir réaliser cette transformation. Par suite, à des problèmes de répartition des tâches, vu qu'à chaque fois que quelqu'un avançait un peu plus sur sa partie, se retrouvait avec un nouveau métamodèle, ainsi à remodifier le travail déja réalisé. D'autant plus que la plupart entre nous avaient des problèmes d'Eclipse lors du déploiement des greffons.

4 Conclusion

Nous avons grâce à ce projet pu conceptualiser le modèle du jeu d'exploration et de le créer. Les outils à disposition nous ont permis de représenter le métamodèle sous différentes formes et de créer des transformations entre eux. De plus le fait de créer entièrement un métamodèle et seulement à partir de consignes était un challenge qu'on devait réussir et qu'on ne pouvait le faire qu'en travaillant en groupe vu qu'il fallait se mettre d'accord sur un seul et unique métamodèle, pour ensuite, nous répartir les tâches.