Janvier 2022

Rapport Projet IDM :
Modélisation, Vérification et Génération de Jeux

Rehel Briag, Riu Guillaume, Beji Wissem, Huc-Lhuillery Alexia

Table des matières

Introduction

Le projet a pour objectif de créer un métamodèle d'un plateau de jeu, qui peut être utilisé via différents outils et transformés dans d'autres modèles vus en cous d'Ingénierie Dirigée par les Modèles. Dans ce projet nous avons choisi de créer le fichier de base du métamodèle, la syntaxe textuelle XText, avec tous les membres du groupe, puis nous nous sommes répartis chacun des fichiers à implanter pour que chacun travaille seul ou à deux sur son implantation.

Syntaxe textuelle Xtext

Le fichier Game.xtext définit une syntaxe concrète textuelle pour les modèles de jeu (extension.game). La vue graphique du métamodèle engendré correspond au fichier game.png.

Exemple enigme

Le fichier enigme.game est un modèle de l'exemple sur l'énigme du Sphinx donné dans le sujet utilisant la syntaxe textuelle définie précédemment. Ce fichier a été transformé en un réseau de Pétri (format Tina) qui donne le déroulement du jeu, que l'on retrouve dans enigme.net. Le fichier enigme.ltl décrit l'existence d'une solution pour le jeu ; à la fin du jeu, soit on se trouve dans la place succes, soit dans la place echec. Le fichier enigme.java exécute le jeu de l'exemple.

Contraintes OCL

Le fichier game.ocl contient les contraintes OCL associées au métamodèle game.ecore :

- Il doit y avoir un moins un lieu de fin (inv auMoinsUneFin)
- La taille cumulée des objets possédés par le joueur ne peut pas dépasser une taille limite (inv respectCapacite)
- Les quantités des objets sont positives (inv quantitePositive)
- Le nombre d'exemplaires d'un objet possédé par le joueur est positif (inv nbPossPositif)
- Les choix proposés dans une interaction sont différents (inv choixDifferent)
- Les actions proposées dans un choix sont différentes (inv actionDifferentes)
- Au plus une personne obligatoire par lieu (inv unePersonneObligatoire)

Le fichier Jeu-ok.xmi est un modèle respectant les contraintes OCL et Jeu-ko.xmi est un modèle ne les respectant pas (plusieurs joueurs, taille limite dépassée, plus d'une personne obligatoire dans un lieu, plus d'un chemin obligatoire depuis un lieu).

Éditeur graphique

L'éditeur graphique contient deux vues que l'on peut afficher ou non, et une vue qui est la vue générale tout le temps visible.

Sur la vue générale on peut observer les différents objets et connaissances contenus dans le jeu. Les liens avec les autres éléments apparaissent lorsque la bonne vue est ajoutée à celle de base. Ces deux éléments généraux au jeu peuvent être créer depuis la vue dans les outils Element. La vue générale possède aussi les outils permettant de modifier les conditions sur éléments du modèle, dans le dossier outils Condition. On peut créer une disjonction dans les éléments qui le permettent, et des conditions sur un objet ou une connaissance qui doit être possédé ou présent selon l'éléments contenant la disjonction. Pour créer ces conditions, il faut sélectionner la disjonction puis l'objet ou la connaissance que l'on souhaite ajouter à celle-ci.

Sur cette vue générale, on peut sélectionner la vue graphique des territoires en sélectionnant le calque Territoires (fig. 1). Chaque territoire est représenté pas une boîte contenant les personnes, objets, et connaissances présentes dans le territoire. Ces éléments peuvent être modifiés par les outils contenus dans le dossier outils Lieux, pour les objets et connaissances il faut sélectionner le

lieu puis l'objet ou la connaissance de la vue générale pour les ajouter au lieu. Les chemins entre les territoires sont représentés par des flèches allant du territoire précédent vers le suivant. Les conditions sur ces chemins sont contenues dans les boîtes intitulées 'Condition(s) chemin [Nom du chemin]', contenant les conditions sur la visibilité et l'ouverture d'un chemin. Les objets et connaissances transmises et les objets consommés sont représentés par une flèche pondérée pour représenter le nombre d'objet que l'on souhaite transmettre ou consommer, allant du chemin vers l'objet ou la connaissance, avec un code couleur : rouge clair pour les objets transmis, rouge foncé pour les objets consommés, et bleu pour les connaissances transmises. Ces éléments peuvent être modifiés et ajouter depuis les outils de l'onglet Chemins, en sélectionnant le chemin puis l'objet ou la connaissance comme précédemment pour les ajouter au contenu du chemin.

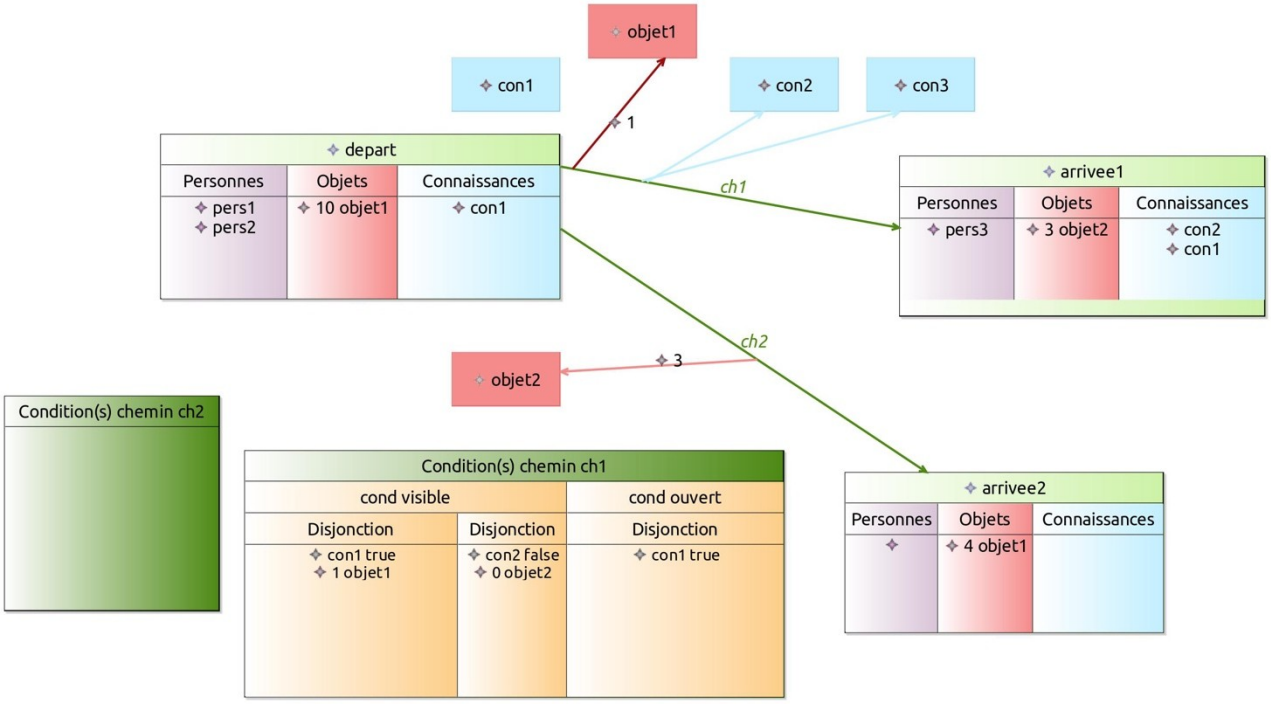

Figure 1 : Vue Territoires

Depuis la vue générale, on peut ensuite ajouter le calque Interactions pour passer à la vue graphique des interactions du jeu (fig. 2). Chaque lieu est représenté par une boite contenant les personnes présentes, et chacune de ces personnes est aussi représentée par une boîte, dans laquelle on retrouve les choix possibles et une condition sur la visibilité de la personne dans le lieu. Les choix sont composés de plusieurs actions ainsi que de l'action choisie par le joueur. Chaque action contient des conditions sur les possessions du joueur et sur la transmission de ses éléments, des objets et connaissances transmises et le choix précédent. Tous ces éléments peuvent être modifiés par les outils contenus dans le dossier outils Personnes. Les éléments impliquant un autre élément du modèle nécessitent de le sélectionner lors de sa création (objet transmis, connaissances transmise, action choisie, choix précédents).

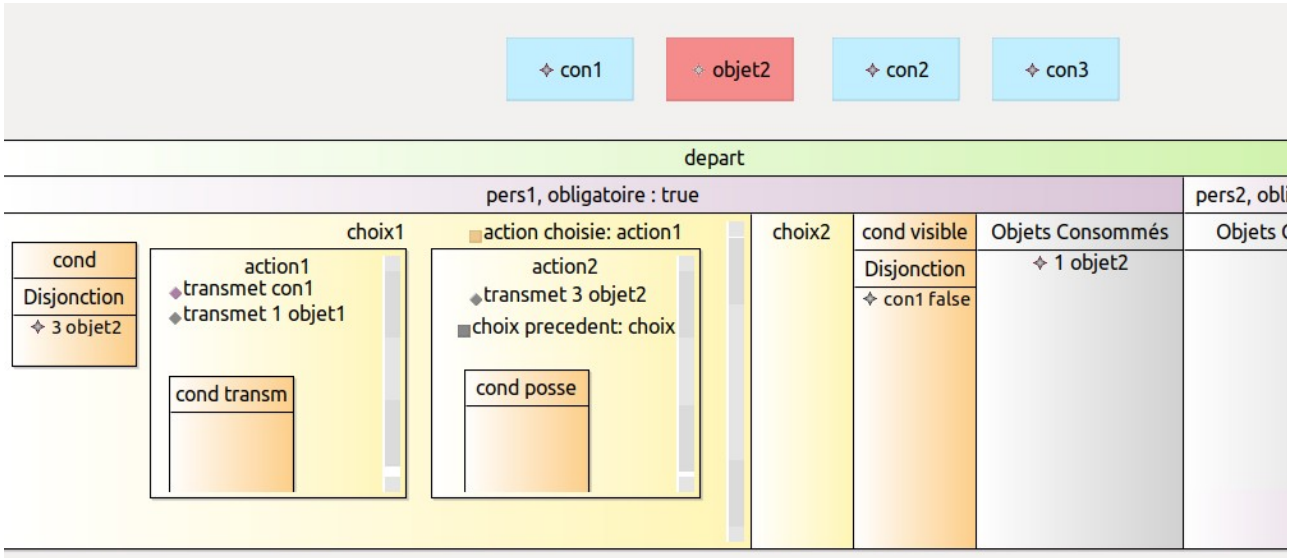

Figure 2 : Vue Interactions

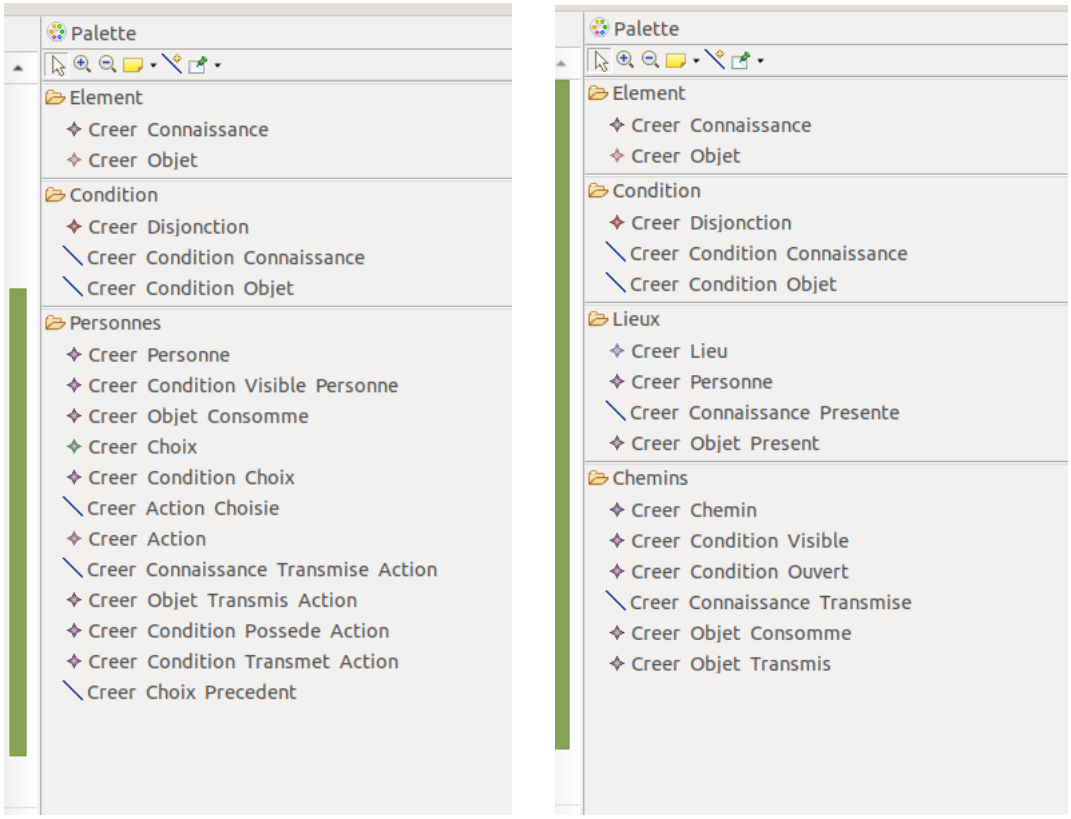

Figure 3 : palettes des outils, sur la vue Interactions et la vue Territoires

Transformation ATL GameToPetrinet

La transformation ATL (fichier game2petrinet.atl) repose sur les points clés suivants :

- Chaque objet et connaissance du jeu sera modélisé par une place dont le nombre de jetons représente combien possède le joueur de cette ressource : avoir un jeton dans une place d'objet ou connaissance signifie que le joueur possède un exemplaire, (ainsi on peut avoir 0, 1, 2, ... comme nombre de jetons pour les places des objets et 0, 1 pour les places des ressources). La transformation d'un objet à un autre en position du joueur est modélisé par deux arcs et une transition entre les deux objets en question : activer ce passage transporte les jetons qui traduisent ce que possède le joueur d'un objet à l'autre.

- Modélisation des conditions : la modélisation précédente des objets et connaissances est très utile pour pouvoir modéliser les conditions : ainsi à chaque fois qu'on a besoin d'une condition pour faire passer des jetons dans le petrinet d'une place initiale à une place finale, on lie la transition entre ces deux places avec des Read-arcs qui proviennent des places des objets et des connaissances qui constituent la condition en question, comme ça on garantit de n'avoir le passage que si le joueur possède ce qui est demandé par la condition.

- Pour chaque lieu on crée des sortes de stations (des places) qui représentent ce que fait le joueur en passant par le lieu, on commence par une place qui représente la récupération des connaissances présentes dans le lieu en question, la transition lié à cette place a des arcs vers les places des connaissances déjà crées existants dans le lieu, la place suivante concerne les objets mais a un traitement différent vu que le joueur peut choisir de prendre un objet ou pas, ainsi on crée pour chaque objet présent dans le lieu une transition qui lie la place des objets et la place de l'objet en question créée initialement, ainsi le joueur peut choisir de passer les jetons exactement aux objets qu'ils veut récupérer. Après on passe vers la station (place) de la personne obligatoire, cette place sera liée au traitement de la personne obligatoire si elle existe (le traitement sera détaillé après), après cette station on trouve la station (place) des personnes non obligatoires qui bien évidement liée aux traitements des personnes non obligatoires, on finit notre tour de lieu dans une place finale qui représente la fin de présence du joueur dans le lieu en question.
Remarque : pour le lieu départ on met un jeton dans la première place qui représente le joueur

- Un traitement de personne crée une place pour chaque choix possible et fait de même pour les actions de chaque choix en appliquant la méthode des conditions déjà expliqué au-dessus à chaque fois qu'on a besoin d'une condition.

- Pour les chemins : Un traitement de test d'obligation pour les chemins similaire à celui des personnes est mis en place avec l'ajout des : on crée deux places une pour confirmation de visibilité et l'autre pour la confirmation de l'ouverture en appliquant le traitement de condition pour chacune. Le résultat de traitement d'un chemin lie la dernière station (place) du lieu prédécesseur et la première station (place) du lieu successeur.

Transformation modèle à texte vers les propriétés LTL

Le ficher game2ltl.mtl contient le code Acceleo de la transformation vers les propriétés LTL. La terminaison d'un jeu est vérifiée par le fait que l'on se trouve dans un des lieux de fin.