# Détection automatique des interactions entre apprenants dans les jeux sérieux multi-joueurs dédiés à l'apprentissage

Mathieu Guinebert

Sorbonne Université

EDITE

*LIP6 / MOCAH*

# Détection automatique des interactions entre apprenants dans des jeux sérieux multi-joueurs dédiés à l'apprentissage

*Approche basée sur l'analyse des scénarios*

Par Mathieu Guinebert

Thèse de doctorat d'informatique

Dirigée par Vanda Luengo

Présentée et soutenue publiquement le 20/09/2019

Devant un jury composé de :

| | |
|---|---|
| Amal El-Fallah Segrouchni | Professeure |
| Elise Lavoué | Maître de Conférences HDR |
| Pierre Lagarrigue | Professeur |
| Jean Charles Marty | Maître de Conférences HDR, Rapporteur |
| Sébastien George | Professeur, Rapporteur |
| Amel Yessad | Maître de Conférences, Encadrante de la thèse |
| Mathieu Muratet | Maître de Conférences, Encadrant de la thèse |
| Vanda Luengo | Professeure, Directrice de la thèse |

# Résumé

Cette thèse, encadrée par Mathieu Muratet et Amel Yessad, est dirigée par Vanda Luengo. L'objectif de ce travail est de permettre la détection automatique des interactions entre pairs pouvant émerger d'un scénario de jeux sérieux multi-joueurs en amont de toute utilisation de celui-ci.

Les interactions entre pairs participent à la motivation et à l'engagement des apprenants dans leur processus d'apprentissage. Le travail engagé dans cette thèse propose des modèles et des outils d'analyse pour permettre aux concepteurs d'un jeu d'obtenir des informations sur les interactions entre pairs pouvant émerger au sein du jeu et ce sans requérir aux traces des joueurs. Les concepteurs pourraient alors s'appuyer sur ces informations pour modifier leur scénario afin qu'il corresponde au mieux à leurs attentes en terme d'interactions. Afin de satisfaire cet objectif nous avons apporté trois contributions principales.

La première contribution est une ontologie grâce à laquelle il devient possible de modéliser des scénarios de jeux sérieux multi-joueurs avec différents niveaux de granularité. Les interactions ayant souvent des définitions abstraites, la deuxième contribution vise à aider leur formalisation à l'aide de propriétés de bas niveau. Les interactions ainsi formalisées deviennent détectables automatiquement. La troisième contribution est un ensemble d'algorithmes d'analyse du scénario modélisé pour détecter les différentes interactions pouvant émerger de celui-ci.

L'ontologie a été testée sur différents scénarios de jeux sérieux. Les deux autres contributions sont mises à l'épreuve à travers une expérimentation menée sur un jeu créé dans le cadre de cette thèse.

# Abstract

This thesis is supervised by Mathieu Muratet, Amel Yessad and Vanda Luengo (thesis director). The goal of this work is to allow the automatic detection of peer interactions that could emerge from a multi-player learning game scenario before any use of it.

The peer interactions contribute to the motivation and involvement of learners in their learning process. The work undertaken in this thesis propose models and analysis tools to allow game designers to obtain information on the peer interactions that could emerge from their game without requiring players' traces. Thus, the designers could rely on that information to modify their scenarios to match with their needs towards peer interactions. In order to fulfill this goal, we brought three main contributions.

The first contribution is an ontology thanks to which it becomes possible to model multi-player learning games scenarios with various granularity levels. The interactions are often abstractly defined; the second contribution aims to help their formalization thanks to low-level features. Interactions formalized in a such a way become automatically detectable. The third contribution is a set of algorithm dedicated to the analysis of the modeled scenario in order to detect the various interactions that could emerge from it.

The ontology has been tested on various serious games scenarios. The two other contributions have been put to the test through an experimentation carried out on a game created in the scope of this thesis.

# Remerciements

Cette thèse n'aurait jamais été possible sans le soutien indéfectible de mes encadrants et je tiens donc à les remercier en premier. Amel Yessad, Mathieu Muratet et Vanda Luengo ont été un soutien constant. Leurs nombreuses relectures et conseils ont été indispensables à la réalisation de ce travail. Il nous est arrivé à de nombreuses reprises d'être en désaccord sur certaines pistes à suivre, mais jamais ils ne m'ont forcé dans une voie qui ne me convenait pas. Je suis particulièrement têtu et pourtant, ils ont su canaliser mes efforts. La reconnaissance que j'ai envers eux est immense.

J'aimerais également remercier Béatrice Bérard, Nathalie Sznajder ainsi que l'ensemble des enseignants de l'UE 2i005 sur les structures discrètes, pour m'avoir permis de mener à bien mon expérimentation auprès de leurs étudiants. Elles m'ont apporté une aide très précieuse quant à la conception d'un jeu sérieux adapté aux besoins de leur UE. Ils ont tous été extrêmement compréhensifs quant à la démarche qui était la mienne et ce en dépit des perturbations que mon expérimentation a pu entrainer dans leurs cours. Je remercie également les étudiants ayant participé à cette expérimentation et sans qui rien n'aurait été possible.

Je tiens à remercier l'ensemble des membres de l'équipe MOCAH pour l'aide et l'amitié qu'ils ont pu m'apporter. Je remercie Odette Auzende, nos nombreuses discussions et trajets me manqueront. Ton soutien lors des derniers mois m'a été indispensable. Je remercie Hélène Giroire, Monique Baron et Jean-Marc Labat qui m'ont fourni de très bons conseils, en particulier en matière de présentation. Je remercie Thibault Carron en particulier pour l'aide qu'il m'a fourni avec Learning Adventure. Je remercie François Bouchet qui m'a conseillé à de nombreuses reprises.

J'aimerai remercier tout particulièrement les doctorants et post-doctorants qui ont pu faire un bout de chemin avec moi. Je pense à Fatima Harrak qui a toujours été présente durant ces quatre années. Nous nous sommes serrés les coudes de bout en bout. Merci infiniment pour toute l'aide que tu as pu me fournir ainsi que pour nos nombreux échanges. Je pense à Guy Mbatchou et Mathieu Vermeulen qui m'ont soutenu dès qu'ils en avaient l'occasion et dont le soutien organisationnel et moral m'a été indispensable. Je pense à Nathalie Labrousse qui m'a fournie une aide précieuse pour ClassCraft. Je pense enfin à Gorgoumack Sambe, Bruno Martin, Iryna Nikolayeva, Yannick Bourrier, Alexis Lebis, Olivier Brunet, Ouarda Dridi et Camila Morais Canellas. Ils m'ont tous apporté un soutien moral et organisationnel pour lesquels je les remercie chaleureusement.

Enfin je tiens bien évidemment à remercier ma famille. Mes parents tout d'abord, qui ont été d'un soutien sans faille, qui m'ont été à surmonter bon nombre d'épreuves y compris (surtout) dans les moments de doute. Je ne pourrais jamais assez les en remercier. Je tiens également à remercier mon frère et ma sœur, qui, s'ils n'ont pas toujours compris ma démarche, m'ont toujours soutenu. Ils ont su supporter mes coups de mou, les nuits blanches, les absences et autres interruptions dû à mon travail de thèse. Pour finir, je tiens tout particulièrement à remercier ma fiancée. Elle a su tout supporter : ma mauvaise humeur, mon travail, mon

épuisement, … Elle n’a jamais cessé de me soutenir et a fait preuve d’efforts particulièrement important tout au long de ces quatre ans. Il ne fait aucun doute à mes yeux que si j’en suis là aujourd’hui c’est grâce à elle. Merci, Lucie.

# Table des matières

# Table des illustrations

## Table des Figures

# Table des Tableaux

# Section 1
# Introduction et Etat de l'art

Le chapitre I de cette section vise d'abord à présenter le contexte dans lequel s'inscrivent nos travaux de recherche et les différentes questions auxquelles nous voulons répondre. Nous présentons dans le chapitre II, différents éléments que nous avons pu trouver dans la littérature en lien avec nos questions de recherche.

# I. Introduction

La production annuelle de jeux sérieux est en augmentation ces dernières années. Nous pouvons par exemple lire dans l'article publié sur le site de *l'allied marked research*[1] que : « … *the global serious games market was valued at $2,731 million in 2016, and is projected to reach $9,167 million by 2023, growing at a CAGR of 19.2% from 2017 to 2023. »,* ce qui signifie une croissance prévue de 235% en 6 ans (soit une moyenne de 22% par an). Ce développement amène donc à l'émergence d'une grande diversité de jeux sérieux traitant de sujet très variés (nous y reviendrons en détail dans la section II.1). Parmi les grandes familles de jeu sérieux, certains sont dédiés à l'apprentissage et peuvent, selon (Paraskeva, Mysirlaki, & Papagianni, 2010) et (Wouters, Van Nimwegen, Van Oostendorp, & Van Der Spek, 2013), améliorer l'engagement de l'apprenant.

Aujourd'hui l'omniprésence d'Internet, des réseaux Informatiques, des ordinateurs personnels et des smartphones, permet d'envisager le passage des jeux sérieux mono-joueur aux jeux sérieux multi-joueurs jouables en réseaux locaux ou à grande échelle via Internet.

D'ailleurs, les jeux sérieux multi-joueurs (JSMJ) suscitent un intérêt grandissant chez les enseignants en leur permettant de mettre en place des scénarios où les apprenants peuvent interagir les uns avec les autres (Sourmelis, Ioannou, & Zaphiris, 2017; Turkay, Hoffman, Kinzer, Chantes, & Vicari, 2014). Plusieurs travaux de recherche ont déjà montré l'intérêt des interactions entre pairs dans les JSMJ et leur impact sur la motivation et l'engagement des apprenants (Bell & Kozlowski, 2007; Eseryel, Law, Ifenthaler, Ge, & Miller, 2013).

Dans ce contexte très favorable à l'introduction des JSMJ auprès des enseignants, nous partons de deux constats qui sont les raisons d'être de cette thèse : d'une part, les définitions des interactions proposées dans la littérature, telles que la collaboration, la coopération ou le conflit manquent de consensus. D'autre part, ces interactions sont souvent analysées en s'appuyant sur les traces laissées par les apprenants à travers des expérimentations dont la mise en place est souvent complexe et coûteuse. Or, nous pensons que les interactions sont fortement dépendantes des éléments de *game design* (ressources, actions et déroulement du jeu) et pourraient donc être anticipés en s'appuyant sur ces éléments.

L'objectif à plus long terme de ce travail est d'assister les enseignants et / ou les concepteurs de JSMJ, en fournissant automatiquement des informations sur les interactions susceptibles d'émerger de leurs scénarios avant même qu'ils aient été joués par des apprenants. Cette analyse automatique des scénarios JSMJ permettrait aux enseignants et / ou aux concepteurs de modifier / corriger leurs scénarios si les interactions ne répondent pas à leurs besoins. Nous avons ainsi fait le choix de proposer une approche d'analyse qui s'appuie uniquement sur le scénario avant même qu'il soit joué par des apprenants. Cette approche est donc une approche alternative ou complémentaire aux approches d'analyse qui s'appuient sur les traces des apprenants.

[1] https://www.alliedmarketresearch.com/press-release/serious-games-market.html

Comme nous le verrons dans l'état de l'art (chapitre II), il n'existe pas de méthodes ou d'outils permettant de détecter automatiquement ou même semi-automatiquement les interactions multi-joueurs pouvant émerger d'un scénario JSMJ complexe et non linéaire. La problématique de ce travail de thèse est donc de proposer des modèles et des algorithmes permettant la détection semi-automatique d'interactions multi-joueurs dans des scénarios de jeux sérieux dédiés à l'apprentissage. Plusieurs questions de recherche découlent de cette problématique :

1. Quel modèle de scénario adopter pour permettre la description puis la détection d'interactions entre pairs ? (QR1) Cette question concerne la modélisation formelle d'un scénario JSMJ dans le but d'analyser des interactions entre pairs.
2. Comment pouvons-nous modéliser formellement les interactions entre pairs ? (QR2)
3. Est-il possible d'analyser de manière fiable un scénario de jeu a priori dans le but de prévoir les interactions multi-joueurs qui seront effectuées par les apprenants ? (QR3)

Au vu de ces questions de recherche et de nos contributions, nous avons décidé d'articuler cette thèse en plusieurs chapitres. Le chapitre II, l'état de l'art, vise à présenter un état des lieux des recherches en lien avec la détection des interactions de groupe dans des environnements informatique pour l'apprentissage (EIAH). Il se découpe en quatre parties :

1. Etat de l'art sur les jeux sérieux et leurs définitions
2. Etat de l'art sur les interactions, leurs formalisations et détections
3. Etat de l'art sur la scénarisation de Jeux sérieux multi-joueurs
4. Etat de l'art sur l'analyse de scénario

Ensuite nous présentons nos trois principales contributions. D'abord, nous avons défini un modèle formel de description de scénarios et d'activités de JSMJ. Ce modèle est opérationnalisé par une ontologie (MPLGO) dont les concepts sont instanciés pour décrire des scénarios et des activités de JSMJ et met l'accent sur l'aspect multi-joueurs à travers la notion de rôles prototypiques (cette première contribution est présentée dans le chapitre III).

Notre hypothèse de travail est qu'il est possible d'analyser les interactions pouvant émerger d'un scénario en se basant sur sa description et en particulier sur ses éléments de *game design*. Cette hypothèse s'explique par le constat suivant : les ressources disponibles et les enchainement d'actions imposés aux joueurs par le jeu peuvent les empêcher ou les amener à progresser dans le scénario d'une certaine manière et donc peuvent favoriser ou empêcher l'émergence d'une interaction ou d'une autre. Ces règles et ces ressources font partie du *game design* du jeu. Ainsi, s'appuyer sur des éléments de description du scénario d'un JSMJ, intégrant les ressources du jeu et ses règles, pourrait aider à la détection des interactions entre pairs. Deux exemples pour comprendre cela serait celui de la collaboration et de la compétition. Pour la collaboration par exemple, si les joueurs se retrouvent dans l'incapacité d'avancer sans mettre en commun leurs ressources, ils vont être amenés à collaborer. Quant à la compétition, si la quantité de ressources disponibles n'est pas suffisante pour que tous les joueurs puissent finir le scénario, ils se retrouveront en compétition pour la possession de ces ressources.

La deuxième contribution (voir chapitre IV) est un *framework* de définition de propriétés d'interactions. Le *framework* proposé permet de définir une interaction à partir de ces propriétés. Grâce à la flexibilité apportée par notre *framework,* il est possible de redéfinir les interactions pour correspondre à la vision souhaitée, ainsi que d'ajouter des définitions d'interactions que nous pouvons juger manquantes. Ces propriétés sont extraites de la modélisation des activités et sont ensuite cumulées à l'échelle du scénario.

Notre troisième contribution (voir chapitre V) a été de mettre en place des algorithmes capables d'analyser un scénario à partir de sa description formelle. En se reposant sur cette description, nos algorithmes cherchent à établir les différentes séquences d'activités que peuvent effectuer les apprenants pour finir le jeu. Le calcul de ces séquences repose à la fois sur la modélisation des interactions présentées dans le chapitre IV, et sur la modélisation des scénarios du chapitre III. Une fois ces séquences calculées, il devient possible de déterminer les interactions que peuvent effectuer les apprenants en analysant les interactions pouvant émerger de nos séquences d'activités.

En résumé, nos trois contributions permettent de :

- Formaliser et modéliser un scénario de JSMJ décomposable en activités multi-joueurs
- Établir un sous ensemble de séquences d'activités d'un scénario JSMJ. Ce sous ensemble devra porter l'essentiel des informations sur les interactions pouvant émerger dans un scénario
- Extraire des propriétés qui permettent d'exprimer des interactions plus abstraites pouvant émerger des activités du JSMJ

Par la suite, nous mettons à l'épreuve le modèle du chapitre III en nous appuyant sur les différents concepts des chapitre IV et V. Pour ce faire, nous avons modélisé quatre jeux sérieux : *Voracy Fish* (Interactive, 2012), *Byte Battle* (Muratet, Delozanne, Torguet, & Viallet, 2012), *Learning Adventure* (Marty & Carron, 2011) *e*t *ClassCraft* (Sanchez, Young, & Jouneau-Sion, 2017). Les modélisations résultantes de cette mise à l'épreuve sont présentées dans le chapitre VI.

Nous avons mené une expérimentation pour évaluer nos contributions. Cette expérimentation, ainsi que l'ensemble de nos résultats et évaluations, sont détaillées dans le chapitre VII.

Les différentes conclusions que nous avons pu tirer de nos travaux sont présentées dans le chapitre VIII. Nous revenons dans ce chapitre sur les trois contributions de ce travail de thèse et nous présentons un certain nombre de perspectives.

Dans le cadre de ce travail, les termes apprenant, joueur ou apprenant-joueur sont utilisés de manière interchangeable et désignent le protagoniste d'un jeu sérieux. En outre, dans la suite du manuscrit, nous utiliserons le terme interaction pour désigner une interaction entre pairs. Les interactions que peuvent avoir les apprenants avec le système seront désignées par IHM (Interactions Humain-Machine).

# II. État de l'art

Ce travail de recherche a pour problématique de définir des modèles et des algorithmes pour détecter automatiquement des interactions entre pairs à partir des éléments de *game design* intégrés dans la description d'un scénario de JSMJ, en amont de son exécution par des joueurs.

Ce chapitre présente les différentes revues de la littérature effectuées afin de traiter cette problématique. Dans le chapitre I, nous avons évoqué les trois questions de recherche que nous nous sommes posées pour y répondre. L'état de l'art sur la question « Comment pouvons-nous modéliser formellement les interactions entre pairs ? » est présenté dans la section II.2. Celui sur la question « Quel modèle de scénario adopter pour permettre la description puis la détection d'interactions entre pairs ? » se trouve, quant à lui, dans la section II.3. La section II.4 rapporte les différents travaux étudiés pour répondre à la question de recherche « Est-il possible d'analyser de manière fiable un scénario de jeu a priori dans le but de prévoir les interactions multi-joueurs effectuées par les apprenants ? ». À ces trois sections sur les questions de recherche évoquées en introduction, nous ajoutons également une section II.1 visant à préciser ce que nous définissons comme un jeu sérieux.

## II.1. Jeux sérieux

Certains chercheurs tel que Julian Alvarez (Alvarez, 2007) datent l'apparition des *serious games* modernes à 2002 avec la création de *America's Army* (Zyda, 2005).

La définition adoptée par Alvarez dans sa thèse (page 4) pour définir un jeu sérieux est la suivante : « *Nous entendons par serious game, une application informatique, dont l'intention initiale est de combiner, avec cohérence, à la fois des aspects sérieux (Serious) tels, de manière non exhaustive et non exclusive, l'enseignement, l'apprentissage, la communication, ou encore l'information, avec des ressorts ludiques issus du jeu vidéo (Game).* ». Cette définition a été étendue en 2007 (Alvarez, Rampnoux, Jessel, & Methel, 2007) en établissant différentes classifications de celui-ci :

- L'*edutainment* dont le but est la transmission de connaissances à l'aide de ressorts ludiques.
- L'*advergaming*, un jeu dont la visée est de faire la publicité d'un produit.
- L'edumarket, qui cherche à véhiculer un message (comme par exemple la sensibilisation à la sécurité routière)
- Les *political games* cherchant à transmettre un message politique avant tout.
- Les simulations et *training games* regroupant des jeux allant de *SimCity* au simulateur d'avion.
- Les jeux sérieux se basant sur la collecte de données (par exemple les *games with a purpose ou GWAP*)

Alvarez et Djaouti ont proposé en 2011 une nouvelle classification des jeux sérieux en se basant sur trois critères : **le *gameplay***, **la visée du jeu** et sa **portée**.

**Le *gameplay*** en jeu vidéo fait généralement référence au type de jeu auquel le joueur se retrouve confronté. Parmi les *gameplay* fréquents nous pouvons par exemple citer les *First Person Shooter* (FPS) caractérisés par le fait que le joueur tire sur ses adversaires en contrôlant un personnage en vue à la première personne ou encore les jeux de stratégie en temps réel, ou les jeux de plateforme. Ces classifications de *gameplay*, assez courante pour les jeux vidéo, manquent cependant de rigueur. Le travail de (Portugal, 2006) auquel (Djaouti, Alvarez, & Jessel, 2011) font référence fournit une classification du *gameplay* plus rigoureuse se basant sur 5 composants : les règles, les méthodes d'entrée (manette, clavier, etc.), les réglages relatifs à l'espace, les réglages relatifs au temps, les réglages relatifs au « drama ».

**La visée du jeu** fait référence à l'intention autre que ludique désirée par le concepteur lors de la création du jeu. (Djaouti et al., 2011) propose une classification en trois grandes catégories, cette classification étant une extension de celle de 2007, les deux dernières catégories y apparaissent également :

- Les jeux à messages qu'ils divisent en plusieurs types : message éducatif (*Edugames*), informatifs (*NewsGame*s), persuasifs (*Advergames*), et enfin subjectifs (*Military/Art games*)
- Les jeux d'entraînement physique et / ou cognitifs.
- Les jeux de collecte de données cherchant à récolter des informations sur un sujet ou sur l'utilisateur (ex : *Foldit*)

Pour finir, le critère de **la portée** sert à catégoriser les jeux en fonction de l'audience et du marché pour lesquels ils sont prévus. (Djaouti et al., 2011) proposent de classifier l'audience sous un format proche de la norme *Pan European Game Information* (PEGI) qui regroupe les jeux en fonction de la tranche d'âge requise pour jouer au jeu. Quant au marché, ils proposent les secteurs suivants : Etat & gouvernement, Militaire et défense, Santé, Education, *Corporate*, *Religious*, Culture, Ecologie, Politique, Humanitaire, Publicité, Recherches scientifiques.

Il existe des définitions beaucoup plus restrictives des jeux sérieux. Ces définitions se basent alors principalement sur les jeux éducatifs. Celle de (Fabricatore, 2000) sur les jeux éducatifs en est un exemple : « […] *a virtual environment and a gaming experience in which the contents that we want to teach can be naturally embedded with some contextual relevance in terms of the game-playing.*[…] ». Nous constatons alors que la définition d'un jeu éducatif est très proche de la définition d'un jeu sérieux qui serait dédié à l'apprentissage (*Edugames*).

Ces deux visions sont divergentes par leur manière de considérer ou non l'intention des concepteurs. La première considère que tout jeu ayant été développé avec une intention autre que l'amusement ou produisant des effets sérieux comme l'apprentissage incident sont des jeux sérieux. La deuxième en revanche indique que la visée doit avoir été voulue, réfléchie, planifiée et pensée pour favoriser l'apprentissage du joueur.

Cette thèse a pour contexte applicatif les jeux sérieux qui permettent de travailler des connaissances ou des compétences relatives à un domaine donné. Ces jeux se caractérisent par la nécessité de prendre en compte l'acquisition et le travail des connaissances chez les joueurs dans le cadre de leur formalisation et de leur modélisation.

Les mondes ouverts, les jeux non-linéaires et autres formes émergentes de *gameplay*, offrent de plus en plus de liberté d'action et de choix aux joueurs. Des scénarios de jeux sérieux multi-joueurs peuvent s'appuyer sur ces éléments pour offrir une expérience ludique plus intéressante aux joueurs. Les problématiques liées à l'exploitation et à la représentation de tels scénario doivent donc être prises en compte.

Le *game design* (Salen, Tekinbaş, & Zimmerman, 2004) définit l'ensemble des éléments et règles constituant le jeu. À l'aide d'un *game design* adéquat, il est possible de contraindre les joueurs à effectuer des actions données pour finir le jeu. Un jeu peut par exemple, obliger ses joueurs à partager des ressources et connaissances pour pouvoir progresser dans le jeu. Un jeu possédant ce type de *game design* favoriserait l'apparition de comportements collaboratifs et coopératifs chez ses joueurs, ceux-ci devant travailler de concert pour avancer. L'exemple de (Wendel, Gutjahr, Göbel, & Steinmetz, 2013) propose aux joueurs de mettre en commun des ressources et des compétences pour pouvoir échapper de l'île dont ils sont prisonniers. Le design du jeu présenté est alors de nature collaborative.

Cette thèse ne traitera pas de jeux dépourvus de buts. En effet, les objectifs de ceux-ci ne peuvent pas être formalisés à l'avance. Un exemple connu serait les jeux qualifiés de jeux « bac à sable » dans lesquels les joueurs évoluent indéfiniment dans un monde où ils peuvent manipuler l'environnement avec des objectifs propres. Minecraft en est l'un des exemples les plus connus. D'autres genres de jeux peuvent être dépourvus de buts comme nous l'indique (Mortara et al., 2014) avec notamment des exemples de jeux sous la forme de tourisme virtuel.

De nombreux jeux sérieux mono-joueurs existent (*Refraction, Blockly*, …) et sont donc dépourvus d'interactions multi-joueurs. Le travail de cette thèse portant sur les interactions entre joueurs, nous ne nous intéresserons pas non plus à ce type de jeux. Nous nous intéressons exclusivement aux jeux sérieux multi-joueurs dédiés à l'apprentissage que nous désignerons dans la suite par JSMJ.

# II.2. Interactions

## II.2.a. Définition d'une interaction

(Moore, 1989) différencie trois grands types d'interactions, les interactions avec le contenu, les interactions avec l'enseignant et les interactions entre apprenants. Cette distinction est récurrente et chacun de ces types possède un intérêt. Cependant, comme précisé dans l'introduction, cette thèse porte exclusivement sur le troisième type d'interactions à savoir les interactions entre apprenants. Ces interactions sont définies comme un ensemble d'actions et de réactions réciproques entre deux ou plusieurs apprenants. Dans ce travail, le terme interaction sera exclusivement utilisé pour désigner les interactions entre pairs médiées par les jeux sérieux multi-joueurs.

Une interaction sociale est bien souvent construite au cours d'un dialogue en situation d'apprentissage mais est dépendante du contexte. La retranscription manuelle de dialogue étant une pratique courante (on peut trouver quelques exemples de ces retranscriptions dans (Roschelle & Teasley, 1995) ou encore (Edwards & Westgate, 2005)), il n'est pas étonnant de

trouver des travaux comme ceux de (Kumpulainen & Mutanen, 1999) qui cherchent à labelliser manuellement des interactions ayant lieu au cours d'un dialogue. Pour ce faire, ces derniers fournissent un très grand nombre de labels (plus d'une vingtaine) pouvant être combinés pour un seul et même échange. Cette diversité des labels montre la complexité d'une interaction. Le fait que ces méthodes de labellisation soient encore manuelles de nos jours, démontre également la difficulté d'expliciter des règles permettant une labellisation automatique.

La variété et la complexité des interactions ne sont cependant pas les seules problématiques qu'il est possible de rencontrer. En effet, Il est toujours difficile d'établir une définition formelle faisant consensus lorsqu'il s'agit d'un concept social. Pourtant cette formalisation est indispensable pour toute tentative d'automatisation de la détection des interactions.

Pour illustrer le problème que pose cette formalisation, concentrons-nous tout d'abord sur trois types d'interactions qui semblent pertinentes dans notre contexte : la collaboration, la coopération et la compétition. L'étude de l'état de l'art concernant ces trois interactions nous permettra de cerner quels critères sont à prendre en compte pour modéliser celles-ci.

## II.2.b. Collaboration

La collaboration et la coopération sont souvent confondues (Roberts, 2004), comme en témoignent les travaux visant à lever cette ambiguïté (Dillenbourg, Baker, Blaye, & O'Malley, 1995; Panitz, 1999). Cette confusion sémantique pourrait être expliquée par le fait que chaque discipline semble avoir sa propre définition des deux concepts. Par exemple (Kreijns, Kirschner, & Jochems, 2003) prennent le parti de s'abstenir de toute différenciation entre les deux justifiant ce choix par leur similarité dans le contexte de l'apprentissage collaboratif médié par ordinateur.

Dans le cadre des EIAH, deux définitions sont particulièrement citées. Celle de (Roschelle & Teasley, 1995) où ils décrivent la collaboration comme *une activité coordonnée et synchrone qui est le résultat d'une tentative continue de construire et de maintenir une vision commune du problème.* (Dillenbourg, 1999) définit la collaboration comme l'exercice d'une activité par un groupe doté d'une « hiérarchie horizontale instable » et ce avec un même et unique but. Autrement dit, la collaboration en EIAH est définie comme une activité où les apprenants sont au même niveau hiérarchique (hiérarchie horizontale) et sont régulièrement amenés à échanger leurs sous-tâches respectives (instable). Ces deux définitions issues du domaine des EIAH font émerger les éléments caractéristiques suivants pour la collaboration :

- Les individus agissent dans un but commun
- Ils peuvent librement échanger leurs rôles
- Tous les membres du groupe doivent avoir conscience de faire une activité ensemble
- Construction commune de connaissances et de compétences
- Ils construisent une vision commune du problème

Dans le domaine de la théorie des jeux, Zagal et ses collègues (Zagal, Rick, & Hsi, 2006) stipulent que dans un jeu collaboratif, tous les participants travaillent ensemble en tant qu'équipe, se partageant les coûts et les récompenses. Ils indiquent que si l'équipe est défaite,

tout le monde est défait, et inversement. Ils définissent par ailleurs une équipe comme une organisation dans laquelle les informations que chacun possède peuvent différer mais où les intérêts et croyances demeurent les mêmes. Selon cette définition, la collaboration existe surtout par la nécessité qu'ont les membres d'une équipe (même les individualistes) de mettre en commun leurs effort pour l'emporter. Ici les joueurs sont d'abord motivés par le succès. Nous remarquons que si la définition selon la théorie des jeux est bien trop restrictive par rapport à la nature complexe des jeux sérieux qui imbriquent des éléments de jeu et des éléments d'apprentissage humain, elle met néanmoins en lumière la notion de victoire et de défaite commune (une forme de gain), relativement absente des définitions généralement employées dans le cadre des EIAH classiques.

Nous avons établi de ces définitions de la collaboration, plusieurs problématiques auxquelles nous devons répondre pour permettre une détection automatique de celle-ci. Premièrement, en fonction de notre interlocuteur, nous pouvons établir des formalisations très différentes. Dans le cadre de la théorie des jeux, l'acquisition de connaissances ne joue aucun rôle dans la collaboration. De fait, pour un interlocuteur issu de la théorie des jeux, la collaboration doit être détectée y compris dans les contextes où les apprenants impliqués n'acquièrent pas les mêmes connaissances au contraire d'un interlocuteur issu des EIAH. En revanche, les définitions issues des EIAH comportent des critères difficilement formalisables du point de vue des actions des joueurs. La construction de la connaissance est un exemple d'un tel critère. Ces 2 problématiques que sont la divergence de définition entre les domaines et la difficulté de formalisation de certains critères liés à la collaboration rend d'autant plus difficile sa formalisation en vue d'une détection automatique.

### II.2.c. Coopération

La coopération est plus difficile à définir tant elle semble proche de la collaboration. Pourtant si nous nous référons encore une fois aux travaux de (Dillenbourg, 1999) et de (Zagal et al., 2006), tous deux donnent une définition de la coopération dans leurs domaines respectifs et la distinguent de la collaboration. Ainsi, en opposition à la « hiérarchie horizontale instable » de la collaboration, Dillenbourg énonce une « hiérarchie verticale stable » pour la coopération. Cette hiérarchie implique une structure où chacun peut avoir un supérieur (hiérarchie verticale) et où chaque individu résout une sous-tâche dont il ne dévie jamais (stable) et s'inscrivant dans un projet global plus important dont tous les individus ont conscience. Ici les connaissances acquises par les individus peuvent être différentes en fonction des rôles.

Dans la définition de la coopération de (Zagal et al., 2006), les joueurs possèdent des objectifs ne rentrant pas en conflit les uns avec les autres. Ils effectuent des actions entraînant chacune une forme de récompense d'abord pour eux-mêmes et ensuite pour les autres. Une illustration souvent citée de cette dernière en théorie des jeux, est le cas où deux prisonniers décident de se taire dans le dilemme du prisonnier (Rapoport, Chammah, & Orwant, 1965).

Dans le cadre de la théorie des jeux, la coopération se définit donc comme une mise en commun des efforts et des ressources pour réaliser des objectifs différents. Cette définition

ne nécessite pas nécessairement une organisation et une répartition claire des tâches au contraire des définitions issues des EIAH.

En résumé, pour les deux domaines, la coopération relève d'une coordination des intérêts des individus impliqués. Ceux-ci divergent en revanche sur la nécessité pour les individus d'organiser leurs efforts respectifs. Les problématiques soulevées restant les mêmes que celles détectées pour la collaboration (section II.2.b).

## II.2.d. Compétition

La problématique soulevée par la compétition est quant à elle tout autre. Cette dernière est peu traitée dans le domaine des EIAH pour des raisons qui relèvent de la psychologie (Brightman, 2006; Johnson & Johnson, 1988). La compétition y est vue comme stressante et démoralisante donc néfaste pour les apprenants à l'inverse de la collaboration ou de la coopération qui sont vues comme positives pour soutenir la motivation des apprenants. (Deutsch, 1949) passe en revue les différentes définitions ayant été considérées pour la compétition et la coopération avant de donner la définition suivante pour une situation de compétition sociale : « *In a competitive social situation the goals for the individuals or sub-units in the situation under consideration have the following characteristic: the goal-regions for each of the individuals or sub-units in the situation are defined so that if a goal-region is entered by any individual or sub-unit, (or by any given portion of the individuals or sub-units under consideration) the other individuals or sub-units will, to some degree, be unable to reach their respective goals in the social situation under consideration.* »

En théorie des jeux, la compétition est là encore très peu définie car élémentaire : si un joueur gagne et l'autre perd, il y a compétition. C'est par exemple le cas des jeux à somme nulle. Dans le cadre des jeux sérieux cependant, la compétition peut être utilisée comme source de motivation, voire comme finalité (Burguillo, 2010). Nous pourrions donc nous attendre à ce qu'il soit nécessaire de détecter une telle interaction dans des jeux sérieux multi-joueurs.

Dans la section II.2, nous évoquions les problèmes qu'apportent la diversité et la complexité des interactions sociales. Cette section montre à travers trois exemples que la question de la formalisation des interactions est loin d'être triviale. Or, l'objectif de cette thèse ne pourrait pas être atteint sans un travail de représentation et de spécification formel des scénarios de JSMJ, en vue d'une détection automatique des interactions. Cette formalisation devra être flexible pour tenir compte des différents points de vue sur les définitions des interactions. En effet, nous voudrions que deux personnes, ayant deux points de vue différents sur une même interaction, puissent exprimer leurs deux représentations afin qu'elles puissent être détectées et analysées. En théorie des jeux par exemple, ces critères sont généralement centrés sur la réussite ou non de la tâche. En EIAH, cet aspect est plus nuancé et se centre davantage sur l'apprentissage et la construction des connaissances. Deux experts issus de deux domaines différents peuvent se centrer sur des aspects différents d'une interaction. Par exemple, un concepteur cherchant à détecter la collaboration ne s'intéressera pas forcément aux connaissances ou compétences acquises ou mobilisées, à la différence d'un enseignant. Les jeux sérieux sont co-construits par des personnes venant de domaines divers (*game designer*,

enseignants, experts du domaine, etc.), il est donc primordial de prendre en compte leurs points de vue et de leur permettre de les exprimer.

## II.2.e. Détection et analyse des interactions

Les interactions dépendent souvent du contexte dans lesquelles elles sont réalisées. L'analyse de celles-ci demande donc de pouvoir comprendre l'état de l'environnement avant, pendant et après l'interaction. (Dyke, Girardot, Lund, & Corbel, 2007) relèvent trois sources possibles d'informations pour retranscrire une interaction médiée par ordinateur : un enregistrement vidéo des participants, les traces du système et enfin la capture vidéo du système.

Ces trois sources ont, selon (Dyke et al., 2007), chacune leurs inconvénients. L'enregistrement vidéo demande au chercheur d'effectuer une retranscription des dialogues et actions des participants. Établir cette retranscription s'avère généralement coûteux en termes de temps et peut engendrer une perte d'information. Les traces sont quant à elles analysables automatiquement par un système, mais peuvent s'avérer difficile à lire pour un humain. La capture vidéo du système est plus facilement exploitable par un humain que les traces mais nécessite également une retranscription et ne rend pas toujours compte des événements surgissant à l'extérieur du système.

Dyke et ses collègues (Dyke et al., 2007) identifient alors un certain nombre d'outils destinés à favoriser l'analyse manuelle de ses différentes sources afin de mieux appréhender les interactions médiées par ordinateur. En analysant 4 outils : *ColAT* (Avouris, Fiotakis, Kahrimanis, Margaritis, & Komis, 2007), *Replayer* (Morrison, Tennent, & Chalmers, 2006), *The Abstract trace analysis tool* (Georgeon et al., 2007) et *DREW* (Corbel et al., 2003), ils établissent la liste des besoins des chercheurs quant à ce type d'analyse : synchroniser différentes sources, annoter les données et enfin pouvoir re-visualiser l'état du système.

Une autre approche intéressante est celle de (Nüssli, Jermann, Sangin, & Dillenbourg, 2009). Dans leurs travaux, les chercheurs utilisent un système d'*eye tracking* et d'enregistrement audio qu'ils synchronisent avec une capture vidéo du système. L'idée est alors d'utiliser des techniques de *data mining* afin de prédire la qualité d'une collaboration, ainsi que la capacité des apprenants à résoudre la tâche qui est la leur. Si la question de la détection de l'interaction elle-même n'est pas posée, la question de la détection de l'implication des apprenants et de la manière dont ils partagent les tâches est intéressante. Ces deux éléments font parties intégrantes de la définition de (Dillenbourg, 1999) de la collaboration. Notre approche visant à établir la détection automatique des interactions dans les JSMJ, le fait qu'un système permette d'évaluer celles-ci automatiquement est encourageant pour notre recherche.

L'ensemble des outils que nous venons d'évoquer peuvent être employés dans le cadre des jeux sérieux, mais ne leur sont pas spécifiques. L'aspect ludique des jeux, qui peut servir de source de motivation pour les joueurs, y est donc le plus souvent ignoré. L'une de nos hypothèses est qu'il est possible de détecter des interactions en se basant sur le *game design* du jeu. Ces outils fournissent donc des pistes intéressantes mais ne peuvent être utilisés tels quels dans le cadre de notre approche.

Des éléments de réponse en faveur de notre hypothèse sont fournis par le travail de (Wendel et al., 2013). Ces derniers passent en revue les différents éléments nécessaires à la conception d'une tâche collaborative, ainsi que ceux nécessaires à la conception d'un jeu sérieux collaboratif. L'idée est alors d'utiliser le *game design* du jeu pour assurer la collaboration à des moments clés du jeu. Ils ont donc développé un jeu sérieux, « *Escape from Winston Island* » (pour lequel nous n'avons pas de description complète du scénario et de ses activités), dans lequel des singes doivent construire un radeau et naviguer sur un océan pour s'échapper d'une île. La construction et la conduite du radeau exigent que les joueurs se coordonnent. Ceci fait directement écho à la définition de la collaboration donnée par (Dillenbourg, 1999). Un tel travail montre qu'il est a priori possible de soulever une interaction spécifique en passant par le design du jeu et de son scénario. Cela renforce donc l'hypothèse émise en introduction selon laquelle la description du scénario pourrait permettre de déterminer les interactions pouvant émerger dans un jeu donné. Le travail de (Wendel et al., 2013), bien que n'ayant pas vocation à permettre l'analyse ou la détection d'interaction, apporte une information importante : si des éléments de *game design* adéquats sont présents, une interaction spécifique (ici la collaboration) peut être possible. Un système capable de détecter ces éléments de *game design* serait donc en mesure de détecter cette interaction.

Une autre approche pour l'analyse des interactions dans le cadre des jeux sérieux, consiste en une approche mixte entre les outils d'aide à l'analyse et les outils de design. Dans (Gendron, Carron, & Marty, 2008), les chercheurs établissent une liste d'indicateurs collaboratifs pour aider un enseignant à analyser une collaboration en temps réel. Pour tester leurs indicateurs, ils font alors une expérimentation avec un *donjon pédagogique* (Carron, Marty, & Heraud, 2008) sur lequel l'enseignant à la main en tant que concepteur et observateur.

Comme dans « *Escape from Winston Island* », les activités collaboratives sont directement intégrées et spécifiées à l'avance dans le scénario du jeu. L'enseignant peut alors contrôler l'avancée des joueurs, observer leurs réactions et réponses en temps réels et ainsi déterminer à l'aide des indicateurs fournis quels seraient les groupes les plus à même d'assurer une bonne collaboration selon l'enseignant. Ce travail de supervision répond donc à la fois aux critères établis par (Dyke et al., 2007), mais aussi aux contraintes de *game design* utilisées par (Wendel et al., 2013). Ces approches sont cependant spécifiques à la collaboration et sont difficilement transposables dans un jeu sérieux n'ayant pas eu cette réflexion préalable. En effet, dans les deux cas, les éléments de *game design* contraignent l'apprenant à la collaboration, un jeu sérieux n'ayant pas eu cette phase de réflexion en amont sur le *game design* demandera des efforts supplémentaires pour identifier la présence de ces éléments dans le jeu.

## II.3. Représentation et modélisation des scénarios de JSMJ

Nous nous sommes fixés comme objectif de travailler sur la détection automatique des interactions entre pairs à partir de la description des scénarios de JSMJ, en amont de leur utilisation par des apprenants. Afin de permettre cette détection automatique, il est indispensable de procéder à la formalisation de scénarios de JSMJ. Dans les sections qui

suivent, nous présentons des travaux issus de l'état de l'art sur la modélisation de scénarios d'apprentissage et sur les activités d'apprentissage qui les composent.

## II.3.a. Théorie de l'activité

(Roth & Lee, 2007) établissent l'origine de la théorie de l'activité aux travaux de Vygotsky (Vygotsky, 1978). Cette théorie permet d'organiser et de regrouper plusieurs tâches en factorisant ces dernières selon le(s) individu(s) impliqué(s), le(s) outil(s) utilisé(s) ainsi que sur la finalité des actions entreprises. Selon cette théorie, dans sa première génération, il est possible de décomposer une activité en trois composantes principales : un sujet, un objet et des outils (Figure 1).

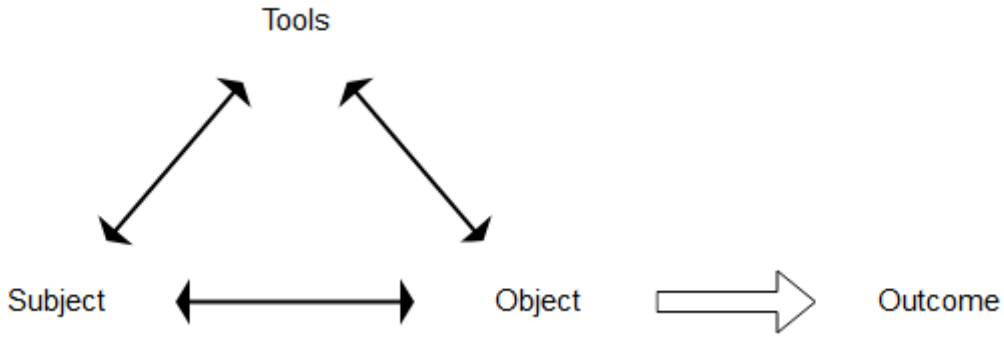


*Figure 1 : Représentation de la première génération de la théorie de l'activité (Engeström, 2001).*

L'idée est alors la suivante : le sujet agit pour atteindre l'objet (le but du sujet) à l'aide des outils (les ressources qu'il a à sa disposition) afin d'obtenir un/e produit/sortie donné/e. Un objet peut être un élément matériel ou quelque chose de moins tangible comme un plan, voir complètement intangible comme une compétence ou une idée tant que la quête de celui-ci peut guider le sujet. Par ailleurs une activité est considérée comme le contexte minimal dans lequel les actions du sujet ont une signification. Les activités sont réalisées comme des tâches individuelles ou collectives dans lesquelles les actions sont enchaînées et reliées les unes aux autres par la même motivation et par le même objet.

Cette vision est particulièrement intéressante dans le cadre d'un jeu vidéo ou d'un jeu sérieux. Il est en effet relativement facile d'établir la liste des outils à la disposition des joueurs et de tracer leur utilisation. Il est également aisé de retrouver les sorties de ses actions et les sujets impliqués. Les logs d'un jeu sérieux deviennent alors la donnée d'entrée permettant de construire une liste d'activités correspondant en tout point aux critères évoqués plus haut.

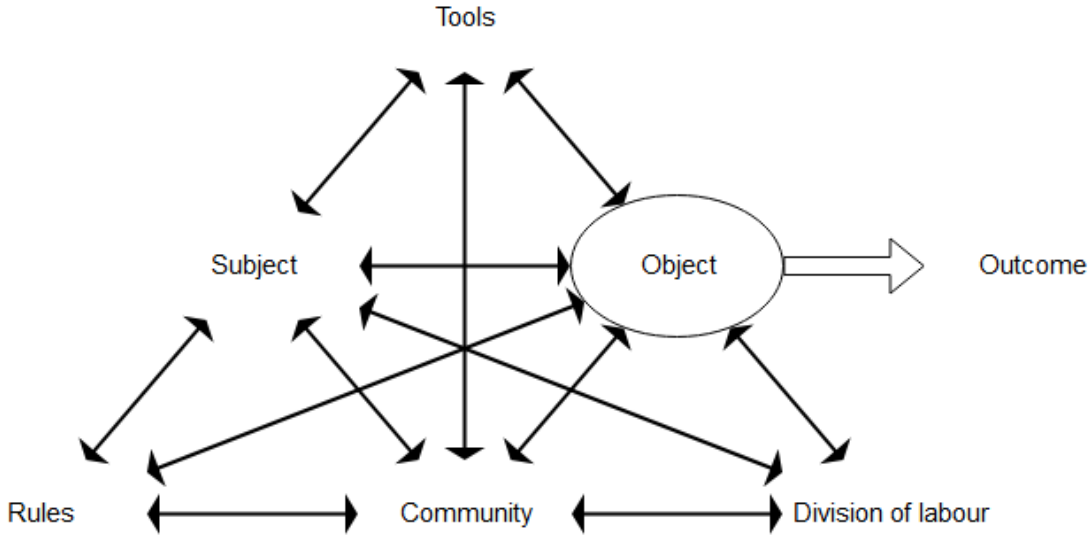


*Figure 2 : Représentation de la seconde génération de la théorie de l'activité (Engeström, 2014).*

Les travaux d'Engeström (2001), sont particulièrement importants pour la théorie de l'activité. Dans (Engeström, 2001), Engeström établit une représentation schématique de la seconde génération (Figure 2) issue des travaux de Leont'iev. Dans celui-ci, trois nouveaux concepts sont ajoutés à ceux de la première génération : les règles, la communauté et la distribution des tâches (ou division du travail). Le concept de règles reflète les différentes contraintes et protocoles que les sujets doivent respecter dans leur quête de l'objet. Le concept de communauté représente les individus et groupes sociaux avec lesquels les sujets peuvent interagir et communiquer au sein de l'activité. Pour finir, le concept de distribution des tâches indique comment les sujets se répartissent les différentes tâches pour atteindre l'objet.

Nous remarquerons également sur la Figure 2, la présence d'un ovale autour de l'objet. Celui-ci provient des travaux d'Engeström qu'il explique ainsi : « *The object is depicted with the help of an oval indicating that object-oriented actions are always, explicitly or implicitly, characterized by ambiguity, surprise, interpretation, sense making, and potential for change.* » Cette citation laisse suggérer que dans la vision d'Engeström de la théorie de l'activité, l'objet d'une activité est mouvant et / ou mal défini pour les sujets.

Une telle approche est donc particulièrement avantageuse lorsqu'il s'agit de jeux sérieux. Cependant celle-ci accorde également énormément de liberté quant à la granularité de l'activité et demeure floue quant à la représentation de ses différentes composantes. Il est donc possible de représenter un jeu complexe, tel que le jeu d'échec, par une seule activité (à somme nulle : un perdant et un gagnant) ou par une suite d'actions mono-joueur où les joueurs déplacent à tour de rôle des pions sur un plateau. Les interactions que nous pourrons alors détecter dépendent de la granularité de la description. Dans le premier cas, les deux joueurs sont en compétition alors que dans le deuxième cas aucune interaction ne peut émerger d'une action mono-joueur. Cette question de la granularité est loin d'être anodine dans le cadre de l'étude des interactions. En effet, il ne peut y avoir d'interactions dans une activité que si celle-ci est multi-joueurs. De plus, les joueurs peuvent avoir une certaine liberté d'actions grâce aux ressources dont ils disposent, et pourraient faire émerger des interactions

différentes. L'utilisation d'activités conformément à la théorie de l'activité nécessite donc de prendre en compte le degré de granularité (quelle granularité envisager lors de la description d'une activité ? la granularité d'activité pourrait varier d'une seule action atomique, l'action de jeu, à l'ensemble du jeu). Pour un niveau de granularité donné, il est également important de prendre en compte le fait que certaines activités peuvent ne pas être intéressantes à modéliser, engendrant donc une abstraction de celles-ci.

## II.3.b. IMS LD : un langage de scénarisation

IMS LD (trouvant son origine dans le langage de modélisation éducationnelle de l'*Open University of Netherlands* (Hummel, Manderveld, Tattersall, & Koper, 2004)) est un cas intéressant de *framework* utilisant la théorie de l'activité. Il permet plus particulièrement la description de scénarios d'apprentissage. Un certain nombre de tentatives ont été effectuées pour développer des outils auteurs utilisant les spécifications de IMS LD. L'idée était alors de créer des outils suffisamment simples pour que des enseignants et des ingénieurs pédagogiques « non techniques » puissent les utiliser (Hermans, Janssen, & Koper, 2016). McAndrew et ses collègues (McAndrew, Goodyear, & Dalziel, 2006) ont par ailleurs relevé que même à son échelle la plus basse (le niveau A), IMS LD a le potentiel pour décrire des tâches collaboratives complexes impliquant des rôles multiples et différents outils.

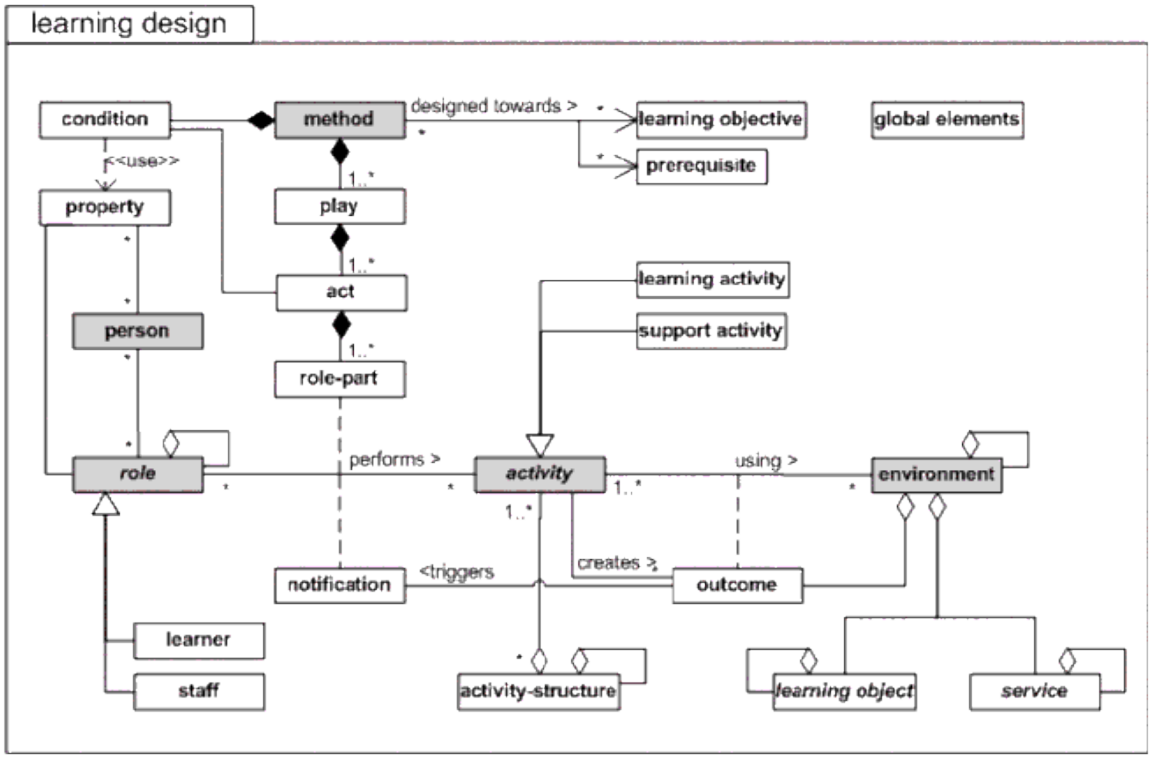


*Figure 3 : Diagramme UML officiel de IMS LD (Hummel et al., 2004)*

Dans IMS LD (voir Figure 3), les individus assument des rôles (comme apprenant ou enseignant). A travers ses rôles, ils accomplissent des activités dans un environnement composé d'objets d'apprentissages et de services afin d'atteindre des objectifs spécifiques.

Les rôles et les activités sont synchronisés à travers un acte. Ces actes composent le jeu et sont effectués consécutivement les uns à la suite des autres.

Toutefois, bien que ce langage fût particulièrement performant au vu de son expressivité sur le plan pédagogique (Derntl, Neumann, Griffiths, & Oberhuemer, 2012), Il n'a jamais vraiment réussi à être adopté à cause de sa complexité (Hermans et al., 2016).

### II.3.c. Pléiades : un outil flexible pour la scénarisation pédagogique

La méthode des Pléiades (David, George, Godinet, & Villiot-Leclercq, 2008; Villiot-Leclercq, 2007; Villiot-Leclercq & David, 2007) pallie aux problèmes d'IMS LD en fournissant une structure davantage réutilisable et manipulable par des enseignants. Ces enseignants (qualifiés d'enseignant-auteur) sont ainsi en mesure de formaliser les actions et les activités que leurs apprenants peuvent rencontrer au sein d'une SACI (Situation d'Apprentissage Collective Instrumentée) et les réutiliser pour des scénarios similaires. Le scénario donné en exemple est axée sur la représentation d'une étude de cas.

Cette méthode prend en compte la granularité (David et al., 2008) p.16: « Nous avons ainsi défini trois niveaux de granularité : un scénario de type activité élémentaire est une étoile, un regroupement de plusieurs activités est une pléiade, un regroupement de plusieurs regroupements ou pléiades est une constellation. Chaque grain est défini comme une entité cohérente dont le sens se construit par rapport aux choix et aux intentions pédagogiques de l'enseignant-auteur et par l'interprétation qu'en fera l'enseignant-lecteur. ».

Nous sommes donc en présence d'un système de formalisation de scénario répondant à un certain nombre de critères :

- Réutilisable
- Conforme à la théorie de l'activité
- Utilisable dans le cadre des jeux sérieux (qui peuvent correspondre à des Situations d'Apprentissage Collectives Instrumentées (Michel, Garrot, & George, 2007))
- Prend en compte les problèmes de granularité

La méthode des Pléiades est une méthode de scénarisation pédagogique. Les liens établis entre les différentes activités et ressources utilisées par les joueurs sont contraints par les enjeux pédagogiques de la Situation d'Apprentissage Collective Instrumentée décrite par l'enseignant. L'approche que nous avons choisie est de considérer que ce sont les objectifs, les ressources du jeu et les stratégies des joueurs qui décident de leurs comportements et donc des actions qu'ils réalisent dans le scénario. La formalisation apportée par la méthode de Pléiades est très intéressante mais ne semble pas être suffisante dans le cadre de cette thèse. En effet, la scénarisation pédagogique s'articule autour des connaissances et compétences apprises par les étudiants. Elle permet de vérifier une acquisition graduelle et logique de celles-ci. En revanche, ce type de scénarisation ne permet pas de reproduire la dynamique de jeu créée par les ressources et actions à disposition des joueurs. Dans notre cas,

c'est cette dynamique qui permet de déterminer si les étudiants sont en mesure ou non d'interagir entre eux.

### II.3.d. MoPPliq : un modèle pour la scénarisation de Jeux sérieux mono-joueur

Une des propriétés recherchées dans les scénarios pédagogiques en général et dans les scénarios de JSMJ en particulier est la non linéarité. *MoPPliq* (Marne, 2014) fournit une approche intéressante pour ce problème. L'idée derrière ce modèle est de relier les différentes activités du jeu en se basant sur des relations de prérequis existant entre les entrées et les sorties d'une activité. L'avantage de cette méthode est qu'elle permet entre autres d'établir des scénarios incluant cycle, embranchement et répétition, en se basant sur la description des activités incluant à la fois des éléments ludiques et pédagogiques.

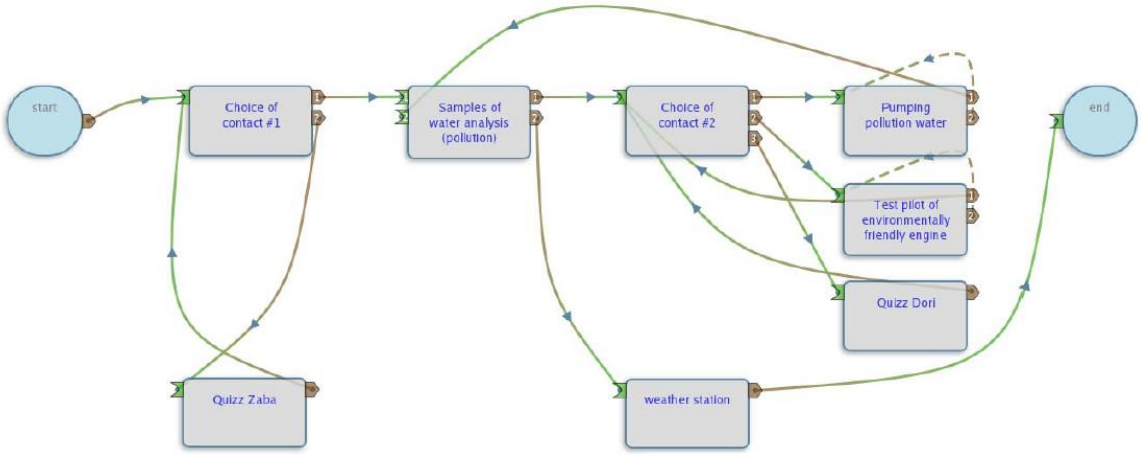


*Figure 4 : Cas d'utilisation de MoPPliq (Marne, 2014)*

Dans la Figure 4 qui représente un cas d'utilisation de *MoPPliq*, les rectangles représentent les activités et les cercles le début et la fin du jeu. Chaque activité est dotée d'une ou plusieurs entrées (languettes vertes sur la gauche) et d'une ou plusieurs sorties (languettes marron sur la droite). Chaque sortie ne doit être reliée qu'à une unique entrée. Dans le cas où la sortie d'une activité ferait référence à une entrée de la même activité, la flèche directionnelle les reliant est alors en pointillé. Au regard de cette Figure 4, nous pouvons nous apercevoir que *MoPPliq* permet la répétition d'activités, mais aussi la multi linéarité du scénario et autorise les cycles dans celui-ci. Ce genre de structure se révèle donc particulièrement intéressante pour représenter graphiquement des scénarios complexes.

*MoPPliq* a cependant plusieurs limites qui ne nous permettent pas d'envisager son utilisation dans le cadre de cette thèse. Tout d'abord, il est destiné exclusivement au traitement de jeux mono joueur. Ensuite, il ne permet pas d'associer des objectifs au joueur ou au rôle que ce dernier assume. Or, il s'avère que des interactions comme la compétition sont dépendantes des objectifs des joueurs : un joueur est en compétition parce que la réussite d'un autre ne lui permettra pas de réaliser ses propres objectifs. L'absence d'objectifs pour les joueurs ou leur rôle rend donc a priori impossible la détection de certaines interactions.

## II.3.e. Outils auteurs de jeux sérieux

Un certain nombre de jeux sérieux permettent également aux enseignants de faire leur propre scénario comme *ClassCraft* (Sanchez et al., 2017) ou encore comme le *donjon pédagogique* de (Carron et al., 2008). Dans *ClassCraft*, l'enseignant crée des groupes de joueurs aux capacités diverses. Les joueurs (des élèves) doivent répondre à des questionnaires leur permettant par exemple de battre des monstres afin de récupérer des points d'expérience ou de l'or. Les joueurs sont également amenés à effectuer des tâches chez eux comme rendre un devoir en avance pour que l'enseignant les récompense. À l'aide de leurs points, les élèves achètent des pouvoirs qu'ils peuvent utiliser en classe : avoir une antisèche, pouvoir changer de place, avoir un délai supplémentaire pour un travail, … *ClassCraft* sert donc à ludifier la salle de classe. Le *donjon pédagogique* fonctionne également sur un système de questionnaires que l'enseignant corrige au fur et à mesure que les élèves répondent. Des bonnes réponses permettent aux élèves de progresser dans le jeu. Certaines salles collaboratives demandent aux élèves d'obtenir chacun suffisamment de clés pour avancer ou de fournir des réponses communes par exemple.

Ces jeux ont plusieurs avantages. Tout d'abord ils sont multi-joueurs. De plus, leur approche est similaire à celle d'un cours et sont, par conséquent, relativement faciles à prendre en main, y compris pour quelqu'un ne possédant pas de connaissances techniques. La représentation, la compréhension et la description des scénarios peuvent donc s'y faire relativement aisément. Cependant, la visée d'origine de ces deux jeux étant de gamifier et scénariser des classes scolaires (ce qui explique leur proximité avec les structures habituelles des cours), ils montrent des limites importantes pour représenter et scénariser d'autre type de *gameplay* ou à formaliser des jeux déjà existant au *gameplay* similaire. De plus, dans le cas de *ClassCraft*, la notion de collaboration est prédominante, la compétition devant être introduite par le biais de moyens externes au jeu. Ainsi, si les deux structures demeurent intéressantes, en particulier dans le cadre scolaire, elles ne peuvent être utilisées pour formaliser et représenter des scénarios de jeux sérieux de manière plus générale.

Une approche assez répandue consiste à fournir des outils auteurs permettant aux enseignants de développer leur propre scénario de jeux. Nous citerons par exemple des plateformes en ligne comme Emergo[2] (Slootmaker, Kurvers, Hummel, & Koper, 2014), *Fablusi*[3] ou en encore *Cyberdam*[4] (Maharg & Nicol, 2009) qui ont l'avantage de permettre la création de jeux où les apprenants peuvent assumer des rôles différents avec des caractéristiques spécifiques. Cet aspect est relativement important pour représenter ou détecter des interactions dans lesquelles les joueurs assument des positions asymétriques, ou lorsque le rôle de ceux-ci évolue au cours du jeu. Ces jeux s'organisent sur le long terme (5 à 6 semaines pour *Cyberdam* par exemple) et consistent en un mélange d'évènements réels et virtuels. La partie virtuelle est là pour structurer le jeu et sert à informer les différents participants des résultats de leurs actions. Les joueurs sont entièrement libres des actions qu'ils entreprennent.

[2] https://sourceforge.net/projects/emergo/
[3] http://www.fablusi.com/
[4] http://www.cyberdam.nl/

Les actions de l'un impactent les autres et la simulation avance en fonction des cas présentés. Ces outils sont intéressants car ils permettent une grande diversité d'interactions, les joueurs y agissent similairement à une simulation participative en suivant le rôle qui est le leur. Les scénarios s'apparentent alors à des cas d'études pour lesquels l'enseignant a défini les rôles, les buts, les réponses possibles, etc.

Ces outils ne permettent cependant que des actions asynchrones et semblent inadaptés pour une utilisation autre que pour des cas d'études. La description des actions individuelles y est assurée, mais les activités multi-joueurs sont en revanche beaucoup plus vagues quant aux ressources que les joueurs doivent utiliser. Or, l'objectif poursuivi par cette thèse est la détection automatique d'interactions dans des activités multi-joueurs. De par cette limite, l'utilisation de ces outils, dans le cadre de ce travail, serait conditionnée par des modifications importantes pour y palier.

En conclusion, les différents systèmes et *framework* qu'il est aujourd'hui possible d'employer afin de formaliser un jeu sérieux sont bien souvent limités sur certaines caractéristiques importantes dans le cadre de cette thèse.

# II.4. Analyse a priori des scénarios de JSMJ

L'hypothèse que nous avons formulée en introduction est qu'il est possible de détecter en amont les interactions que réaliseront les apprenants si nous sommes en mesure de savoir quelles situations de jeu ces derniers peuvent rencontrer. Exploiter un scénario consiste donc dans notre approche à établir les séquences d'activités que peut réaliser un joueur lors des phases de jeu.

Nous envisageons de nous inspirer de la théorie de l'activité pour décrire des activités multi-joueurs incluant des ressources de jeu, des connaissances, des buts et des rôles. Dans le cadre de ce travail, ces activités constituent donc l'unité de base pour l'analyse des interactions. Déterminer dans quelles situations les joueurs peuvent se trouver, revient donc à déterminer quelles activités sont nécessaires pour atteindre la fin du jeu.

## II.4.a. STRIPS

L'activité finale d'un jeu permet de connaître les contraintes que les joueurs doivent satisfaire pour atteindre la fin d'un jeu. STRIPS (Fikes & Nilsson, 1971) est un travail de référence sur la planification et la recherche de chemins à l'aide d'actions et d'activités pour atteindre une situation finale donnée. Dans STRIPS, l'évolution de la simulation se fait étape par étape, un inventaire des objectifs (nos contraintes) de l'agent (notre apprenant) est présent, ainsi qu'un ensemble de prédicats concernant l'état actuel du monde. L'agent a à sa disposition plusieurs activités consommant un prédicat du monde ou plusieurs prédicats du monde pour en produire un nouveau. Pour établir quelle suite d'activités effectuer, le système cherche une activité produisant l'un des objectifs et le met en bas de la pile des actions que devra exécuter l'agent. Les prédicats nécessaires à la réalisation de cette action sont rajoutés à la liste des

objectifs. L'opération est alors répétée jusqu'à ce que la liste des objectifs corresponde aux prédicats actuels de description du monde.

L'avantage de ce type de méthode, et plus particulièrement de STRIPS, est qu'il devient alors possible de trouver un chemin pour atteindre la fin d'un jeu. Cependant STRIPS ne permet d'obtenir qu'un seul chemin ce qui ne satisfait pas les besoins de ce travail puisqu'une seule situation est ici déterminée. STRIPS est un langage de programmation à la fois impératif et fonctionnel. Et bien que ce dernier ne puisse être utilisé tel quel pour exploiter le scénario, l'utilisation de fonctions pour représenter les actions et des listes d'objectifs à satisfaire sont des concepts particulièrement intéressants. En prenant ces concepts en considération, l'utilisation d'automates pour représenter les modifications des ressources et inventaires devient alors envisageable.

### II.4.b. Réseaux de Petri

Les réseaux de Petri (RdP) (Reisig, 2012) sont des modèles mathématiques puissants pour modéliser la dynamique des systèmes complexes. Plusieurs méthodes de *model checking* ont été proposées pour vérifier les propriétés d'un système complexe en analysant le graphe d'état d'un système ou graphe d'accessibilité. L'aspect formel de ce modèle rend possible son analyse automatique à l'instar du système *LaaLys* (Muratet, Yessad, & Carron, 2016).

Formellement, un RdP est un graphe biparti, valué et composé de deux types de nœuds : des places et des transitions. Chaque arc permet de connecter une place à une transition ou une transition à une place. Les places sont marquées et contiennent un nombre de jetons positif ou nul. Le vecteur qui associe à chaque place son marquage est appelé marquage du RdP et représente son état à un instant donné. Une transition du RdP peut être exécutée si elle est sensibilisée, c'est-à-dire, si chacune de ses places d'entrée satisfait la contrainte définie par le type et le poids de l'arc qui la relie à cette transition. Lorsqu'une transition est exécutée, des jetons des places d'entrée sont consommés (en fonction du type de l'arc) et d'autres jetons sont ajoutés aux places de sortie. L'ensemble des marquages atteignables à partir du marquage initial du RdP en exécutant les transitions sensibilisées représente l'ensemble des états du système et est appelé le graphe d'accessibilité.

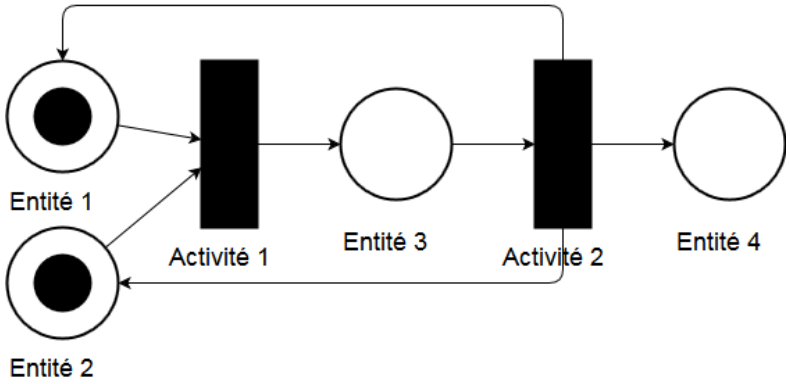

*Figure 5 : Exemple de réseau de Petri*

La Figure 5 présente un exemple d'un réseau de Petri classique. Dans notre contexte, le formalisme des réseaux de Petri pourrait représenter un scénario de jeu de la manière suivante :

- Les places modéliseraient les différentes entités du jeu
- Les jetons représenteraient les données des entités
- Les transitions modéliseraient les activités du scénario
- Les arcs entre les places et les transitions spécifieraient les contraintes permettant de réaliser une activité en fonction des données des entités du jeu

Ainsi, une activité (transition) ne pourrait être réalisée (déclenchée) que si les prérequis (jetons dans les places) sont suffisants (type et poids des arcs). Une activité réalisée modifierait l'état des entités du jeu (places) en produisant de nouvelles données (jetons). Modéliser un scénario consisterait donc à exprimer les contraintes autorisant la réalisation des activités en fonction de l'état des différentes entités du jeu.

Le formalisme des réseaux de Petri possède cependant des limites. La plus grande de ces limites est l'explosion du graphe d'accessibilité lorsqu'une activité peut être répétée indéfiniment. Dans ce cas un graphe de couverture peut être calculé afin de fusionner les états divergents en un nombre fini d'états. Cette fusion introduit alors une approximation des états du scénario et il deviendrait alors impossible de vérifier l'ordonnancement de deux états du scénario.

Des extensions des réseaux de Petri existent. Nous citerons notamment les réseaux de Petri colorés (Beaudouin-Lafon et al., 2001). Ces derniers permettent de définir des attributs aux différents jetons, de placer des contraintes sur les arcs et transitions, et enfin d'effectuer des opérations sur les attributs des jetons. Ces particularités font des réseaux de Petri colorés des candidats particulièrement intéressants pour la représentation et la simulation de fonctions.

Les solutions offertes par STRIPS et les réseaux de Petri ne répondent pas toutes deux complètements à notre problématique. Un élément important de notre travail est de pouvoir déterminer à l'avance les chemins que peut suivre un apprenant. STRIPS cherche une solution sans pour autant chercher à employer toutes les actions qui lui sont fournies. Il est envisageable que certaines des actions ainsi ignorées puissent soulever des interactions différentes de toutes celles soulevées par les actions employées par STRIPS. En ce sens, STRIPS n'est pas utilisable en l'état. L'utilisation de réseaux de Petri pour exploiter la description des

scénarios auraient pu être envisageable. L'utilisation de ceux-ci en l'état aurait nécessité des modifications comme une borne pour éviter les cycles (les actions peuvent être répétées), mais aussi de pouvoir revenir à un état antérieur du réseau en cas d'impasse avant de pouvoir explorer une autre piste. Les graphes de couvertures de Rdp auraient pu répondre à cette problématique mais auraient là encore nécessité des modifications importantes, les graphes générés pour des scénarios complexes étant particulièrement grands. Ainsi, ces deux modèles ne sont pas utilisables en l'état, mais sont très intéressants pour notre approche. C'est pourquoi, dans le cadre de ce travail et pour répondre à celle-ci, nous avons décidé de nous appuyer sur ceux-ci pour développer notre propre modèle de scénarisation permettant l'analyse à priori de scénarios de JSMJ.

La vérification de l'hypothèse selon laquelle il est possible d'exploiter un jeu sérieux à priori nécessite la mise en place de deux tâches majeures. Premièrement, établir un modèle de formalisation de scénarios de JSMJ. Deuxièmement, créer un algorithme fonctionnant dans des cas de scénarisation complexe et fournissant des solutions dépendantes des situations / activités ciblées.

## II.5. Synthèse

L'ambition de cette thèse est de permettre la détection automatique des interactions dans les Jeux sérieux multi-joueurs dédiés à l'apprentissage (JSMJ). Pour ce faire, nous avons décidé de baser notre approche sur l'analyse de leur scénario. Notre idée est la suivante : les ressources et actions à disposition des joueurs contraignent les interactions auxquelles ceux-ci seront soumis dans le cadre du jeu. Or, la description formelle d'un scénario de jeu permettrait d'établir les ressources et actions à disposition des joueurs. De fait, notre approche nécessite donc d'être en mesure d'effectuer trois tâches :

1. Modéliser formellement les scénarios de JSMJ,
2. Modéliser formellement les interactions entre apprenants dans le cadre d'un scénario de JSMJ formalisé,
3. Analyser les chemins effectuables dans un scénario de JSMJ à partir de sa description.

La revue de l'état de l'art indique clairement qu'il n'existe pas de définitions consensuelles pour les interactions. L'exemple le plus flagrant étant celui de la collaboration et de la coopération. Pour traiter la tâche 2, nous souhaitons donc explorer une approche donnant une solution à cette problématique. Le chapitre IV présente en détail la solution que nous avons apportée pour pouvoir décrire formellement des interactions, tout en palliant à la problématique issue de la non-existence de définitions consensuelles.

De plus, en dépit de solutions intéressantes comme *MoPPliq* ou les Pléiades, l'état de l'art n'a pas permis de faire émerger un modèle permettant de formaliser des scénarios de JSMJ se basant sur les ressources et actions de jeu accessibles aux joueurs pour recréer la dynamique du jeu. Cette dynamique étant selon nous responsable de l'émergence des interactions que nous souhaitons détecter (à travers le *game design* notamment), nous avons décidé de proposer notre propre modèle. Celui-ci est présenté dans le chapitre III.

Établir à l'avance les chemins parcourus par les joueurs, permettrait de prédire les situations dans lesquelles ceux-ci pourraient se retrouver et donc de déterminer les interactions générées par un jeu sérieux. Trouver une méthode pour obtenir automatiquement les chemins d'un scénario de JSMJ demande donc de réfléchir à la question suivante : Comment analyser un scénario de JSMJ ?

Les différentes méthodes de planification comme STRIPS ou d'exploitation des automates comme les réseaux de Petri, fournissent des éléments de réponse quant à la manière d'analyser un scénario.

Dans le cas de STRIPS, la planification s'effectue à l'aide d'activités consommant une ou plusieurs ressources pour en produire d'autres. L'idée est alors de produire un nouvel état se rapprochant de l'état final désiré. En appliquant à notre scénario des activités inspirées des systèmes concurrents (la dynamique du système pouvant être créée par des fonctions produisant ou consommant des ressources (Magee & Kramer, 1999; Manna & Pnueli, 2012)), il devient possible d'utiliser STRIPS sur des scénarios de jeu.

STRIPS possède cependant le désavantage de ne produire qu'un seul chemin et ce sans nécessairement utiliser toutes les activités à sa disposition. L'approche que nous avons décidé d'employer a donc été de concevoir un algorithme s'inspirant de STRIPS et en mesure de remédier aux problématiques liées à STRIPS. Nous présentons cet algorithme dans le chapitre V.

# Section 2
# Contributions

Dans cette section, nous allons décrire nos différentes contributions. Ainsi, la Figure 6 donne un aperçu de l'articulation des différentes contributions que nous proposons pour répondre à nos problématiques de recherche. Pour résumer, dans le chapitre III, nous présentons en détail le modèle que nous proposons pour décrire les scénarios et leurs activités. Le chapitre IV, quant à lui, présente les différentes méthodes de formalisation conçues pour modéliser les interactions. Pour finir, le chapitre V est dédié aux algorithmes développés ayant permis l'analyse à priori de scénario de JSMJ modélisés selon la formalisation proposée en chapitre III.

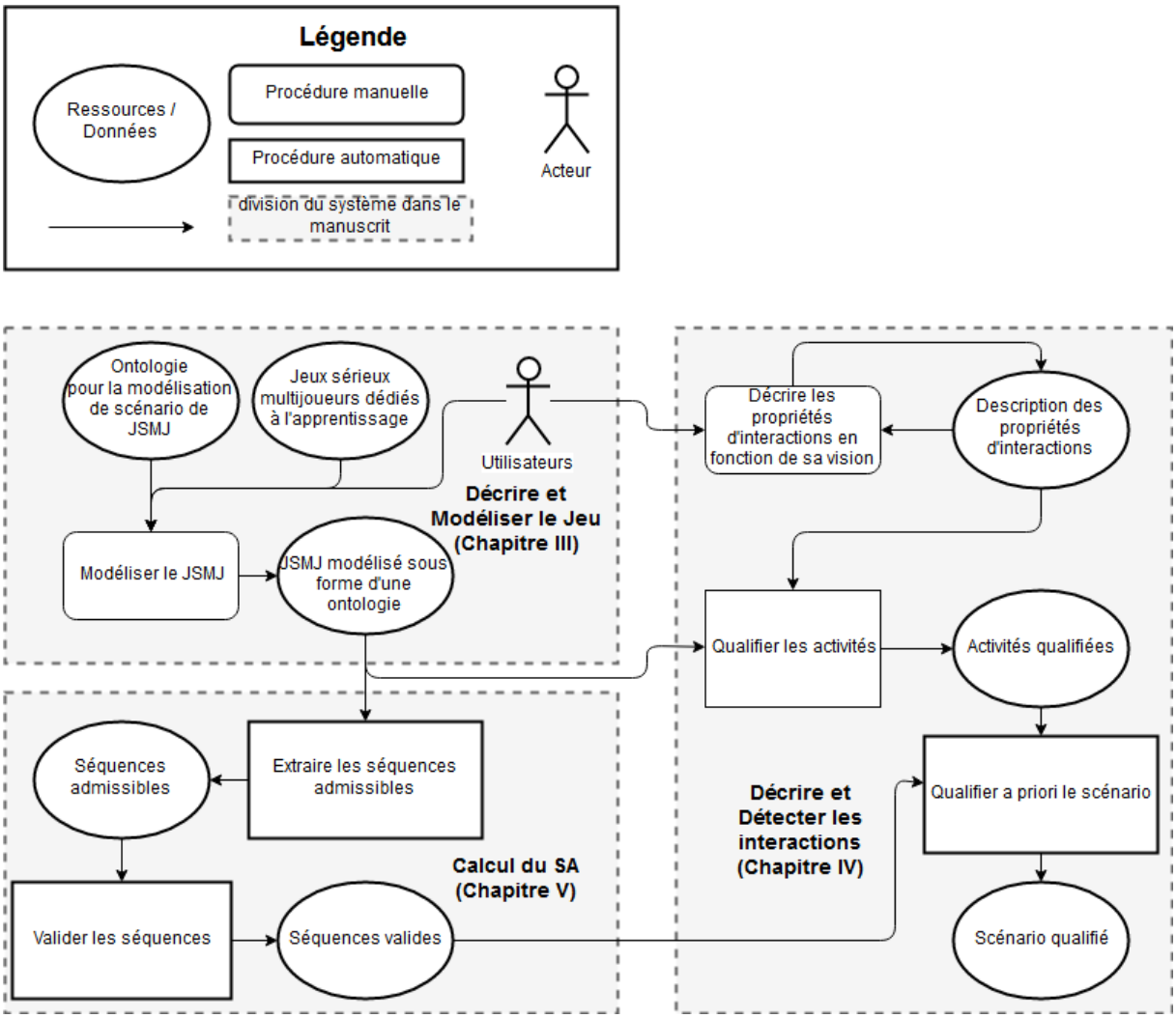


Figure 6 : Vue d'ensemble du système permettant l'analyse à priori de scénario

# III. Modélisation et Formalisation de scénarios de jeux sérieux multi-joueurs

Nous rappelons que l'objectif principal de cette thèse est de permettre la détection automatique des interactions pouvant émerger d'un scénario de JSMJ. Une des conceptions qu'il est possible d'avoir lorsque nous cherchons à analyser des interactions multi-joueurs, est que ces dernières sont dépendantes du contexte et de la situation dans laquelle se trouvent les joueurs. Dans un jeu sérieux multi-joueurs, le scénario fait office de ligne directrice pour les joueurs mais ne présage pas du déroulement du jeu. En effet, un scénario ne fournit pas nécessairement un ordre précis dans lequel exécuter les différentes activités et actions du jeu. En revanche, il fournit l'ensemble des informations et contraintes que les activités doivent remplir avant de pouvoir être effectuées. Il peut également arriver que l'exécution d'une activité donnée soit obligatoire pour pouvoir accéder à la suite du scénario (et donc à de nouvelles activités ou à la fin du jeu). Dans ce cas, le scénario fournit les informations permettant d'identifier ce point de passage obligatoire pour le joueur. Le scénario regroupe donc l'ensemble des informations relatives au jeu et à son exécution, il est alors toujours possible de situer le joueur dans le scénario (qu'a-t-il réalisé, à quelle étape de la résolution en est-il, …).

La modélisation formelle d'un scénario en général et de celui d'un JSMJ en particulier est donc une étape indispensable pour permettre le traitement automatique du scénario. Or, la revue de la littérature ne nous a pas permis d'identifier des modèles ou des langages de description de scénarios en vue de leur analyse automatique et de la détection des interactions. Dans ce chapitre, nous présenterons les choix conceptuels que nous faisons pour modéliser et formaliser un scénario de JSMJ. Ces choix s'appuient sur l'analyse de plusieurs jeux multi-joueurs.

Le positionnement de ce chapitre par rapport aux autres de la section 2 sur les contributions est présenté dans la Figure 6.

## III.1. Contraintes de modélisation d'un scénario JSMJ

### III.1.a. Analyse de jeux multi-joueurs divers

Afin de proposer un modèle adapté pour décrire les scénarios de JSMJ, il faudra dégager les caractéristiques d'un scénario. Pour cela, nous avons analysé manuellement 17 jeux dont 13 sont des jeux vidéo et 4 sont des jeux sérieux dédiés à l'apprentissage. Les jeux vidéo sont usuellement classés en différents genres (stratégie, First Person Shooter, action, plateforme, …). Ces genres sont particulièrement importants car ils définissent la nature du *gameplay* et le type de structure narrative et ludique qu'un joueur peut rencontrer dans un jeu. Les genres pouvant être appliqués aux jeux sérieux suivent les mêmes règles et sont par conséquent les mêmes genres.

Les 13 jeux vidéo sélectionnés sont représentatifs de différents genres couramment rencontrés dans les milieux vidéoludiques ont été choisis pour compléter les structures de *gameplay* présentent dans Voracy Fish, ByteBattle, ClassCraft et LearningAdventure qui sont quatre jeux sérieux.

Les 17 jeux concernés par cette analyse sont les suivants :

1. *Europa Universalis IV* : Jeu de grande stratégie dans lequel chaque joueur prend le contrôle d'un pays qu'il doit faire prospérer. Le jeu se prête davantage à la compétition mais permet tout de même aux joueurs de s'allier et de coopérer.
2. *MineCraft :* Jeu de construction et de survie dans un environnement vaste généré aléatoirement. Il n'y a pas de fin réelle au jeu et les joueurs peuvent interagir librement entre eux.
3. *BattleField III : First Person Shooter.* Deux équipes de joueurs s'affrontent et doivent s'entretuer sur une carte aux contours définis. Les joueurs d'une même équipe sont encouragés à s'entraider. Les rôles des joueurs et leurs capacités peuvent varier.
4. *Portal 2* : *Puzzle Game* dans lequel deux joueurs doivent collaborer pour évoluer à travers les niveaux et atteindre la fin du jeu.
5. *World Of Warcraft* : Jeu de rôle massivement multi-joueurs, les joueurs peuvent s'entraider et s'affronter à travers différentes structures : groupes, factions, guildes. Le but des joueurs est de faire progresser leur personnage en tuant des monstres et des joueurs et en collectant des ressources.
6. *Age of Empire II* : jeu de stratégie en temps réel opposant de 2 à 8 joueurs. Le but des joueurs est de détruire l'ensemble des unités de leurs adversaires. Ils peuvent s'allier ou non en cours de partie en fonction des réglages choisis.
7. *League of Legends* : Il s'agit d'un MOBA, genre dans lequel les joueurs incarnent un héros combattant d'autres héros dans une arène, le but étant de détruire le « *Nexus »* de l'équipe adverse. Les joueurs font évoluer leurs héros en tuant les héros adverses et les monstres disséminés dans l'arène.
8. *HearthStone :* Jeu de cartes opposant deux joueurs. Les cartes, de différents types, permettent d'effectuer différentes actions et d'attaquer l'adversaire. Il est nécessaire de ramener ses points de vie à 0 pour gagner.
9. *Chess* : Jeu d'échec classique numérisé.
10. *StreetFighter :* Jeu de combat où deux joueurs contrôlent un personnage possédant différents coups et aptitudes. Leur but est d'amener la vie de l'adversaire à 0.
11. *DeadSpace 3* : Jeu d'action / horreur incluant un mode coopératif. Les deux joueurs évoluent dans un environnement hostile peuplé de zombies et d'extraterrestres qu'ils doivent affronter à deux. Les ressources trouvées dans leur aventure sont partagées. Ces ressources leurs sont nécessaires pour confectionner des armes et munitions plus puissantes.
12. *Journey* : Jeu de puzzle / aventure dans lequel les joueurs incarnent un être capable de voleter et de chanter. Ils rencontrent d'autres joueurs (aléatoirement) dans leur aventure avec lesquels ils doivent coopérer pour progresser.
13. Pong : oppose deux joueurs devant renvoyer une balle dans l'embut adverse. Le premier à atteindre le score fixé gagne la partie.
14. *Voracy Fish* (Interactive, 2012): Un jeu sérieux visant à favoriser la rééducation de personnes ayant subi un AVC conduisant à une perte importante de mobilité des membres supérieurs. Le jeu s'effectue à l'aide d'une Wiimote et consiste à contrôler

un poisson dans un océan. Le poisson doit ingérer des poissons plus petits que lui pour grandir, éviter les poissons plus grands et autres obstacles (comme des mines aquatiques).

15. *Byte Battle* (Muratet et al., 2012) : Dans *Byte Battle*, un jeu de stratégie en temps réel, les apprenants codent des programmes informatiques visant à contrôler des unités de jeux. Le scénario étudié ici est la version « Tournoi » de *Byte Battle* dans lequel deux apprenants s'affrontent. Leur but est alors de détruire entièrement les unités en possessions de l'équipe adverse.
16. *Learning Adventure* (Marty & Carron, 2011) : *Learning Advendure* permet à des apprenants d'évoluer dans un environnement partagé similaire à un MMORPG. Dans le scénario que nous avons sélectionné, les apprenants doivent reconstituer une manette de jeu basée sur un Arduino en s'aidant des informations et ressources apportées par le jeu.
17. *ClassCraft*[5] : *ClassCraft* est un jeu sérieux permettant de ludifier une classe scolaire typique. Les élèves sont confrontés à des questionnaires dont la résolution leur apporte des points d'expérience ou de l'or. Ils peuvent par la suite utiliser ces ressources pour acquérir des privilèges et outils utiles dans le cadre du cours (comme le droit d'ouvrir la fenêtre ou le droit à posséder une antisèche par exemple). Nous nous sommes basés sur le témoignage d'une enseignante ayant par ailleurs introduit dans le scénario un système de classement des différents élèves avec un prix à la clé pour la meilleure équipe.

Lorsque l'un de ces jeux possède un mode multi-joueurs et un mode mono-joueur, nous considérons uniquement le mode multi-joueurs. Pour mener à bien notre analyse, nous avons défini la grille d'analyse suivante :

1. Nombre de joueurs impliqués
2. Uniformité des actions à disposition des joueurs
3. Liberté d'actions laissé aux joueurs
4. Format du jeu (arcade, bac à sable, scénarisé)
5. Nombre de chemins différents disponibles pour atteindre la fin du jeu
6. Nombre de fins différentes possibles

Notre analyse nous a conduits à établir différentes caractéristiques structurelles. L'une des premières caractéristiques de ces jeux vient de leur différence quant au nombre de joueurs qu'ils impliquent. Des jeux comme *MineCraft* ou *World of Warcraft* sont dits massifs et impliquent plusieurs centaines voire plusieurs milliers de joueurs simultanément. A l'inverse *Chess*, *HearthStone* ou encore *Portal 2* n'impliquent que deux joueurs qu'ils associent ou opposent. Certains jeux comme *BattleField* ou *League of Legends* prennent un nombre de joueurs variable mais borné (10 max pour *League of Legends*, 128 pour *Battlefield 3*). Cette première caractéristique structurelle révèle des informations intéressantes sur un scénario car un plus grand nombre de joueurs suggèrent des interactions plus nombreuses.

Une seconde caractéristique structurelle directement associée au nombre de joueurs émerge de notre liste de jeux. Certains jeux, comme *MineCraft* ou *DeadSpace 3* sont uniformes quant aux actions mis à disposition des joueurs (ils ont tous les mêmes « pouvoirs »), ce n'est pas le

[5] https://www.classcraft.com/fr/

cas de jeux comme *BattleField 3* ou *World of Warcraft* qui permettent aux joueurs de se spécialiser. Ainsi la possibilité ou non d'assumer des rôles différents dans un jeu constitue la seconde caractéristique établie par notre analyse.

Dans un jeu comme *HearthStone*, une carte posée ne peut être reprise (sauf cas particulier), ainsi chaque action effectuée par les joueurs dans un état donné ne peut être répétée à l'identique. Les joueurs de *Portal 2* ou *Chess* en revanche, sont libres de revenir sur leur pas et d'effectuer certaines actions autant de fois qu'ils le souhaitent. La possibilité ou non de répéter des actions en tant que joueurs constitue donc la troisième caractéristique structurelle que nous relevons.

Dans le cadre des jeux d'arcades comme *Pong* ou *Street Fighter*, le format implique que chaque session de jeu est indépendante des autres et se focalise sur les actions à disposition des joueurs dans un intervalle de temps donné. À l'inverse, le format des jeux scénarisés comme *DeadSpace 3* offrent la possibilité aux joueurs de suivre une histoire dans laquelle la progression du joueur est maintenue entre plusieurs sessions de jeu. Le jeu s'articule alors en fonction de l'avancée du joueur dans l'histoire dont la fin sert de motivation au joueur. Les jeux bacs à sable sont hybrides dans le sens où aucun scénario n'est fixé à l'avance similairement aux jeux d'arcade, mais où la progression est conservée d'une session à l'autre comme pour les jeux scénarisés. Dans ces jeux, les joueurs fixent eux-mêmes les objectifs et étapes qu'ils veulent franchir et atteindre. La différence entre ces trois genres repose sur la nature et le nombre de ces étapes. Pour les jeux d'arcades, le jeu lui-même est composé d'une unique étape fixe dans laquelle le joueur évolue à son gré. Pour les jeux scénarisés, les étapes sont fixées à l'avance et caractérisent la progression du joueur. Les étapes d'un jeu bac à sable sont mouvantes et réversibles au bon vouloir du joueur. L'agencement de ces étapes définit l'existence ou non de plusieurs chemins pour finir un jeu. La nature de cet agencement constitue notre quatrième caractéristique structurelle.

Beaucoup de ces jeux ont une fin prédéfinie : *DeadSpace 3, Chess, Europa Universalis IV, ClassCraft*, etc. Cependant, ce n'est pas le cas de tous les jeux. En effet, *MineCraft* ou encore *World of Warcraft* n'ont pas de fin à proprement parler. Dans le cadre de notre approche l'existence d'une fin pour un jeu est capitale (nous le verrons plus en détail dans le chapitre V). Ainsi, l'existence ou non d'une fin pour le scénario du jeu constitue notre cinquième et dernière caractéristique structurelle.

Les différentes caractéristiques structurelles que nous avons dégagées de cette analyse ne sont cependant pas les seules à prendre en compte pour rendre compte de la variété des scénarios de JSMJ. Notre travail porte sur l'étude des interactions. Or, l'ensemble des éléments (linéarité, nombre de joueurs, distribution des rôles, etc.) évoqués précédemment n'y font pas référence. Deux scénarios possédant des structures de jeux similaires peuvent susciter des interactions différentes. Pour avoir une idée de la manière dont les interactions émergent des scénarios de jeux sérieux, nous avons mené une seconde analyse axée sur le traitement de celles-ci dans les scénarios.

Les scénarios de *Voracy Fish* sont exclusivement « virtuels », l'intégralité des actions entreprises par les joueurs sont effectuées au sein de l'environnement virtuel / ludique du jeu.

D'autres encore, comme *Byte Battle* et *Learning Adventure*, ancrent une partie de leur scénario dans le réel (notamment dans la salle de classe), des interactions et des activités du scénario peuvent être effectuées en dehors du système informatique. Le scénario de *Byte Battle* que nous avons pris en compte fonctionne par exemple sous la forme d'un tournoi géré manuellement par l'enseignant et tous les apprenants ont accès à un classement public, inscrit sur un tableau noir. Dans *Learning Adventure*, il était demandé au joueur de fabriquer un Arduino et de chercher les pièces à l'intérieur même de la salle de classe. Ces deux exemples montrent que l'interaction entre apprenants peut se passer à l'extérieur du système « virtuel » du jeu. La présence d'éléments réels dans le scénario peut donc avoir un impact important sur les interactions et leur identification automatique.

Un autre aspect intéressant des scénarios de jeux sérieux repose sur le caractère obligatoire ou optionnel de l'interaction entre pairs (entre joueurs). Dans *ClassCraft* par exemple, l'interaction n'est pas obligatoire. Cette dernière peut être récompensée et son absence punie, mais il est possible d'atteindre la fin du jeu et de réussir celui-ci sans prendre en compte ses coéquipiers ou ses adversaires. A l'inverse, dans les scénarios de *Voracy Fish, ByteBattle* et *LearningAdventure*, il est impossible de progresser sans interagir. Cet aspect peut être particulièrement intéressant pour un enseignant cherchant à favoriser l'interaction entre les apprenants. Ce caractère obligatoire ou optionnel des interactions devrait être un paramètre de la scénarisation et impacter la détection automatique des interactions. Cette liberté d'action face aux interactions dans *ClassCraft* est notamment possible en raison de la nature asynchrone du jeu. Un joueur utilisant un bonus « collaboratif » peut ne pas avoir de réaction de la part de ses camarades, ou en obtenir une bien plus tard. Cet asynchronisme des interactions peut rendre difficile leur détection et indique un aspect supplémentaire à prendre en compte par un système dont l'objectif est de traiter les interactions dans des scénarios complexes. Dans des scénarios comme celui de *Byte Battle* en revanche, outre leur aspect « obligatoire », les interactions sont instantanées. Par exemple, une action compétitive a des effets immédiats sur l'autre joueur (destruction d'une unité, victoire, montée du score, etc.).

Pour finir, les jeux sérieux varient également selon les interactions qu'ils peuvent susciter. Dans le cas de *ClassCraft*, la coopération y est naturellement présente à travers les différents bonus que peuvent prendre les joueurs. La compétition y est en revanche plus sous-jacente et s'exprime à travers l'existence de différents groupes d'élèves aux résultats disparates. Dans *Byte Battle*, la compétition s'avère omniprésente par la présence des éléments suivants : tournoi, score, vainqueur, etc. De même pour *Voracy Fish* où seule la survie du poisson compte, ce qui pousse à pourchasser les cibles des autres joueurs, ainsi que les joueurs eux-mêmes. Nous retrouvons quelque chose de plus nuancé et équilibré dans le scénario de *Learning Adventure*, où les apprenants d'un même groupe sont amenés à collaborer pour parvenir à leur fin avant les autres groupes avec qui ils se retrouvent naturellement en compétition.

## III.1.b. Caractéristiques des scénarios de JSMJ

L'analyse de ces 17 jeux nous a conduit à établir deux classes différentes de caractéristiques. D'abord, la classe des cinq caractéristiques structurelles que nous avons présentées dans la section III.1.a (les aspects mentionnés ici sont deux de la grille d'analyse en section III.1.a) :

1. Nombre de joueurs prenant part au scénario multi-joueurs étudié. (Aspect 1)
2. Différenciation des rôles : cette caractéristique discrimine entre les scénarios où tous les joueurs réalisent les mêmes actions (et où ils ont les mêmes ressources initiales) et les scénarios où les joueurs peuvent effectuer des actions spécifiques à leur rôle (Aspect 1, 2),
3. Répétabilité des actions : cette caractéristique caractérise les scénarios où il est possible d'effectuer une action plusieurs fois (Aspect 3),
4. Linéarité du scénario : un scénario est dit multilinéaire s'il existe plusieurs séquences d'activités distinctes permettant d'atteindre la fin de celui-ci. Dans le cas contraire, le scénario est dit linéaire (Aspect 4 et 5),
5. Finalité du scénario : définit l'existence ou non de plusieurs fins (états finaux) possibles pour un scénario (Aspect 6).

La première classe faisant référence aux caractéristiques structurelles du scénario, la deuxième classe de caractéristiques que nous avons dégagées est quant à elle dédiée aux traitements des interactions. Ces caractéristiques sont au nombre de quatre :

1. Niveau d'intégration du scénario dans l'environnement ludique : caractérise la présence ou non d'éléments extérieurs au système informatique (tableau noir, tchat, etc.) dans le scénario. La présence de ces éléments peut induire des interactions entre pairs ayant lieu en dehors du système mais qui sont à l'intérieur du scénario,
2. Synchronisation des interactions : une interaction est dite synchrone si l'ensemble des joueurs y prenant part agissent simultanément dans le cadre de celle-ci,
3. Degré de liberté face aux interactions : caractérise le fait qu'une interaction soit obligatoire ou non à la réalisation du scénario,
4. Variété des interactions : définit la liste des différentes interactions (coopération, compétition, collaboration, etc..) pouvant avoir lieu dans un scénario.

*MoPPliq* (Marne, 2014)) que nous avons vu en section II.3.d permet de répondre aux caractéristiques structurelles 3 et 4 en décomposant son scénario en activités reliées entre elles par un système d'entrées / sorties. Cette décomposition du scénario en éléments plus simples nous a conduits à établir une première contrainte que notre modélisation de JSMJ doit respecter : la modularité du scénario.

La deuxième contrainte que nous avons établie est la nécessité de gérer l'aspect multi-joueurs des scénarios de JSMJ, avec la possibilité d'avoir plusieurs rôles et plusieurs équipes. Ce critère étant là pour répondre aux caractéristiques structurelles 1 et 2, ainsi qu'à la caractéristique 2 sur les interactions. Cette contrainte nous a notamment été inspirée par la théorie de l'activité (section II.3.a).

Des travaux comme STRIPS (section II.4.a) ou les réseaux de Petri (section II.4.b) basent la faisabilité d'une action sur les ressources à disposition de l'utilisateur. Ces ressources permettent donc de définir ce qu'un apprenant peut ou ne peut pas faire et ne sont pas limitées aux ressources virtuelles. De plus, l'une de nos hypothèses de travail évoquée en

introduction est que la structure du *game design* d'un scénario induit certaines interactions. Un jeu où il n'y aurait pas assez de ressources pour tout le monde par exemple serait par essence compétitif. Ainsi, en étant à même de prendre en charge la gestion des ressources, notre modèle devrait être en mesure de satisfaire la caractéristique structurelle 3 et les caractéristiques sur les interactions 1, 3 (partiellement) et 4. La caractéristique 3 sur les interactions implique de pouvoir déterminer si une interaction est obligatoire ou non à l'échelle du scénario. Or la gestion des ressources permet de déterminer les activités à disposition des joueurs et les interactions y ayant lieu. Pour établir les séquences d'activités permettant la réalisation du jeu (et donc les interactions effectuables), il est cependant nécessaire de connaitre la fin ou les fins de celui-ci conformément à ce qu'indique la caractéristique structurelle 5.

Les activités de *MoPPliq* sont reliées à des entités symbolisant le début et la fin du scénario, traitant ainsi la caractéristique structurelle 5 et finissant d'établir les critères à suivre pour satisfaire la caractéristique 3 sur les interactions. En s'appuyant sur nos neuf caractéristiques et sur l'état de l'art (chapitre II), nous avons donc établi une liste de 4 contraintes que notre modélisation de scénarios de JSMJ devrait respecter :

- La modularité : possibilité de décomposer un scénario en scénarios moins complexes,
- La gestion de l'aspect multi-joueurs, avec la possibilité d'avoir plusieurs rôles et plusieurs équipes,
- La possibilité de gérer des ressources (des objets de jeu mais aussi des connaissances ou compétences) ce qui permet d'identifier certains éléments de *game design* nécessaires à la réalisation de certaines interactions,
- La possibilité de définir un ou plusieurs états finaux du scénario afin d'être sûr que le scénario ait une fin.

Ces contraintes nous ont permis d'orienter nos choix de modélisation de scénarios de JSMJ. Nous nous baserons sur cette modélisation pour détecter automatiquement des interactions pouvant émerger d'un scénario de JSMJ. La solution que nous avons choisie pour formaliser cette modélisation consiste à développer une ontologie qui conceptualise formellement le modèle. Tous les éléments de notre modèle sont aisément représentables par les concepts d'une ontologie (classes, relations, axiomes, restrictions, etc.). Un scénario de JSMJ représente ainsi la base de connaissances (un ensemble d'assertions) qui instancie les concepts de l'ontologie.

L'approche que nous avons utilisée pour notre modèle se divise en quatre concepts clés. Les trois premiers concepts correspondent aux différents niveaux de granularité de notre modèle à savoir : le scénario, les activités utilisées par le scénario et les rôles prototypiques composant les activités. Un quatrième concept permet de modéliser les joueurs.

## III.2. Scénario

Nous représentons un scénario comme une entité modulaire décomposable en scénarios de granularité plus fine que nous appelons scénarios élémentaires. Un scénario élémentaire est une structure composée de deux éléments : un ensemble d'activités et une activité étape (voir

Figure 7). L'activité étape correspond à l'activité qu'il faut exécuter pour parvenir à la fin du scénario élémentaire. L'ensemble des activités du scénario élémentaire, quant à lui, correspond à l'ensemble des activités que les apprenants peuvent potentiellement utiliser en vue de réaliser l'activité étape du scénario élémentaire. Un lien de précédence explicite existe entre l'ensemble des activités et l'activité étape d'un même scénario élémentaire. Il n'est en revanche pas nécessaire de définir de liens de précédence entre les différentes activités de l'ensemble. En effet, comme nous le présenterons plus en détail dans la section III.3, la dynamique du scénario vient des transformations opérées par les activités sur leur environnement (cf. contrainte sur la gestion des ressources).

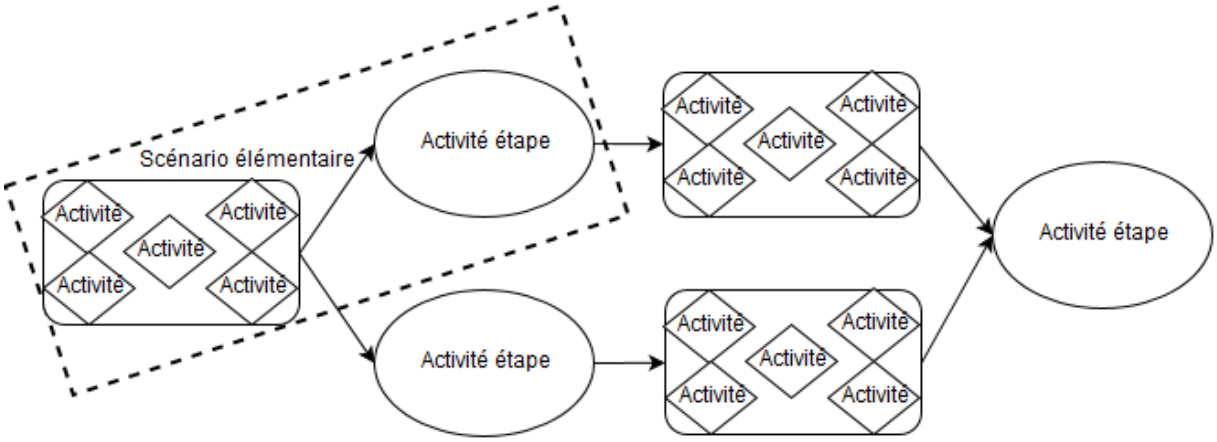


*Figure 7 : Scénario composé de scénario élémentaires*

L'activité étape d'un scénario élémentaire SE marque un point de passage important au sein du scénario global dont il est un composant. Dans le cas où le SE se trouverait à la fin du scénario, son activité étape marquerait la fin du scénario lui-même. Pour définir les différents liens de précédences reliant les activités d'un scénario, nous avons défini un ensemble de règles :

- Un lien relie un ensemble d'activités à une activité étape.
- Un ensemble d'activités peut précéder plusieurs activités étapes, ce qui permet de prévoir plusieurs états finaux pour une étape de jeu.
- Une activité étape ne peut précéder qu'un seul ensemble d'activités.
- Une activité étape doit être précédée d'au moins un ensemble d'activités

Les liens de précédence ainsi définis fournissent différentes informations sur un scénario et ses activités étapes. Premièrement, lorsqu'il est possible d'atteindre plusieurs activités étapes à partir d'un ensemble (comme c'est le cas dans la Figure 7), cela indique que le scénario contient un embranchement et n'est donc pas linéaire. Deuxièmement, lorsqu'une activité étape ne précède aucun ensemble d'activité, c'est qu'il s'agit d'une activité finale du scénario dans son ensemble et pas seulement du scénario élémentaire auquel elle se rapporte. En effet, il est impossible de sortir d'une activité étape ne précédant aucun ensemble.

La flexibilité apportée par ce modèle de description de scénarios de JSMJ permet de satisfaire le besoin de modularité d'un scénario de JSMJ (scénario linéaire ou non, activité répétable ou non, avec cycle, etc.). La question de l'état final est quant à elle traitée à deux niveaux :

- Au niveau local : les activités étapes qui précèdent des ensembles d'activités font office d'états intermédiaires obligatoires dans le scénario,
- Au niveau global : les activités étapes qui ne précèdent aucun ensemble d'activités font office d'états finaux pour le scénario dans son ensemble.

## III.3. Décomposition d'un scénario JSMJ en Activités

Dans un jeu sérieux dédié à l'apprentissage, les joueurs réalisent des actions leur permettant de progresser dans le jeu, travailler des compétences ou réaliser des objectifs (Djaouti et al., 2011). Nous pouvons concevoir un scénario comme une suite d'actions individuelles réalisées par les joueurs. Toutefois, le niveau de granularité d'une telle modélisation n'est pas adapté pour permettre l'observation des interactions. En effet, des interactions comme la compétition ou la collaboration sont difficiles à observer en analysant une seule action isolée d'un joueur mais émergent plutôt d'une suite d'actions réalisées par plusieurs joueurs impliqués dans l'interaction. Ainsi, il paraît nécessaire de regrouper les actions permettant d'observer des interactions entre pairs dans des entités de granularité moins fine ; c'est ce que nous appelons « Activité ». Ce sont ces activités qui composent les Scénarios élémentaires (que ce soit au niveau des activités étapes ou des ensembles d'activités les précédant).

La première génération de la théorie de l'activité, évoquée dans l'état de l'art, stipule qu'une activité doit être composée de trois éléments identifiables :

- Un sujet : l'individu prenant part à l'activité
- Un objet : le but de l'activité, la raison d'être de l'activité pour le sujet
- Des outils : l'ensemble des ressources à la disposition du sujet pour réaliser l'objet

Dans le cadre de nos activités, nous avons donc les trois éléments suivants :

- Le *sujet* grâce à la représentation des rôles que les joueurs endossent dans les activités.
- *L'objet* de notre activité se réfère aux objectifs des joueurs qui guident leurs actions dans l'activité et déterminent les interactions pouvant émerger de celle-ci,
- *L'outil* regroupe l'ensemble des ressources consommées ou produites par les joueurs à l'issues de leurs actions au sein de l'activité.

Voici un exemple pour comprendre ce que ces activités représentent à l'échelle d'un scénario de JSMJ. Prenons le cas du dilemme du prisonnier. Dans cet exemple, deux prisonniers sont interrogés dans le cadre d'une enquête policière sur un braquage. Le problème des policiers, est qu'ils ont assez peu d'indices et doivent faire avouer les prisonniers pour les mettre en prison. Les deux prisonniers ont deux possibilités : avouer ou se taire. Si les deux avouent, ils sont condamnés chacun à 5 ans de prison. Si l'un avoue et pas l'autre, celui qui n'a rien dit est condamné à 10 ans de prison. Pour finir, si aucun n'avoue, les deux sont libres au bout d'un an. Une représentation habituelle de ce problème se fait sous la forme d'une matrice carré 2X2.

Dans ce jeu chaque joueur n'a à sa disposition que 2 actions individuelles, à savoir : « se taire » ou « avouer ».

*Tableau 1 : Dilemme du Prisonnier*

| Prisonnier 1 \ Prisonnier 2 | Se Taire | Avouer |
|---|---|---|
| Se Taire | P1 = 1 an, P2 = 1 an | P1 = 10 ans, P2 = 0 an |
| Avouer | P1 = 0 an, P2 = 10 ans | P1 = 5 ans, P2 = 5 ans |

Lorsqu'un prisonnier se tait, il prend une approche coopérative du problème. Lorsqu'il avoue, il prend une approche compétitive de celui-ci. Or, il existe un cas où l'un avoue, et l'autre se tait. Dans ce cas précis, nous avons donc une action individuelle supposément coopérative qui débouche tout de même sur une situation de conflit. Dans le cas où les deux se taisent en revanche, nous avons de la coopération. L'action individuelle ne permet donc pas de discerner l'interaction émergeant du comportement des deux prisonniers, il est nécessaire d'avoir toute la séquence.

En choisissant un niveau de granularité plus élevé, nous pouvons obtenir des informations relatives aux interactions. Le jeu décrit quatre sorties possibles dont deux sont symétriques (l'un avoue, l'autre se tait). Nous pouvons décrire ces sorties à l'aide des trois situations suivantes :

1. Situation 1 : les deux prisonniers ont pour objectifs d'être libre mais préfèrent se taire. Ils consomment tous les deux la ressource « Libre » indiquant leur liberté et produisent la ressource « 1 an d'emprisonnement ».
2. Situation 2 : les deux prisonniers ont pour objectifs d'être libre mais l'un préfère parler. Le prisonnier se taisant assume un rôle qui consomme la ressource « Libre » indiquant sa liberté et produit la ressource « 10 ans d'emprisonnement ». L'autre assume un rôle ne consommant et ne produisant rien. Il conserve donc sa liberté.
3. Situation 3 : les deux prisonniers ont pour objectifs d'être libre mais préfèrent parler. Ils consomment tous les deux la ressource « Libre » indiquant leur liberté et produisent la ressource « 5 ans d'emprisonnement ».

La coopération apparaît clairement dans la première situation et le conflit dans les deux autres. Dans la situation 2, seul un joueur est satisfait, et dans la situation 3, aucun des deux n'est satisfait. Ces trois situations peuvent être représentées à l'aide d'activités (une par situation). Celles-ci sont alors l'agrégation de nos actions. Il est ainsi possible d'y extraire des informations sur les interactions mises en œuvre par les prisonniers.

Par conséquent, nous avons décidé de proposer une modélisation flexible, laissant le choix à l'enseignant ou au concepteur du scénario de décider du degré de granularité des activités composant un scénario de JSMJ. Ainsi, une activité peut être composée d'une seule action joueur mais peut également représenter l'intégralité du scénario. L'activité constitue la brique élémentaire du modèle. Elle doit correspondre à une forme de défi pour les joueurs (gagner

une course, résoudre une formule, mobiliser des connaissances, etc.). Une activité correspond à l'agrégation d'une ou plusieurs actions en fonction des critères suivants :

- Les actions agrégées impliquent le même ensemble de joueurs.
- Elles sont entreprises dans un intervalle continu de temps
- Elles suivent la même visée / le même objectif

Illustrons ces règles avec un exemple. Dans le jeu de cartes « la Bataille », les deux joueurs possèdent chacun un paquet de cartes. Le but du jeu est de récupérer toutes les cartes de l'adversaire. Pour ce faire, chaque joueur tire une carte de son paquet puis la révèle. Nous comparons ensuite les valeurs des deux cartes et la plus haute l'emporte (il existe des règles spéciales en cas d'égalité que nous ignorerons ici). La représentation d'une manche de la Bataille passe donc par trois actions individuelles :

- Tirer une carte
- Présenter la carte
- Récupérer les cartes

La suite d'actions réalisées par les joueurs constitue une unique activité : faire une manche de la Bataille. Les deux acteurs y sont représentés, la continuité de l'intervalle de temps est respectée, et les actions ont bel et bien été entreprises dans l'objectif commun de faire une manche.

Pour faire écho au problème de granularité précédemment évoqué, il est également possible de représenter chacune des actions entreprises par les joueurs comme des activités individuelles manipulant l'environnement de jeu et les ressources à leur disposition. Il est cependant complexe d'extraire des interactions de cette suite d'actions. En effet, nous devrions alors simuler les conséquences éventuelles de chaque action par rapport aux autres joueurs, mais aussi prendre en compte l'intervalle temporel de la séquence. La représentation de ces actions avec une unique activité de granularité supérieure, nous fournit ces informations de manière plus aisée. Ceci montre, encore une fois, que le choix du niveau de granularité des activités est le point critique de notre approche : des interactions ayant été extraites pour un choix de granularité donné, auraient pu ne pas être détectées pour un degré de granularité plus faible ou plus gros. Le Tableau 2 présenté en section III.4 donne un aperçu du niveau de granularité que nous aurions adopté à savoir modéliser chaque manche individuellement. Dans cette modélisation, seule l'issue importe : le vainqueur produit deux cartes dans son inventaire conformément à ses objectifs au contraire du perdant qui en consomme une. Ainsi chaque manche est compétitive et est nécessaire à la complétion du jeu : un joueur possède les 56 cartes.

Le schéma présenté en Figure 8 représente les éléments du modèle formalisés sous forme d'une ontologie appelée *MultiPlayer Learning Game Ontology* (MPLGO). Les quatre éléments du scénario que nous avons identifiés comme importants à la détection des interactions sont représentés à l'aide d'un code couleur (présenté dans la légende en haut à droite de la Figure 8).

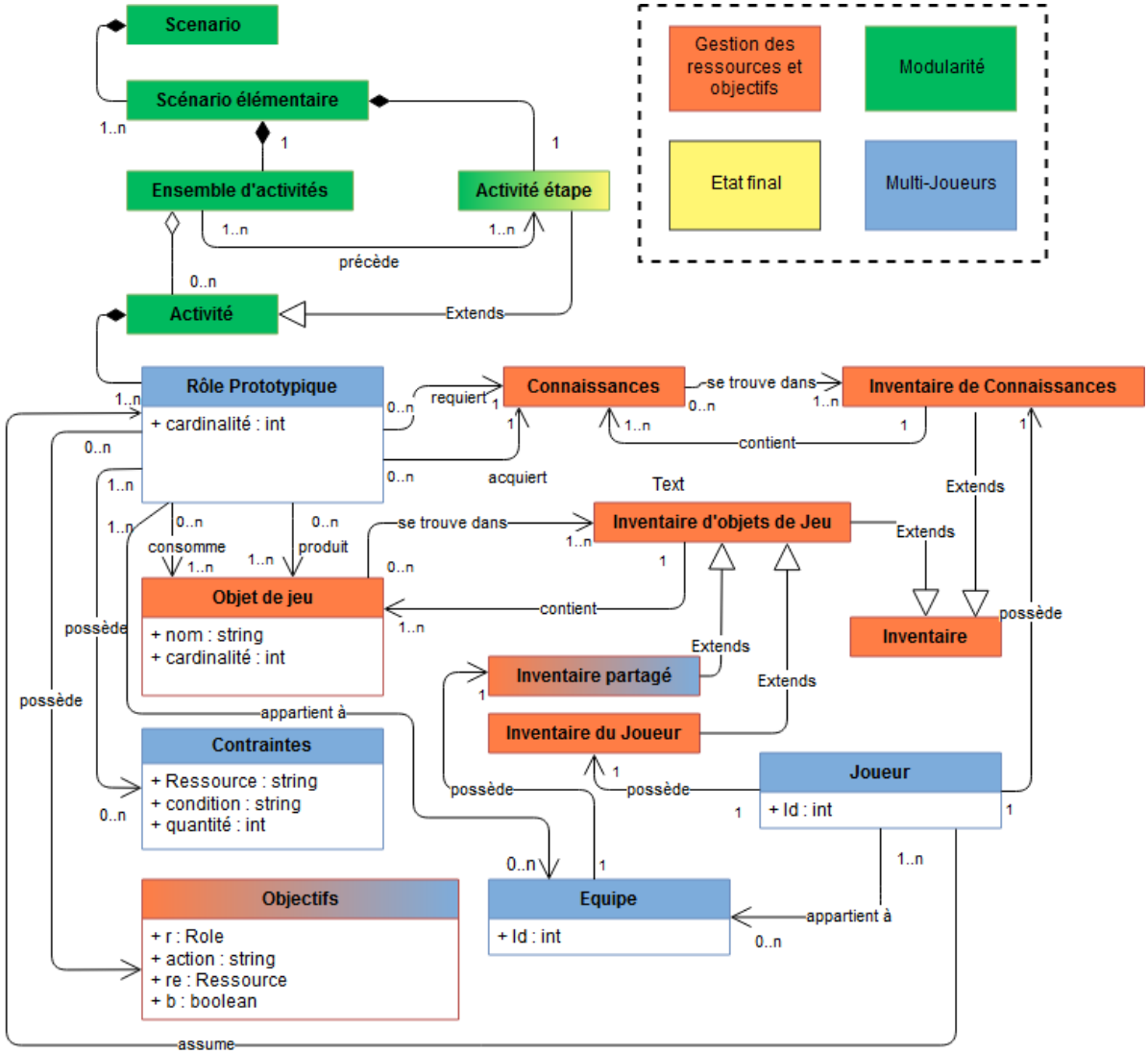


*Figure 8 : Ontologie MPLGO*

L'activité est l'élément central du modèle, elle permet la liaison entre le scénario lui-même et les rôles prototypiques auxquels se lient les joueurs. Une activité est constituée des différents rôles prototypiques qui lui sont associés.

# III.4. Rôles prototypiques d'une activité de JSMJ

Dans la littérature, le terme « rôle » est communément utilisé pour indiquer aux joueurs les comportements qu'ils sont censés avoir au sein d'une activité. Endosser un rôle implique donc adopter un comportement dans l'activité. Dans notre cas, nous parlons plutôt de rôles prototypiques pour marquer la différence avec l'acception commune. Ainsi un rôle prototypique est défini par des objectifs et l'utilisation d'objets de jeu et de connaissances. Si un joueur atteint un objectif en utilisant les objets et les connaissances définis dans un rôle prototypique alors nous considérons que le joueur a incarné ce rôle. Autrement dit, nous parlons de rôle lorsque le joueur incarne consciemment un comportement dans une activité (le joueur choisit d'incarner un rôle ce qui oriente son comportement dans une direction) et nous parlons de rôle prototypique lorsque le joueur réalise des actions de jeu sans avoir

connaissance du rôle explicite qu'il est en train de jouer (c'est le comportement du joueur qui permet au système de lui associer un rôle prototypique). C'est cet aspect sur la conscience du joueur qui distingue pour nous la notion de rôle de la notion de rôle prototypique.

Afin de représenter les actions entreprises par les joueurs, nos rôles prototypiques décrivent l'impact qu'auront les joueurs sur leurs inventaires et environnement de jeu. Ces modifications sont de deux types : consommation et production.

- Un objet de jeu consommé par un rôle prototypique est un prérequis au commencement de l'activité, il sera alors retiré de l'environnement ou de l'inventaire du joueur ayant endossé ce rôle prototypique.
- Un objet de jeu produit par un rôle prototypique est ajouté à l'environnement ou à l'inventaire du joueur qui a pris ce rôle prototypique.

Il est cependant utile de pouvoir modéliser qu'une ressource soit utilisée par le joueur (sans être défaussée). Cette ressource est présente dans l'environnement ou dans l'inventaire du joueur avant et après la réalisation de l'activité. Dans ce cas, nous considérons que la ressource est successivement consommée puis produite par le rôle prototypique.

Une autre fonction des rôles prototypiques consiste à décrire la gestion des connaissances d'un joueur dans le cadre de l'activité concernée. Cette gestion est caractérisée par deux aspects : les connaissances requises et les connaissances ciblées.

- Les connaissances requises spécifient les connaissances qui doivent être maîtrisées pour que le comportement d'un joueur puisse être associé à un rôle prototypique dans l'activité considérée. Néanmoins elles ne sont pas bloquantes et un joueur n'ayant pas les connaissances requises peut tout de même s'engager dans l'activité. Dans ce cas le joueur risque de se trouver en difficulté pour réaliser l'activité.
- Les connaissances ciblées par un rôle prototypique indiquent les connaissances sur lesquelles portent l'activité.

Les objets de jeu et les connaissances ne sont pas les seules composantes associées à un rôle prototypique. Les buts des joueurs dans le cadre d'une activité y sont également modélisés et associés à un rôle prototypique. Ainsi, les joueurs assumant des rôles prototypiques dans une activité peuvent chercher à effectuer certaines modifications de l'environnement (consommation et production de ressources par exemple), ou encore à empêcher d'autres joueurs de procéder à ces modifications.

Enfin, un ensemble de contraintes peut également être associé à un rôle. Ces contraintes sont de deux natures. Les premières sont des contraintes d'équipes. Le but de ces contraintes est d'établir si les joueurs désirant s'impliquer dans une activité peuvent le faire au vu de la composition des équipes de joueurs. Par exemple, des joueurs pourraient n'être autorisés à prendre part à une activité donnée que s'ils appartiennent à la même équipe et non pas à des équipes adverses. Les secondes sont des contraintes d'inventaires. Ces contraintes (qui se distinguent des ressources consommées) permettent de vérifier la présence ou l'absence de ressources dans des quantités données dans les inventaires des joueurs.

En plus des contraintes définies pour chaque rôle prototypique et de sa transformation de ressources, le rôle prototypique possède une cardinalité qui participe à la définition des règles

d'attribution des rôles prototypiques dans le cadre d'une activité. Ces règles sont les suivantes :

- Un joueur ne peut être associé qu'à un seul rôle prototypique dans une activité en cours. Un joueur peut être associé à un rôle prototypique différent s'il refait la même activité,
- La cardinalité d'un rôle prototypique est toujours supérieure ou égale à 1. Nous notons que la cardinalité d'un rôle prototypique est le nombre de joueurs pouvant lui être associés lors de la réalisation d'une activité,
- Une activité ne peut s'exécuter que si l'ensemble des rôles prototypiques qui la composent sont assumés par un nombre de joueurs correspondant aux cardinalités.

Cet ensemble de règles permet d'assurer que lorsque la somme des cardinalités de tous les rôles prototypiques d'une activité est supérieure ou égale à 2, nous soyons en présence d'une activité multi-joueurs. La possibilité d'assurer qu'une activité soit multi-joueurs est particulièrement important pour pouvoir détecter des interactions dans celle-ci.

Dans la Figure 8, nous constatons que les joueurs ont à leur disposition un inventaire d'objets de jeu et un inventaire de connaissances. Ces deux inventaires permettent de représenter les ressources auxquelles seul le joueur concerné a accès. Cependant, ce ne sont pas les seuls inventaires auxquels il a accès. En effet, il peut accéder à un certain nombre d'inventaires qu'il partage avec d'autres joueurs. Ces inventaires portent le nom d'inventaires partagés. L'objectif d'un inventaire partagé est d'indiquer les objets de jeu accessibles aux membres d'un sous-ensemble des joueurs (typiquement une équipe). Ces inventaires sont donc associés à des équipes. Par définition, il n'y a pas d'inventaires partagés pour les connaissances. Ces notions donnent un éclairage sur la manière avec laquelle sont gérées les actions multi-joueurs et leurs impacts dans le cadre d'une activité.

En début de section III.3, nous évoquions la théorie de l'activité dont nous nous sommes inspirés. Elle n'est cependant pas la seule à être représentée ici. Une similarité importante avec les systèmes concurrents peut également être notée. En effet, ces derniers recréent la dynamique d'un système en se basant sur des relations de consommation / production de ressources. Il devient alors évident que la transformation des objets de jeu (et l'acquisition des connaissances dans une moindre mesure) reproduit une dynamique similaire à l'échelle d'une activité dans un premier temps, puis à l'échelle du scénario. Cet aspect est particulièrement important pour le fonctionnement de l'algorithme d'analyse de scénario que nous présentons en chapitre V.

*Tableau 2 : Exemple d'activité : Faire une Manche de Bataille*

| Faire une Manche | | |
|---|---|---|
| Rôles prototypiques | Description | Caractéristiques du rôle |
| Vainqueur | Le Vainqueur tire la carte la plus haute et récupère la carte de son adversaire | Le Vainqueur a une cardinalité de 1 |
| | | Le Vainqueur appartient à l'équipe A |
| | | Le Vainqueur consomme une Carte dans son inventaire personnel |
| | | Le Vainqueur produit deux Cartes dans son inventaire personnel |
| | | Le Vainqueur ne requiert aucune connaissances |
| | | Le Vainqueur a pour objectif de produire deux Cartes |
| Perdant | Le Perdant tire la carte la plus faible et la perd | Le Perdant a une cardinalité de 1 |
| | | Le Perdant appartient à l'équipe B (≠A) |
| | | Le Perdant consomme une Carte dans son inventaire personnel |
| | | Le Perdant ne produit rien |
| | | Le Perdant ne requiert aucune connaissances. |
| | | Le Perdant a pour objectif de produire deux Cartes |

En modélisant l'exemple de la Bataille décrit dans la section III.3, nous obtenons le Tableau 2. Dans cet exemple nous apercevons 2 rôles prototypiques. Le rôle « Vainqueur » qui remporte la manche, et le rôle « Perdant » qui la perd. Pour pouvoir faire une manche, chaque joueur doit appartenir à des équipes différentes ici A et B. Ils doivent également avoir une carte dans leur inventaire. Cette carte est consommée par l'activité pour indiquer sa nécessité. Une fois la partie finie, le Vainqueur récupère les deux cartes. Son rôle produit donc 2 fois la ressource carte au contraire du Perdant. Pour finir, l'objectif des deux joueurs est de gagner, ce qui est symbolisé par la production des deux cartes. Or seul le Vainqueur produit ces deux cartes. Nous percevons à travers cet exemple la signification que nous donnons à un rôle

prototypique : c'est un rôle qui n'est pas endossé consciemment par un joueur ; un joueur ne prend pas consciemment le rôle de « Perdant » mais c'est une conséquence de son comportement au sein de l'activité.

## III.5. Joueurs

Un joueur, du point de vue du jeu, est un agent capable d'effectuer différentes actions et de transformer (consommer et produire) des ressources du jeu. Le joueur peut ainsi être résumé à l'aide des éléments suivants :

- Son inventaire d'objets de jeu : les objets possédés uniquement par le joueur.
- Son inventaire de connaissances : les connaissances qui ont été travaillées par le joueur.
- Les équipes auxquels il appartient. Chaque équipe étant dotée d'un inventaire partagé caractérisant les objets de jeu à disposition de tous ses membres.

L'ensemble de ces éléments permettent de déterminer les rôles prototypiques auxquels un joueur peut correspondre, puisque ces derniers sont caractérisés par trois paramètres, tous couverts par les éléments précédemment décrits :

- Les ressources à disposition du joueur
- La cardinalité du rôle prototypique
- Les contraintes d'équipe

La modélisation d'un joueur passe donc par la description des objets et connaissances présents dans ses inventaires, ainsi que par son association aux différentes équipes auxquelles il doit appartenir.

## III.6. Synthèse

L'approche que nous avons choisie nécessite la modélisation et la formalisation des scénarios de JSMJ. Ceux-ci sont variés, tant d'un point de vue structurel que du point de vue des interactions. Afin de cerner les différentes caractéristiques responsables de cette variété, nous avons analysé 17 jeux (dont 4 jeux sérieux). Cette analyse nous a permis d'identifier une liste de 9 caractéristiques (les 5 premières sont structurelles, les 4 suivantes sont relatives aux interactions) :

1. nombre de joueurs impliqués
2. différenciation des rôles
3. répétabilité des actions
4. linéarité du scénario
5. finalité du scénario
6. niveau d'intégration du scénario dans l'environnement vidéoludique
7. synchronisation des interactions
8. degré de liberté quant aux interactions
9. variété des interactions

En nous appuyant sur ces caractéristiques, nous avons pu établir 4 contraintes que notre modèle formel doit satisfaire pour pouvoir traiter ces 9 caractéristiques. En voici la liste :

- La modularité : possibilité de décomposer un scénario en scénarios moins complexes.
- La gestion de l'aspect multi-joueurs, avec la possibilité d'avoir plusieurs rôles et plusieurs équipes
- La possibilité de gérer des ressources (des objets de jeu mais aussi des connaissances ou compétences) ce qui permet d'identifier certains éléments de *game design* nécessaires à la réalisation de certaines interactions.
- La possibilité de définir un état final afin d'être sûr que le jeu ait une fin.

N'ayant pas trouvé de modèle répondant à nos attentes vis-à-vis de ces contraintes, nous avons décidé de proposer le nôtre. La structure du modèle à laquelle nous sommes arrivés s'articule autour de plusieurs concepts :

- Le scénario élémentaire, pour plus de modularité et permettant de représenter les différents embranchements offerts par un scénario.
- Les activités articulées autour des rôles prototypiques les composant, servent à agréger des actions individuelles en une entité multi-joueurs.
- Les rôles prototypiques permettent de décrire pleinement les comportements attendus des joueurs, leurs impacts sur l'environnement, et les contraintes associées.
- Les joueurs, représentés par leurs inventaires et équipes, peuvent assumer des rôles prototypiques, manipuler et posséder des ressources. Ce sont les acteurs du scénario.

La modélisation du scénario à l'aide de scénarios élémentaires nous permet d'assurer une certaine forme de modularité en permettant de représenter des scénarios linéaires et non-linéaires.

La description des activités composant un scénario élémentaire, permet de décrire des situations multi-joueurs dans lesquelles les relations que chaque joueur entretient vis-à-vis des autres sont définies. Elle établit également la liste exhaustive des transformations opérées par les joueurs sur leur environnement de jeu. Pour finir, elle nous informe à la fois des objectifs des joueurs au début de l'activité, et de la satisfaction ou non de celles-ci, une fois l'activité réalisée.

La consommation et la production d'objets de jeu, nous fournit également des informations implicites sur l'ordonnancement des activités. En effet, si une ressource est produite par une activité, celle-ci précède alors toutes les activités consommant ladite ressource.

Ainsi si nous reprenons les quatre contraintes de modélisation du scénario que nous avons évoqués plus tôt à savoir la modularité, la présence d'un ou plusieurs états finaux, la gestion des ressources et la gestion des aspect multi-joueurs, le modèle d'activités que nous proposons intègre bien ces contraintes.

La modélisation à laquelle nous sommes parvenus a par la suite été formalisée à l'aide d'une ontologie appelée *MultiPlayer Learning Game Ontology* (MPLGO). Les différents tests et expérimentations ayant été menés pour mettre à l'épreuve ce modèle sont décrits dans le chapitre VI.

# IV. Traitement et représentation des interactions multi-joueurs

Le positionnement de ce chapitre par rapport à ceux de la section 2 sur les contributions est présenté dans la Figure 6 (page 24).

L'approche proposée dans le cadre de ce travail de thèse consiste à analyser des scénarios de JSMJ dans le but de détecter automatiquement des interactions susceptibles d'émerger de ces scénarios. Cette détection automatique des interactions nécessite deux modélisations :

- La modélisation des scénarios et des activités les composant,
- La modélisation des interactions au sein des activités.

Ce chapitre a pour vocation à présenter en détail notre approche pour la modélisation et la formalisation des interactions. Il se divise en trois sections. La section IV.1 contextualise et présente notre approche qui consiste à décrire les interactions comme étant des combinaisons de propriétés moins abstraites. Dans la section IV.2, nous présentons ces propriétés et la manière dont elles sont extraites de la modélisation des activités (cf. chapitre III). Pour finir, la section IV.3 détaille le processus mis en place pour détecter une propriété au sein d'une séquence et ensuite d'un scénario.

## IV.1. Contexte

Les interactions sont des phénomènes sociaux complexes touchant de nombreux domaines (EIAH, théorie des jeux, systèmes multi-agent, etc…). Nous l'avons vu dans la section II.2, de nombreuses tentatives de définitions ont été entreprises (Dillenbourg, 1999; Kozar, 2010; Roschelle & Teasley, 1995). Cependant aucune de ces tentatives n'a permis de faire le consensus entre les différents chercheurs s'intéressant aux interactions entre pairs.

Ainsi, comment pouvons-nous modéliser une interaction en l'absence de consensus sur sa définition ?

Répondre à cette question était nécessaire à notre approche et a constitué l'un des objectifs de cette thèse. En analysant les différentes définitions, il nous est apparu qu'en dépit des divergences certaines parties des définitions des interactions étaient communes. Par exemple, nous avons constaté que bien que les travaux en EIAH et en théorie des jeux avaient des définitions différentes de la collaboration (voir section II.2.b), (la collaboration en EIAH devant impliquer une construction commune des connaissances, là où la collaboration en théorie des jeux nécessitait une réussite ou un échec commun des participants) ces deux définitions ont en commun la nécessité d'un travail conjoint des protagonistes de l'interaction.

Ainsi, nous avons décidé de ne plus chercher à trouver des définitions consensuelles pour les interactions entre pairs mais plutôt d'extraire des propriétés de bas niveau, issues de la modélisation des activités, et qui, en les combinant pourraient décrire des interactions de plus haut niveau comme la collaboration, la coopération ou le conflit. Notre but étant alors de

permettre à des enseignants ou à des concepteurs de scénarios de combiner à leur guise des propriétés pour spécifier une interaction particulière. Notre question de recherche s'est donc transformée ainsi : quelles propriétés peut-on extraire de la modélisation des activités ?

## IV.2. Formalisation et détection des propriétés des interactions

Dans la section II.2, nous avons pu observer plusieurs définitions de la collaboration, de la coopération et de la compétition. Nous présentons ici un *framework* contenant actuellement 5 propriétés qui peuvent servir à modéliser ces trois interactions.

1. La Validation Commune : indique que les apprenants se sont coordonnés avant de finir l'activité,
2. La Construction Commune de Connaissances : indique que les apprenants ont construit ensemble la solution leur permettant de résoudre l'activité,
3. Le Conflit Interne : indique qu'il y a un conflit entre les membres d'une même équipe par rapport à la gestion des ressources,
4. Le Conflit Externe : indique une opposition de résultats entre deux membres d'équipes différentes au sein d'une activité,
5. L'Individualisme : indique qu'un apprenant joue seul et n'interagit pas avec les autres dans le cadre de l'activité.

Comme évoqué précédemment, les interactions que nous cherchons à définir deviennent ainsi des combinaisons de propriétés d'interactions. Voici trois définitions possibles de nos interactions à l'aide de ces propriétés :

1. Collaboration : Présence de Validation Commune et de Construction Commune des Connaissances au sein d'une même activité.
2. Coopération : Présence de Validation Commune et absence de Construction Commune des Connaissances au sein d'une même activité
3. Compétition : Présence de Conflit Interne ou de Conflit Externe

Maintenant que nous avons une idée de comment décomposer nos interactions, nous devons formaliser leurs différentes propriétés pour être en mesure de les détecter automatiquement. Comme nous l'avons énoncé plusieurs fois au cours des différents chapitres précédents (notamment dans la section II.3 dédié à l'état de l'art sur la scénarisation), nous avons fait le choix de baser notre approche sur des éléments de *game design*. Or, le modèle de scénarios que nous proposons intègre ces éléments à travers les objets de jeu et à travers la modélisation des activités qui s'avère particulièrement cruciale pour la détection des propriétés d'interactions.

Dans le cadre de certains systèmes concurrents, la dynamique d'un système est modélisée à l'aide de fonctions produisant ou consommant des ressources. Nous nous inspirons de cette approche pour la modélisation formelle des activités d'un JSMJ dont les rôles décrivent les consommations et les productions de ressources opérées par les joueurs. En reprenant la

modélisation d'un scénario présentée dans le chapitre III, nous pouvons examiner une activité à l'aide du *framework* d'analyse suivant :

- **Rôles (R)** : l'ensemble des rôles prototypiques présents dans une activité. Un rôle est défini par ses ressources, buts, équipes et contraintes.
    - **Ressources** : définies par quatre ensembles d'objets de jeu et de compétences gérés par le rôle.
        - **Objets de jeu consommés (Cons)** : l'ensemble des objets de jeu consommés par le rôle.
        - **Objets de jeu produits (Prod)** : l'ensemble des objets de jeu produits par le rôle.
        - **Compétences requises (Cr)** : l'ensemble des compétences et connaissances requises par le rôle.
        - **Compétences ciblées (Cc)** : l'ensemble des compétences et connaissances ciblées par le rôle.
    - **Equipes (E)** : l'ensemble des équipes auxquelles appartient le rôle.
    - **Buts (B)** : l'ensemble des buts du rôle. Chaque but est composé de quatre éléments (**r** : Rôle, **action** : {« consomme », « produit », « requiert », « cible »}, **re** : {Objet de jeu, Compétence}, **b** : {vrai, faux}). Lorsque « b » vaut vrai, le but est atteint si la ressource « re » est contenue dans l'ensemble de ressource associé à « action » pour le rôle « r ». Par exemple, si « action » est égal à « consomme », la ressource « re » doit être présente dans l'ensemble « r.Ressources.Cons ». Lorsque « b » vaut faux, le but est atteint si la ressource « re » n'est pas contenue dans l'ensemble de ressource associé à « action » pour le rôle « r ».
      Il est à noter que le premier élément d'un but (le rôle) peut tout à fait être différent du rôle auquel ce but est associé. Par exemple si nous souhaitons décrire l'objectif suivant : le rôle 1 doit empêcher le rôle 2 de produire la ressource re1 ; nous ajouterons le but suivant au rôle 1 : (R2, « produit », re1, faux).
    - **Contraintes (C)** : l'ensemble des contraintes de ressources liés à l'inventaire du rôle. Chaque contrainte est composée de trois éléments (**re** : {Objet de jeu, Compétence}, **comparaison** : {<, >, <=, etc…}, **nb** : $\mathbb{N}$). Une contrainte est respectée si la « comparaison » entre la quantité de « re » dans l'inventaire du rôle et le nombre « nb » est vérifiée. Ces contraintes permettent de vérifier la présence ou l'absence d'une ressource dans des quantités données. Pour représenter le fait qu'un joueur assumant r1 ne doit pas avoir de carte dans sa main, nous aurions la contrainte suivante : (« Carte », « == », 0).
- **Fonction Equipe (FE)** : <**r1** : Rôle, **r2** : Rôle>-→ {vrai, faux}. Cette fonction retourne vrai si les rôles « r1 » et « r2 » appartiennent à la même équipe et faux dans le cas contraire.
- **Fonction But (FB)** : <**r** : Rôle> → {vrai, faux}. Cette fonction retourne vrai si l'ensemble des buts de « r » sont atteint et faux dans le cas contraire.
- **Fonction Contraintes (FC)** : <**r** : Rôle> → {vrai, faux}. Cette fonction retourne vrai si l'ensemble des contraintes de « r » sont respectées et faux dans le cas contraire.

L'idée est alors d'utiliser les éléments du *framework* d'analyse pour détecter nos différentes propriétés d'interaction, répondant ainsi à la question « Quelles propriétés peut-on extraire

de la modélisation des activités ? » évoquée dans la section précédente. Ainsi, nous considérons qu'une interaction émerge d'une activité de JSMJ si les différentes propriétés la composant sont extraites des éléments du *framework* modélisant l'activité. Une propriété est donc une combinaison logique de prédicats en lien avec les éléments du *framework*.

Nous avons défini ainsi nos 5 propriétés d'interactions. Bien entendu, d'autres propriétés peuvent être définies pour répondre aux besoins des enseignants ou des concepteurs de JSMJ.

1. Individualiste (Ind) : Un apprenant qui prend un rôle dans une activité est individualiste s'il est l'unique joueur actif dans cette activité, c'est à dire, que tous les joueurs des autres rôles n'ont pas de buts. La propriété « Individualiste » est extraite si :

   $\exists R_1 \in R \; \forall R_2 \in R$ tel que $(R_2 \neq R_1) \wedge (R_1.B \neq \emptyset) \wedge (R_2.B = \emptyset)$

2. Conflit Externe (CE) : cette propriété est vérifiée si, dans une activité, des joueurs appartenant à une équipe atteignent leurs objectifs alors que des joueurs d'une autre équipe ratent les leurs. La propriété « Conflit Externe » est extraite si :

   $\exists R_1, R_2 \in R$ tel que $\neg FE (R_1, R_2) \wedge \neg FB(R_1) \wedge FB (R_2)$

3. Conflit Interne (CI) : cette propriété est vérifiée lorsqu'un apprenant échoue dans la réalisation de ses objectifs, alors qu'il aurait pu y arriver en consommant ou produisant les ressources (objets de jeu et connaissances) d'un autre rôle de la même équipe. Pour extraire cette propriété, nous devons définir une nouvelle fonction qui vérifie si un rôle aurait atteint ses buts avec d'autres ressources. En effet, dans le cas présent, nous voulons vérifier qu'un joueur atteindrait ses buts avec les ressources d'un autre joueur. Par conséquent, nous avons défini la fonction suivante :

   **Fonction But avec Remplacement de Ressources (FBRR)** : <**r** : R, **res** : Ressources> → {vrai, faux}. Cette fonction affecte « res » à « r.Ressources » et retourne l'appel de FB sur ce nouveau « r ».

   La propriété « Conflit Interne » est extraite si :

   $\exists R_1, R_2 \in R$ tel que $FE (R_1, R_2) \wedge \neg FB (R_2) \wedge FBRR (R_2, R_1.Ressources)$

4. Validation Commune (VC) : cette propriété est vérifiée si les apprenants d'une même équipe sont actifs et ne sont pas en conflit interne. La propriété « Validation Commune » est extraite si :

   $\forall R_1, R_2$ tel que $FE (R_1, R_2) \wedge (FB (R_1) = FB (R_2)) \wedge (R_1.B \neq \emptyset) \wedge \neg\exists R_3(FE (R_1, R_3) \wedge (FB (R_3) \neq FB (R_1)))$

5. Construction Commune des Connaissances (CCC) : cette propriété est vérifiée si les apprenants sont actifs, ne sont pas en conflit interne (Validation Commune VC) et ciblent tous les mêmes compétences à travers le rôle auquel ils sont associés dans l'activité. La propriété « Construction Commune des Connaissances » est extraite si :

   $\forall R_1, R_2$ tel que $VC (R_1, R_2) \wedge (R_1.Ressources.Cc = R_2.Ressources.Cc)$

Grâce à cette modélisation, la détection de la présence d'une propriété au sein d'une activité revient à vérifier la formule logique correspondante.

Dans le chapitre III, nous avons décrit un scénario comme un ensemble de séquence d'activités. Ainsi, la détection des interactions ayant lieu au sein d'un scénario revient à détecter les propriétés présentes dans l'ensemble des séquences d'activités possibles d'un scénario. Cependant, nous l'avons déjà énoncé dans le chapitre III et nous reviendrons dessus dans le chapitre V, le nombre de séquences composant un scénario peut être particulièrement grand. Par conséquent, nous devons répondre aux deux questions suivantes : « Comment calculer l'ensemble des séquences d'un scénario ? » et ensuite « Comment détecter les interactions au sein d'une séquence ? ». Le traitement de la première question est détaillé dans le chapitre V. La section IV.3 présente la solution que nous avons apportée pour répondre à la deuxième question.

## IV.3. Détection des interactions dans une séquence d'activités

Une séquence est composée d'activités. Détecter les propriétés pouvant avoir lieu dans une séquence revient donc à détecter les propriétés des activités la composant. Nous avons donc établi une procédure permettant de résumer les informations relatives aux propriétés détectées dans les activités et les séquences du scénario. Cette procédure est décrite comme suit.

La première étape consiste à extraire les propriétés des activités du scénario. Ainsi, pour chaque activité, nous construisons un vecteur dont la taille correspond au nombre de propriétés possibles (dans notre cas 5). Lorsqu'une propriété est observée dans une activité alors sa valeur dans le vecteur est à 1, sinon elle est à 0. Par exemple, pour une activité « a » où nous détectons les propriétés conflit externe (CE) et validation commune (VC), le vecteur est le suivant : « a » [Ind = 0 ; CI = 0 ; CE = 1 ; VC = 1 ; CCC = 0]

Une fois que nous avons détecté toutes les propriétés dans chacune des activités apparaissant dans le scénario, nous calculons pour chaque séquence d'activités les occurrences des différentes propriétés. La distribution est obtenue en additionnant les vecteurs des différentes activités (la résultante). Par exemple, si nous avons 3 activités « a », « b » et « c » décrites comme suit :

« a » [Ind = 0; CI = 0; CE = 1; VC = 1; CCC = 0]

« b » [Ind = 0; CI = 0; CE = 0; VC = 1; CCC = 1]

« c » [Ind = 0; CI = 1; CE = 1; VC = 0; CCC = 0]

Alors la séquence <a, b, c> est décrite par la résultante suivante :

<a, b, c> [Ind = 0; CI = 1; CE = 2; VC = 2; CCC = 1]

Le vecteur de la séquence « a->b->c » représente le fait que les propriétés « Validation Commune » et « Conflit Externe » apparaissent deux fois dans la séquence, et que les

propriétés « Conflit Interne » et « Construction Commune de Connaissances » n'y apparaissent qu'une fois.

Nous répétons à nouveau ce processus additif pour calculer le vecteur d'occurrences des propriétés pour un scénario en additionnant les vecteurs associés aux différentes séquences du scénario. Nous rappelons que le calcul de ces séquences est présenté dans le chapitre V.

Par exemple, si le scénario est composé des trois séquences suivantes :

<a, b, c> [Ind = 0; CI = 1; CE = 2; VC = 2; CCC = 1]

<a, b> [Ind = 0; CI = 0; CE = 1; VC = 2; CCC = 1]

<b, c> [Ind = 0; CI = 1; CE = 1; VC = 1; CCC = 1]

Alors le vecteur de distribution des propriétés pour un scénario est le suivant :

[Ind = 0; CI = 2; CE = 4; VC = 5; CCC = 3]

Nous reviendrons plus tard (dans le chapitre VII) sur la manière d'utiliser ces informations quantitatives, mais nous pouvons d'ores et déjà analyser le résultat de cet exemple comme suit : les conflits externes (CE) et conflits internes (CI) sont des marqueurs de compétition, toutefois le conflit interne peut être évité (chemin <a, b>). La construction commune des connaissances (CCC) et la validation commune (VC) sont des marqueurs de collaboration et de coopération et sont indispensables dans l'ensemble des chemins évoqués. Nous pouvons conclure que le scénario présenté est davantage collaboratif que compétitif (ce qui est confirmé par le fait que la quantité VC + CCC est supérieure à la quantité CE + CI), bien qu'il soit impossible d'échapper au conflit externe (CE) qui est présent dans toutes les séquences. Le scénario décrit peut-être, selon la description donnée par l'utilisateur, un jeu collaboratif opposant différentes équipes.

## IV.4. Synthèse

Notre revue de la littérature ne nous a pas permis d'identifier des définitions consensuelles des interactions et ce même entre les chercheurs et les experts d'un même domaine de connaissances. Pourtant, il est indispensable pour notre approche de trouver un cadre de formalisation de ces interactions en vue de leur détection automatique. La solution que nous proposons se base sur le principe suivant : décomposer les interactions abstraites et souvent complexes en propriétés objectivables plus simples pouvant être extraites à partir de la modélisation des activités. L'avantage principal de cette décomposition, est qu'elle permet à l'utilisateur de notre système de définir les interactions qui les intéressent en combinant de manière assez flexible des propriétés de plus bas niveau.

Décomposer des interactions en propriétés n'est cependant pas suffisant. En effet, il est également nécessaire d'être en mesure de formaliser et de détecter ces propriétés. Pour ce faire, nous nous sommes basés sur la modélisation des activités que nous avons détaillée dans le chapitre III avec MPLGO.

En introduction, nous avions évoqué notre ambition de permettre à un utilisateur : enseignant et/ou concepteur de jeu de définir ses propres interactions à partir de propriétés. La manipulation de nos propriétés pour construire ces interactions nous semble faisable pour un enseignant. En revanche, nous pensons qu'il n'en est pas de même pour ce qui est de la création de nouvelles propriétés. Or, cela apporterait beaucoup de diversité aux interactions représentables par les utilisateurs. C'est pourquoi nous sommes convaincus que la création d'un *framework* modulable permettant à un utilisateur disposant de compétences techniques (un concepteur de jeu par exemple) de partager les propriétés qu'il a définies, est une perspective intéressante voire indispensable de ce travail.

Grâce à notre approche se basant sur les éléments de *game design* constituant les activités d'un JSMJ, nous pouvons déterminer à l'avance les propriétés qu'elles peuvent susciter. Le processus de qualification d'un scénario en termes d'interactions est alors le suivant :

1. Décrire les activités composant un jeu (chapitre III),
2. Détecter les propriétés d'interactions pouvant émerger de ces activités (section IV.2),
3. Établir les séquences d'activités du scénario représentatives de l'ensemble des interactions pouvant émerger d'un scénario de JSMJ (Chapitre V),
4. Extraire des propriétés de chacune des activités de la séquence (section IV.3),
5. Cumuler ces propriétés à l'échelle d'une séquence et ensuite à l'échelle du scénario (section IV.3).

Ce processus nous permet d'analyser un scénario décrit comme des séquences d'activités respectant la modélisation de MPLGO. Le chapitre V présente donc les méthodes que nous avons employées pour obtenir l'ensemble des séquences d'un scénario.

# V. Algorithmes d'analyse

Le positionnement de ce chapitre par rapport aux autres de la section 2 sur les contributions est présenté dans la Figure 6 (page 24).

Ce travail de thèse vise à déterminer l'ensemble des interactions entre pairs possibles lorsque des joueurs participent à une session de jeu. Or, les JSMJ sont des environnements qui peuvent engendrer une grande variété d'interactions selon le degré de liberté accordé aux joueurs. La formalisation du chapitre IV nous indique que les interactions sont liées aux activités. Or, les séquences d'activités permettant aux joueurs d'atteindre la fin du scénario n'emploient pas nécessairement toutes les mêmes activités et peuvent par conséquent faire émerger des interactions différentes. Il est donc nécessaire pour extraire toutes les interactions pouvant émerger d'un scénario JSMJ de calculer l'ensemble des séquences de ce dernier et ensuite, d'extraire les interactions de ces séquences. Toutefois, la complexité algorithmique d'un tel calcul sur des scénarios complexes montre qu'une telle démarche est coûteuse. Pour répondre à cette problématique, nous avons choisi de chercher et de travailler sur un sous-ensemble de séquences d'activités appelé SA. Celui-ci devra porter l'essentiel de l'information sur les interactions suscitées par le scénario. Bien entendu, le risque de cette approche est de perdre de l'information. En effet, des séquences pertinentes du point de vue des interactions risquent d'être filtrées. Pour cela, nous avons mené une étude pour évaluer l'algorithme de calcul de SA proposé (cf. section VII.5).

Afin de limiter le nombre de séquences à étudier et donc de construire avec une complexité algorithmique raisonnable l'ensemble SA, nous avons fait le choix de nous appuyer sur les liens de précédences entre les activités que nous détaillons dans la section V.1.

Ainsi, le calcul du SA se décompose en deux étapes. La première consiste à déterminer l'ensemble des séquences dite « admissibles » respectant les contraintes de précédence et la deuxième consiste à vérifier la faisabilité de ces séquences admissibles du point de vue du jeu. Cette deuxième étape est indispensable car une séquence d'activités respectant des contraintes de précédences relatives aux productions et aux consommations des ressources n'est pas forcément une séquence qui permet aux joueurs d'atteindre la fin du jeu. Cette deuxième étape sélectionne donc les séquences « admissibles » permettant aux joueurs d'atteindre la fin du jeu, c'est-à-dire, un des états finaux du jeu.

## V.1. Etape 1 : Déterminer les séquences d'activité admissibles du scénario

Comme indiqué dans l'introduction du Chapitre V, le calcul du SA repose sur les liens de précédence qui relient les différentes activités d'un scénario de JSMJ. Nous avons fait ce choix car le nombre de séquences respectant les liens de précédence est obligatoirement inférieur (ou égal dans le cas où les activités sont toutes indépendantes) au nombre de toutes les combinaisons d'activité possibles. Ce dernier étant égal à n!. Ainsi, plus le nombre de liens de

précédence est important, plus le nombre de combinaisons d'activités possibles diminue. L'étape 1 est divisée en trois sous-étapes :

1. Etablir le graphe de précédence du jeu : il s'agit d'un graphe orienté où les nœuds représentent des activités et les arcs les liens de précédence reliant les activités,
2. Calculer la structure d'états : cette structure est un graphe (de type diagramme de Hasse) où les nœuds de la structure indiquent l'ensemble des activités effectuées à un état donné du scénario et les arcs précisent les activités qu'il est nécessaire de réaliser pour passer d'un nœud à un autre. La construction de cette structure se fait à partir du graphe de précédence,
3. Extraire des séquences admissibles : l'exploitation de la structure d'activités nous fournit les différentes séquences admissibles respectant les liens de précédence du scénario.

### V.1.a. Sous-étape 1 : Établir le graphe de précédence du jeu

Dans un graphe de précédence, les nœuds représentent les activités du JSMJ et les arcs les liens de précédence. La construction du graphe de précédence à partir de la modélisation des activités (en particulier, à partir des ensembles d'objets consommés « Cons » et produits « Prod » cf. section IV.2) se fait comme suit :

- Créer un nœud pour chaque activité du scénario
- Relier par un arc une activité $A_1$ à une autre activité $A_2$ si $A_1$ produit une ressource consommée par $A_2$.

Prenons un exemple pour illustrer le processus de construction du graphe de précédence. Soit a, b, c, d et e, 5 activités d'un scénario de JSMJ. Ces activités sont définies comme suit :

- a ne consomme aucune ressource et produit la ressource $r_1$,
- b consomme une ressource $r_4$ et produit la ressource $r_1$,
- c consomme la ressource $r_1$ et produit la ressource $r_2$,
- d ne consomme aucune ressource et produit la ressource $r_3$,
- e consomme les ressources $r_2$ et $r_3$ et ne produit aucune ressource.

Premièrement nous créons un nœud pour chacune des activités, ce qui nous donne la Figure 9.

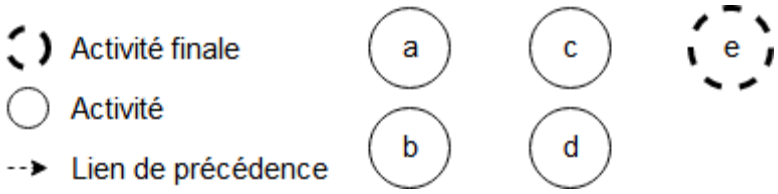


*Figure 9 : Première étape de la construction d'un graphe de précédence*

Par la suite, nous relions nos différentes activités entre elles conformément à la méthode décrite plus haut. L'activité e consomme $r_2$ et $r_3$ respectivement produites par les activités c et d. Ainsi nous pouvons déduire que c et d précèdent e. L'activité c quant à elle, consomme

la ressource $r_1$, or les activités a et b permettent de produire cette ressource. Ces deux activités précèdent donc l'activité c. La Figure 10 présente le graphe ainsi obtenu.

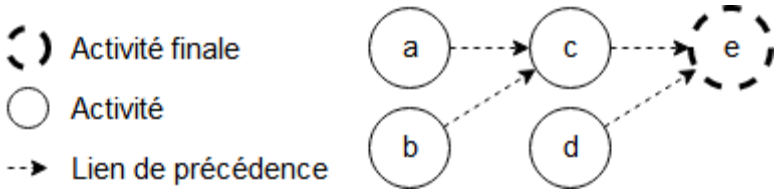


*Figure 10 : Exemple d'un graphe de précédence*

Cette représentation est un graphe de précédence « classique », dans le sens où l'ensemble des activités précédant une activité A doivent avoir été effectuées avant de pouvoir effectuer cette activité A. Il peut donc arriver que deux activités (ou plus) produisent la même ressource $R_1$ consommée par une troisième activité. Dans ce cas précis, l'exécution d'une seule de ces activités produisant $R_1$ est suffisant pour effectuer la troisième activité. Pour représenter ce cas de figure, nous étendons le graphe de précédence classique (possédant uniquement des liens de type « ET ») à un graphe de précédence ET-OU (possédant des liens de type « ET » et des liens de type « OU »).

En reprenant notre exemple, nous constatons que les liens de précédence entre a, b et c devraient donc être modifiés, puisque a et b produisent exactement les mêmes ressources nécessaires à c. La Figure 11 présente le graphe de précédence ET-OU créé à partir de l'exemple précédent. Ici, les activités c et d précèdent e et sont toutes deux obligatoires. A l'inverse, seulement une activité parmi a et b doit être effectuée pour pouvoir exécuter c.

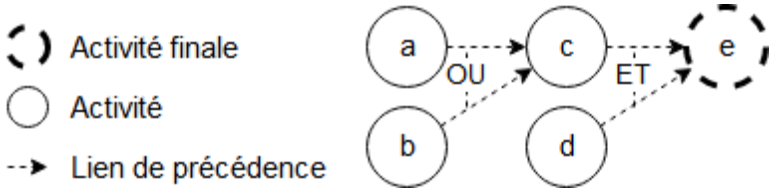


*Figure 11 : Exemple d'un graphe de précédence ET-OU*

Ce nouveau graphe retranscrit mieux notre scénario que le graphe classique puisqu'il modélise le fait qu'il soit tout à fait possible de n'effectuer qu'une seule des deux activités précédant c pour pouvoir effectuer cette dernière. Naturellement, ce cas peut être généralisé. Par exemple, si n activités précèdent une activité A par un lien « OU », seule une activité parmi ces n activités est nécessaire pour pouvoir effectuer cette activité A.

## V.1.b. Sous-étape 2 : Création d'une structure d'états

Nous exploitons le graphe de précédence produit dans la sous-étape 1 pour calculer les séquences réalisables du scénario. La théorie appelée *Knowledge Space Theory* (KST) (Doignon & Falmagne, 1985; Falmagne, Cosyn, Doignon, & Thiéry, 2006; Jean-Paul Doignon & Falmagne, 1999) utilise un graphe de précédence pour calculer les enchaînements des activités à disposition des apprenants en construisant une structure de type diagramme de Hasse. Nous

nous appuyons sur cette structure pour extraire les différentes séquences d'activités qu'il est possible d'effectuer pour atteindre l'état final.

Ainsi, les nœuds de cette structure, désignés par états, correspondent aux états du jeu. Un état est défini par l'agrégation des activités ayant déjà été effectuées par les apprenants. Les arcs quant à eux sont orientés et indiquent l'activité qu'il est nécessaire d'effectuer pour passer d'un premier état à un second. Par exemple, à partir de l'état « ab », l'apprenant pourrait passer à l'état « abc » en réalisant l'activité « c ». L'algorithme de création de cette structure d'états est en chaînage arrière ; le premier nœud ou état créé est celui qui contient toutes les activités du scénario. L'Algorithme 1 présente les étapes de la création de la structure d'état.

*Algorithme 1 : création de la structure d'état*

```
Soit S un scénario de jeux sérieux.

Procédure Structure(S) :
      Soit L une liste vide.
      Soit St une structure vide
      Créer un état E incluant toutes les activités de S
      Mettre E dans L.

      Tant que L n'est pas vide, pour chaque état de taille n, E(n) de L
      Faire :
            Pour toutes les activités Ai de E(n) avec i allant de 1 à n :
                  Soit b un booléen valant Faux
                  // Cas 1
                  Si Ai ne précède aucune activité de E(n) :
                        b vaut vrai
                  // Cas 2
                  Si Ai précède une activité Ak de E(n) par un lien OU et
                  qu'il existe une activité Aj de E(n) qui précède Ak par un
                  lien OU :
                        b vaut vrai
                  Si b vaut vrai :
                        Créer un état E' équivalent à E(n) privé de Ai et
                        lié à E par un arc valant Ai.
                        Ajouter E' à L
            Retirer E(n) de L
            S'il existe un état F équivalent à E(n) dans St :
                  Ajouter à F les arcs de E(n)
            Sinon :
                  Ajouter E(n) à St
      Retourner St
```

Nous notons que seuls les enchaînements d'états qui mènent à un état final du jeu sont pris en compte, les autres sont ignorés. Un état est dit « final » s'il inclut l'activité finale du scénario. Dans la section V.1.a, nous avons donné un exemple d'un graphe de précédence (cf. Figure

11) pour un scénario composé de 5 activités. À partir de ce graphe de précédence, nous détaillons ci-après la construction de sa structure d'activités (cf. Figure 12).

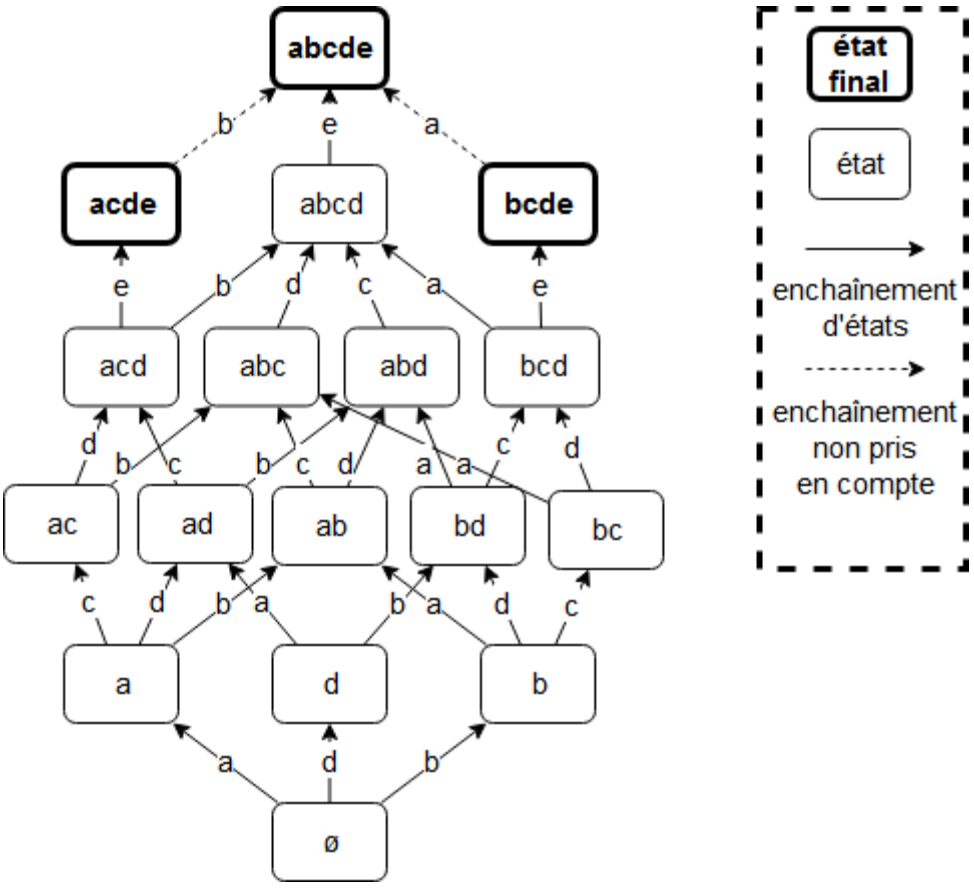


*Figure 12 : Structure d'états construite à partir d'un graphe de précédence ET-OU de la Figure 11*

Tout d'abord, nous créons le nœud agrégeant toutes les activités du scénario « abcde ». D'après la Figure 11, et les conditions décrites précédemment pour la construction de la structure, nous créons trois nouveaux états : l'état « abcd » qui n'est pas final et qui atteint « abcde » par e car e ne précède aucune autre activité dans le diagramme de précédence (Cas 1 de l'Algorithme 1), et les états « acde » et « bcde » qui sont des états finaux (ils contiennent toujours l'activité finale e) et sont obtenus en retirant respectivement les activités b et a qui précèdent l'activité c par un lien OU (Cas 2 de l'Algorithme 1). Nos trois nouveaux nœuds sont ajoutés à la liste des nœuds à développer. Nous réitérons le processus pour chacun de ces trois nœuds.

L'algorithme ainsi décrit permet de construire une structure d'état composée de 22 enchaînements d'activités possibles de l'état vide aux états incluant l'activité finale du scénario. Les enchaînements extraits de cette structure sont les suivants :

<a, c, d, e> ; <a, c, d, b, e> ; <a, c, b, d, e> ; <a, d, c, e> ; <a, d, c, b, e> ; <a, d, b, c, e> ; <a, b, c, d, e> ; <a, b, d, c, e> ; <b, c, d, e> ; <b, c, d, a, e> ; <b, c, a, d, e> ; <b, d, c, e> ; <b, d, c, a, e> ;

<b, d, a, c, e> ; <b, a, c, d, e> ; <b, a, d, c, e> ; <d, a, c, e> ; <d, b, c, e> ; <d, a, b, c, e> ; <d, a, c, b, e> ; <d, b, a, c, e> ; <d, b, c, a, e>

Tous ces enchaînements ne vont pas forcément appartenir à l'ensemble SA. Dans la section V.1.c, nous présentons ce processus d'extraction de séquences.

### V.1.c. Sous-étape 3 : Obtention des séquences d'activités dans le scénario

L'obtention des séquences se fait en partant de l'état initial (l'état ø en bas de la structure d'états) vers l'état final visé (les états en gras). Chaque arc indique l'activité à effectuer dans la séquence pour passer à l'état suivant. Si un état possède plusieurs arcs de sorties, c'est autant d'embranchements possibles pour la séquence en cours de construction. Les arcs en pointillés, faisant le lien entre deux états finaux, ne seront jamais exploités par l'algorithme qui termine son traitement au premier état final rencontré. En effet, tenant compte du modèle de scénarios que nous avons proposé, une séquence de JSMJ ne peut pas contenir plusieurs activités finales successives.

Dans notre approche, l'ordre des activités n'a aucune influence sur le type ou le nombre des propriétés détectées dans une séquence puisque nous nous intéressons uniquement à la répartition (ou fonction de distribution cumulative) des propriétés dans un scénario. Si deux séquences (ou enchaînements) utilisent les mêmes activités mais ordonnées différemment alors les mêmes types et nombres de propriétés d'interactions seront identifiés. Il est donc pertinent de filtrer les séquences obtenues à l'aide du graphe de précédence en éliminant les séquences redondantes (séquences contenant les mêmes activités qu'une autre séquence déjà identifiée mais ordonnées différemment), afin d'éviter une explosion combinatoire due au calcul de l'ensemble de toutes les séquences d'un scénario de JSMJ. Ainsi, cela signifie que nous sommes en mesure de réduire la liste des séquences admissibles en ne conservant qu'une seule séquence pour toutes les combinaisons des mêmes d'activités, surtout que les séquences ignorées n'auraient apporté aucune information supplémentaire. De cette façon, dans l'exemple, nous ne gardons que les séquences suivantes pour l'étape de vérification de faisabilité :

- <a, c, d, e>
- <b, c, d, e>
- <a, b, c, d, e>

Grâce à ce choix, nous réduisons le nombre des séquences admissibles à seulement trois. Bien entendu, ce choix peut engendrer une perte d'information dû au fait que nous écartons certaines séquences. Nous faisons l'hypothèse que l'impact de cette perte est limité et que l'information sur les types et la répartition des propriétés sur l'ensemble du scénario sera préservée. Nous vérifions cette hypothèse à travers l'expérimentation présentée au chapitre VII.

Dans l'exemple, ces trois séquences correspondent à l'ensemble des stratégies disponibles et ceci malgré la perte d'information. En effet, la première séquence consiste à réaliser l'activité

« a » et à ne pas faire « b », la deuxième à réaliser « b » et ne pas faire « a » et la dernière séquence consiste à réaliser les deux. Ces trois séquences correspondent bien aux stratégies qu'il est possible d'employer dans cet exemple.

Déterminer ces séquences n'est cependant que la première partie de l'algorithme. Le fait qu'une séquence soit conforme aux liens de précédences menant à l'activité finale, ne signifie pas qu'elle soit exécutable en jeu. En effet, pour qu'une séquence soit exécutable, elle doit permettre d'atteindre la fin du jeu à partir de l'état initial de celui-ci et des inventaires des joueurs. Nous vérifions donc la faisabilité de chacune des séquences obtenues à l'aide des algorithmes présentés dans la section suivante.

## V.2. Étape 2 : Vérifier la faisabilité d'une séquence

Les séquences trouvées à l'aide de la méthode décrite dans la section précédente ne sont pas nécessairement toutes faisables. Ainsi, il est nécessaire d'établir la faisabilité de celles-ci. En effet, ces séquences sont obtenues sans tenir compte des ressources présentes à l'état initial du jeu et des inventaires des joueurs. Pour vérifier la faisabilité d'une séquence, notre algorithme cherche une exécution du jeu (autrement dit une suite de transformations relatives aux objets de jeu consommés et produits cf. section IV.2) permettant d'atteindre la fin du jeu en utilisant exclusivement les activités de la séquence testée dans l'ordre fourni (en répétant la séquence et les activités autant de fois que nécessaire).

D'après notre modèle de scénarisation (cf. section III.5), les joueurs possèdent un inventaire d'objets de jeu qu'ils doivent faire évoluer afin d'atteindre la fin du jeu. La description de l'activité finale nous fournit, quant à elle, les listes des ressources que les joueurs doivent obtenir pour finir le jeu ; nous appellerons ces listes les inventaires requis.

L'idée que nous avons suivie, est alors de faire évoluer ces inventaires requis en fonction des consommations et productions associées aux activités de la séquence en partant de la fin de la séquence (l'algorithme exécute la séquence en chaînage arrière). Ainsi, de manière itérative l'algorithme remonte jusqu'à la première activité de la séquence et construit la succession des inventaires requis permettant aux joueurs de réaliser cette séquence d'activité. Si l'inventaire initial d'un joueur englobe l'inventaire requis obtenu à l'issue du chainage arrière, le joueur serait alors en mesure de participer à cette séquence d'activité. Cette dernière permet alors aux joueurs d'atteindre la fin du scénario.

La vérification de la faisabilité d'une séquence consiste donc à analyser les activités de la séquence en chaînage arrière afin d'inférer les inventaires requis pour réaliser cette séquence et vérifier que les inventaires réels des joueurs les englobent bien.

Pour simuler les activités de la séquence en chaînage arrière, nous avons besoin de définir deux éléments clés : la notion de combinaison d'inventaire requis et de rôle prototypique et le principe de la simulation d'une activité en chaînage arrière.

Concernant la notion de combinaison, il s'agit de déterminer quels inventaires requis doivent être associés aux différents rôle prototypiques et donc d'évaluer toutes les combinaisons possibles. En effet, dans le cadre de notre modèle, les apprenants peuvent être associés à n'importe quel rôle prototypique aux seules conditions qu'ils aient en leur possession les ressources adéquates (les ensembles de contraintes et d'objets de jeu consommés associés aux rôles prototypiques de l'activité) et appartiennent aux bonnes équipes (ensemble des équipes d'un rôle + fonction FE sur les équipes entre les rôles d'une activité). Ainsi il est possible d'avoir plusieurs sorties possibles pour une activité effectuée par les mêmes joueurs en fonction des rôles attribués à ceux-ci. Les inventaires requis sont là pour symboliser les ressources que les joueurs doivent avoir en leur possession pour finir le jeu et en ce sens servent à représenter les inventaires des joueurs lors du chaînage arrière. Ainsi, si en chaînage avant il peut exister plusieurs combinaisons de joueurs par rapport aux rôles prototypiques à disposition, nous avons en chaînage arrière plusieurs combinaisons d'inventaire requis pour les rôles à disposition.

Maintenant que nous avons établi la notion de combinaison d'inventaire requis et de rôles prototypiques, nous explicitons le principe de la simulation d'une activité en chaînage arrière. La simulation d'une activité en chaînage **avant** consiste à retirer des inventaires des joueurs les ressources consommées par leurs rôles prototypiques, puis à y rajouter les ressources produites et les connaissances travaillées. La simulation en chaînage **arrière**, pour chaque combinaison (inventaire requis, rôles prototypiques) de l'activité procède à la transformation inverse suivante :

1. Enlever de l'inventaire requis toutes les ressources produites par le rôle prototypique s'y trouvant.
2. Ajouter à l'inventaire requis toutes les ressources consommées par le rôle prototypique.

Notre algorithme de chaînage arrière repose sur cette notion de combinaison des inventaires requis et des rôles prototypiques et sur ce principe de simulation d'une activité en chaînage arrière pour vérifier la faisabilité d'une séquence. Il évolue dans la séquence d'activité en comparant les exécutions en chaînage arrière des différentes combinaisons possibles (inventaire requis, rôles prototypiques) pour chacune des activités testées. L'idée étant alors de choisir la combinaison qui nous semble la plus efficace. Rappelons que nous souhaitons trouver un chemin allant de l'état initial du jeu à l'état final et que pour ce faire, notre algorithme doit trouver en chaînage arrière un état des inventaires requis inclus dans l'état initial du jeu. Ainsi, dans le cadre du chaînage arrière, nous visons un état où les inventaires requis sont tous vides. Cet état étant inclus *de facto* dans l'état initial du jeu (et ce pour tous les scénarios de JSMJ). Compte tenu de cet état de fait, nous privilégions les combinaisons qui nous font converger rapidement vers l'ensemble vide. Pour ce faire, nous avons défini une fonction d'utilité. Cette fonction se base sur le nombre de modifications apportées aux inventaires requis par l'étape 1 de l'exécution en chaînage arrière. Cette fonction vise ainsi à maximiser le nombre de ressources retirées des inventaires requis. Son fonctionnement est davantage détaillé dans la section 0.

*Algorithme 2 : Description générale simplifiée de l'algorithme de vérification de la faisabilité d'une séquence*

```
Soit S une séquence de n activités.
Soit E l'état actuel des inventaires requis.
Soit j l'indice dans la séquence de l'activité simulée la plus proche du
début de la séquence. Il s'agit d'un entier appartenant à [1, n].
Procédure Validation (S, E, j) :
      Soit P une pile vide.
      Pour toutes les activités Ai de la séquence S pour i allant de n à 1 :
            Calculer les différentes combinaisons (inventaire requis, rôle
            prototypique)
            Si (i < j) ou ((i == n) et (j == n)) : //Nous forçons la
            // simulation de la première occurrence d'une activité de la
            // séquence
                  Empiler dans P toutes les combinaisons (inventaire
                  requis, rôle prototypique) dont l'application sur
                  l'activité Ai retourne une utilité positive ou nulle
                  j = i
            Sinon : // Ai a déjà été simulée dans une récursion passée
                  Empiler dans P toutes les combinaisons (inventaire
                  requis, rôle prototypique) dont l'application sur
                  l'activité Ai retourne une utilité positive
            Modifier E avec la combinaison ayant la meilleure utilité

            // Si i ou j vaut 1, la séquence a déjà été finie une fois.
            Si ((i == 1) ou (j == 1)) et que l'état actuel des inventaires
            requis est inclus dans l'état initial du jeu :
                  Alors retourner vrai // Fin de procédure

            Si (i ! = n) : // l'activité finale n'est jamais répétée
                  Calculer C l'ensemble des combinaisons possibles en
                  répétant Ai à partir de E
                  Pour chaque combinaison de C ayant une utilité positive :
                        Soit E' l'état généré par cette combinaison
                        Si Validation (S, E', j) retourne vrai : //Appel
                        // récursif
                              Alors retourner vrai // Fin de procédure

      // Nous explorons les combinaisons stockées en mémoire
      Tant que la pile P n'est pas vide :
            Dépiler P
            Soit E' l'état généré par la combinaison dépilée et k
            L'emplacement de son activité dans la séquence
            Si Validation (S, E', k) retourne vrai // Appel récursif
                  Alors retourner vrai // Fin de procédure
      Retourner faux. // Si cette ligne est atteinte, c'est qu'il n'y a pas
      // de chemin valable.
```

L'Algorithme 2, récursif, présente les grandes lignes du processus de vérification de la faisabilité d'une séquence. Dans le cadre de notre approche, nous souhaitons que toutes les activités d'une séquence soient utilisées au moins une fois (chaque activité de la séquence pouvant susciter des interactions différentes).

Comme indiqué plus tôt dans cette section, dans le cadre du chaînage arrière, lorsque nous analysons une activité pour trouver un chemin faisable au sein du scénario, il faut étudier les différentes associations (inventaire requis, rôle prototypique) disponibles pour établir quelle combinaison serait la plus intéressante à effectuer en maximisant la fonction d'utilité.

Cependant, il n'est pas garanti que le chemin induit par la combinaison la plus intéressante pour une activité donnée de la séquence permettra bien de réaliser le scénario. Pour pallier cela, les autres combinaisons intéressantes et l'état auquel ces dernières s'appliquent sont gardées en mémoire dans une pile. Ainsi, si le chemin en cours d'exploration s'avère impossible, il est possible de revenir à un état antérieur et d'essayer une autre combinaison (inventaire requis, rôle prototypique).

L'algorithme de vérification doit établir l'existence d'un chemin faisable pour la séquence. Ne sachant pas à l'avance si la répétition d'activités précédemment simulées est nécessaire ou pas, l'algorithme procède à un embranchement. Ici, le chemin effectuant la répétition est d'abord exploré. Si celui-ci n'apporte aucun chemin faisable, l'algorithme continue à explorer sans cette répétition.

### V.2.a. Heuristique guidant le choix des combinaisons (inventaire requis, rôle)

Etant donné que nous tentons d'atteindre un état des inventaires requis inclus dans l'état initial du jeu, nous sommes partis de l'hypothèse que la combinaison la plus intéressante devait maximiser le nombre de changements opérés par les règles de production des rôles. Ainsi, nous avons décidé de guider notre choix par une heuristique basée sur ce nombre. Soit h(C) la métrique associée à une combinaison (inventaire requis, rôle prototypique) pour une activité A. h(C) correspond pour la combinaison C au nombre de ressources retirées des inventaires requis en chaînage arrière sur A (cf. section précédente). Il est à noter, qu'une ressource utilisée (à la fois consommée et produite) n'est pas retirée de l'inventaire et ne compte donc pas dans l'heuristique.

Le choix des combinaisons (inventaires requis, rôle prototypique) s'effectue en trois temps :

1. Etablir la liste des combinaisons respectant les contraintes d'équipe liées aux rôles
2. Calculer l'utilité de chaque combinaison
3. Classer les combinaisons en fonction de leurs utilités

Nous cherchons dans un premier temps à établir les combinaisons respectant les contraintes d'équipes liées aux rôles. Ces contraintes étant fixes et les équipes étant fixes également, il est possible de déterminer ces combinaisons au moment de la modélisation du scénario.

Une activité est composée de rôle. Soit « n » le nombre de ces rôles. Soit R, l'ensemble des rôles, nous notons $R_i$ le ième rôle d'une activité. Notons card ($R_i$) la fonction retournant la cardinalité associée à $R_i$. Soit « m » le nombre de joueurs devant participer à l'activité. Nous définissons alors :

$$m = \sum_{i=1}^{n} card\ (R_i)$$

Le nombre maximum de combinaisons nbC que nous pouvons générer est donc :

$$nbC = \frac{n!}{(n-m)!}$$

Ce nombre est potentiellement important. Etablir les nbC combinaisons puis vérifier une à une si elles respectent les contraintes d'activité reste donc coûteux. Pour pallier cela, nous vérifions ces contraintes au fur et à mesure que nous construisons la combinaison. Autrement dit, pour déterminer si un m-uplet de joueurs (donc une combinaison) respecte les contraintes d'équipe, nous commençons par déterminer un (m-1) -uplet respectant les contraintes d'équipe sur les (m-1) premiers rôles auquel nous rajoutons un $m^{ième}$ joueur pour construire le m-uplet. Pour déterminer un (m-1) -uplet, nous déterminons similairement un (m-2) -uplet et ainsi de suite. Ainsi, nous commençons par établir l'ensemble des 2-uplets à partir desquels nous créons les 3-uplets etc...

Prenons un exemple pour illustrer le choix de nos combinaisons. Nous définissons une activité A composée d'un rôle R avec une cardinalité de 2. Ce rôle consomme une ressource $r_2$, produit la ressource $r_1$ et nécessite que l'ensemble des inventaires qui lui sont associés soient de la même équipe.

Soit les cinq inventaires requis suivants **avant** l'analyse de l'activité A en chaînage arrière :

- $I_1$ : En équipe avec $I_2$. Contient [$r_1$, $r_3$]
- $I_2$ : En équipe avec $I_1$. Contient [$r_3$, $r_4$], à noter que $I_2$ ne contient pas la ressource $r_1$ alors que l'activité A produit cette ressource, ceci peut tout à fait être possible et signifie qu'en l'état actuel de l'analyse $I_2$ n'a pas besoin d'avoir cette ressource dans son inventaire pour pouvoir accéder à l'une des activités finales du scénario
- $I_3$ : En équipe avec $I_4$. Contient [$r_1$, $r_4$]
- $I_4$ : En équipe avec $I_3$. Contient [$r_1$]
- $I_5$ : Seul. Contient [$r_1$]

D'après les contraintes liées au rôle prototypique de l'activité A, seules deux combinaisons sont possibles : $I_1$ et $I_2$, $I_3$ et $I_4$. En effet, $I_5$ n'a pas de coéquipier pour effectuer l'activité avec lui.

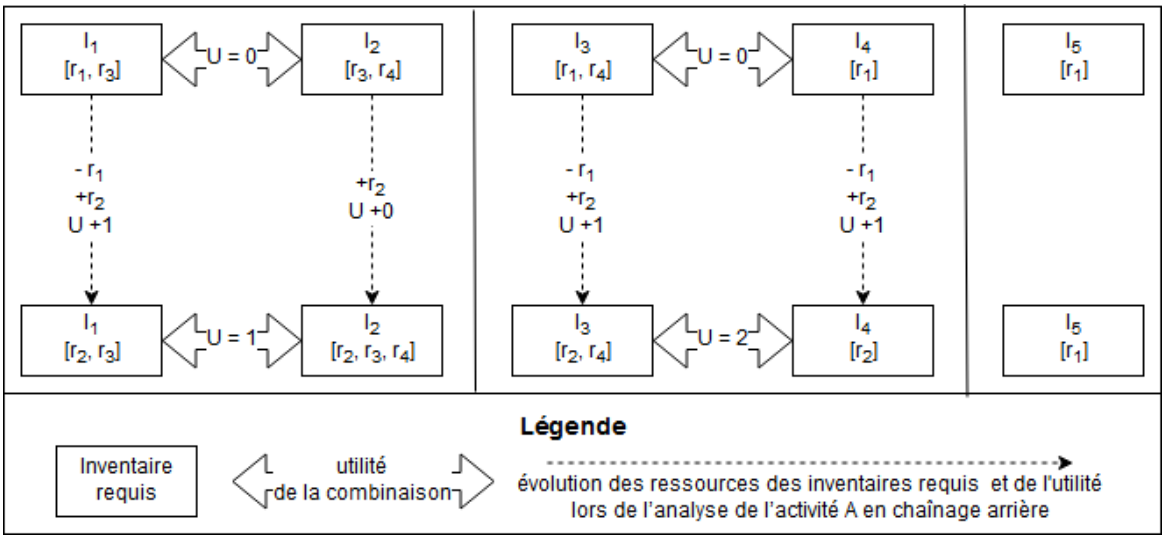


*Figure 13 : Calcul des utilités des différentes combinaisons envisageables pour l'activité A*

Calculons l'utilité associée à l'heuristique du duo ($I_1$, $I_2$) (cf. Figure 13). Pour rappel le rôle R de l'activité A produit la ressource r1, hors lors de l'analyse d'une activité en chaînage arrière, les ressources produites par les rôles doivent être retirées des inventaires requis. Par conséquent, le chaînage arrière sur A retire $r_1$ de $I_1$. Par ailleurs, le rôle associé à $I_1$ consomme la ressource $r_2$, qui est donc elle ajoutée par le chaînage arrière selon le principe rappelé précédemment. L'utilité associée à $I_1$ est donc de 1 car une seule ressource a été retirée de cet inventaire lors de l'analyse de l'activité. $I_2$, qui est associé au même rôle que $I_1$, produit également la ressource $r_1$ et consomme $r_2$. Toutefois, au contraire de $I_1$, $I_2$ ne contient pas $r_1$. Ainsi, le seul changement opéré pour $I_2$ est le rajout de $r_2$ dans son inventaire. Aucune ressource n'a été retirée de $I_2$ lors de l'analyse de l'activité, son utilité est donc de 0.

Pour obtenir l'utilité du duo ($I_1$, $I_2$), nous effectuons la somme de leurs utilités individuelles. L'utilité du duo vaut donc 1. Suite à l'analyse de A en chaînage arrière sur ($I_1$, $I_2$), nous avons les informations suivantes :

- $I_1$ : En équipe avec $I_2$. Contient [$r_2$, $r_3$]
- $I_2$ : En équipe avec $I_1$. Contient [$r_2$, $r_3$, $r_4$]
- Utilité = 1

En appliquant similairement le chaînage arrière sur le duo (I3, I4), nous obtenons :

- $I_3$ : En équipe avec $I_4$. Contient [$r_2$, $r_4$]
- $I_4$ : En équipe avec $I_3$. Contient [$r_2$]
- Utilité = 2

D'après notre heuristique, le duo ($I_3$, $I_4$) est donc plus intéressant que le duo ($I_1$, $I_2$). Le sous-algorithme 1 établit donc le classement suivant pour l'activité A associé à nos 5 inventaires :

1. ($I_3$, $I_4$), Utilité = 2
2. ($I_1$, $I_2$), Utilité = 1

## V.2.b. Application des algorithmes sur un exemple

La section qui suit a pour vocation d'illustrer le fonctionnement de la section V.2 à l'aide d'un exemple. Soit un jeu opposant deux équipes de deux joueurs. Chaque joueur doit affronter un joueur de l'équipe adverse dans une partie de bataille. Pour participer à cette bataille, les joueurs doivent payer des frais d'inscription. Le vainqueur de chaque bataille remporte un jeton qu'il peut échanger contre un ticket lui octroyant la victoire. Nous décomposons ce scénario en 3 activités et 4 Joueurs. Voici la liste des activités :

- Activité Finale (AF) : activité finale du jeu, composé de deux rôles différents
    - Rôle Vainqueur : cardinalité = 2 ; consomme : Ticket ; produit : Victoire ; contraintes : Les deux vainqueurs sont de la même équipe.
    - Rôle Perdant : cardinalité = 2 ; consomme : ø ; produit : Défaite ; contraintes : Les deux perdants sont de la même équipe.
- Activité Échange : le joueur échange un Jeton contre un Ticket, un seul rôle
    - Rôle R : cardinalité = 1 ; consomme : Jeton ; produit : Ticket
- Activité Bataille : un joueur gagne un Jeton et l'autre rien, 2 rôles
    - Rôle Vainqueur : cardinalité = 1 ; consomme : Argent ; produit : Jeton
    - Rôle Perdant : cardinalité = 1 ; consomme : Argent ; produit : ø

Les Joueurs partent tous avec l'inventaire suivant : [Argent] et forme deux équipes : ($J_1$, $J_2$) et ($J_3$, $J_4$). La séquence proposée est la séquence [Bataille → Échange → AF]. Le déroulement de l'algorithme est résumé par la Figure 14. L'analyse de la dernière activité de la séquence (AF) nous donne deux combinaisons (les changements opérés par chaque combinaison sont en gras) :

- Cas où $J_1$ et $J_2$ seront les vainqueurs, ils devront avoir dans leurs inventaires requis un ticket : Inventaire requis $J_1$ = [**Ticket**] ; Inventaire requis $J_2$ = [**Ticket**] ; Inventaire requis $J_3$ = [ø] ; Inventaire requis $J_4$ = [ø] ; Utilité = 0
- Cas où $J_3$ et $J_4$ seront les vainqueurs, ils devront avoir dans leurs inventaires requis un ticket : Inventaire requis $J_1$ = [ø] ; Inventaire requis $J_2$ = [ø] ; Inventaire requis $J_3$ = [**Ticket**] ; Inventaire requis $J_4$ = [**Ticket**] ; Utilité = 0

Nous commençons par développer la première combinaison. La répétition de AF n'est pas intéressante (aucune des combinaisons n'a d'utilité positive), nous continuons donc à remonter la séquence. Nous passons à l'activité Echange. Nous obtenons deux combinaisons :

- Cas où $J_1$ échangera son jeton contre un ticket, il devra avoir dans son inventaire requis un Jeton : Inventaire requis $J_1$ = [**Jeton**] ; Inventaire requis $J_2$ = [Ticket] ; Inventaire requis $J_3$ = [ø] ; Inventaire requis $J_4$ = [ø] ; Utilité = 1
- Cas où $J_2$ échangera son jeton contre un ticket, il devra avoir dans son inventaire requis un Jeton : Inventaire requis $J_1$ = [Ticket] ; Inventaire requis $J_2$ = [**Jeton**] ; Inventaire requis $J_3$ = [ø] ; Inventaire requis $J_4$ = [ø] ; Utilité = 1

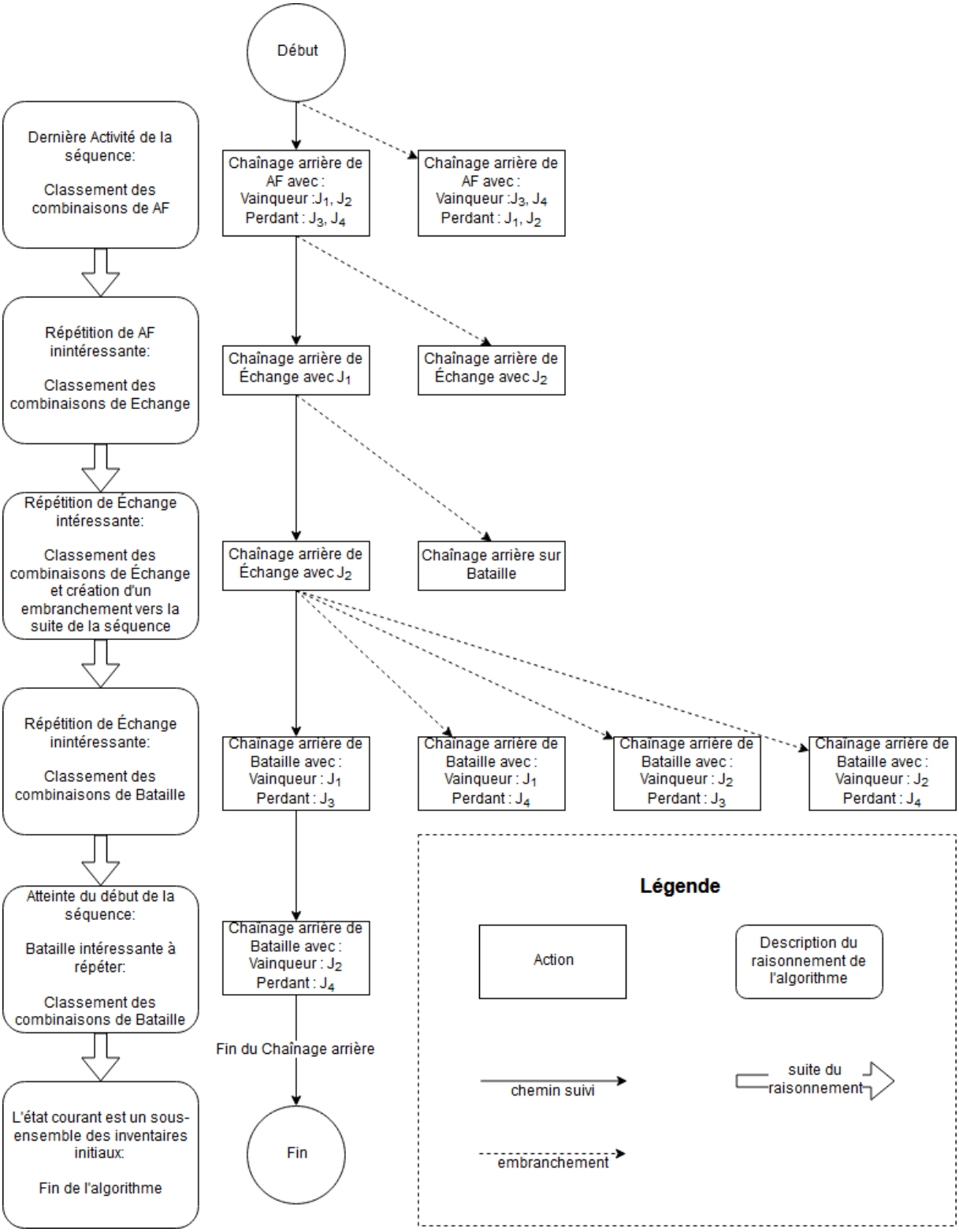


*Figure 14 : Représentation du déroulement de l'algorithme sur un exemple*

Encore une fois nous poursuivons sur la première combinaison. Cette fois-ci la répétition est intéressante (il existe une combinaison avec une utilité strictement positive). Nous créons donc un embranchement partant sur l'activité Bataille et un embranchement répétant l'activité Échange. Comme dit précédemment, en cas d'embranchement, la priorité est

donnée à la répétition. Nous répétons donc Échange qui n'offre qu'une seule combinaison intéressante :

- Cas où $J_2$ échangera son jeton contre un ticket, il devra avoir dans son inventaire requis un Jeton : Inventaire requis $J_1$ = [Jeton] ; Inventaire requis $J_2$ = [**Jeton**] ; Inventaire requis $J_3$ = [ø] ; Inventaire requis $J_4$ = [ø] ; Utilité = 1

Une fois cette combinaison exécutée, il n'y a plus d'intérêt à répéter Échange (aucune combinaison ne possède d'utilité strictement positive), nous procédons donc à l'activité Bataille. Celle-ci produit quatre combinaisons intéressantes :

- Cas où $J_1$ bat $J_3$ dans la Bataille, ils devront avoir Argent dans leurs inventaires requis : Inventaire requis $J_1$ = [**Argent**] ; Inventaire requis $J_2$ = [Jeton] ; Inventaire requis $J_3$ = [**Argent**] ; Inventaire requis $J_4$ = [ø] ; Utilité = 1
- Cas où $J_1$ bat $J_4$ dans la Bataille, ils devront avoir Argent dans leurs inventaires requis : Inventaire requis $J_1$ = [**Argent**] ; Inventaire requis $J_2$ = [Jeton] ; Inventaire requis $J_3$ = [ø] ; Inventaire requis $J_4$ = [**Argent**] ; Utilité = 1
- Cas où $J_2$ bat $J_3$ dans la Bataille, ils devront avoir Argent dans leurs inventaires requis : Inventaire requis $J_1$ = [Jeton] ; Inventaire requis $J_2$ = [**Argent**] ; Inventaire requis $J_3$ = [**Argent**] ; Inventaire requis $J_4$ = [ø] ; Utilité = 1
- Cas où $J_2$ bat $J_4$ dans la Bataille, ils devront avoir Argent dans leurs inventaires requis : Inventaire requis $J_1$ = [Jeton] ; Inventaire requis $J_2$ = [**Argent**] ; Inventaire requis $J_3$ = [ø] ; Inventaire requis $J_4$ = [**Argent**] ; Utilité = 1

Nous suivons la première combinaison. Bataille est la dernière activité de la séquence, mais nous ne sommes toujours pas arrivés à l'état initial. De fait, il faut donc vérifier qu'il soit intéressant de répéter Bataille. C'est le cas pour une unique combinaison :

- Cas où $J_2$ bat $J_4$ dans la Bataille, ils devront avoir Argent dans leurs inventaires requis : Inventaire requis $J_1$ = [Argent] ; Inventaire requis $J_2$ = [**Argent**] ; Inventaire requis $J_3$ = [Argent] ; Inventaire requis $J_4$ = [**Argent**] ; Utilité = 1

Cette combinaison étant incluse dans l'état initial (tous les joueurs ont la ressource Argent), l'algorithme se termine en retournant VRAI. Le chemin ainsi trouvé pour valider la séquence « Bataille → Échange → AF » est : Bataille ($J_2$, $J_4$) → Bataille ($J_1$, $J_3$) → Échange($J_2$) → Échange($J_1$) → AF ($J_1$, $J_2$, $J_3$, $J_4$).

## V.3. Synthèse

L'objectif principal de notre travail de thèse est d'aider à la détection de propriétés d'interactions dans un scénario de JSMJ. Pour rendre cette détection, la plus fiable possible, il faudrait calculer l'ensemble des séquences ou chemins menant les joueurs de l'état initial du scénario à l'état final. Or, selon le degré de liberté accordé aux joueurs dans un JSMJ, un grand nombre de séquences d'activités différentes peut permettre de finir le jeu. L'importance de ce nombre peut mener à une explosion combinatoire des calculs des séquences possibles d'un scénario. Pour répondre à cette problématique, nous avons travaillé des algorithmes appliquant des heuristiques permettant de calculer un sous-ensemble de séquences d'activités (SA) ayant des propriétés d'interactions représentatives de l'ensemble des

propriétés d'interactions du scénario. Du moins, nous faisons l'hypothèse que les distributions des propriétés sont similaires dans le scénario et le sous-ensemble calculé. L'expérimentation décrite dans le chapitre VII a pour objectif de vérifier cette hypothèse.

Le calcul du SA se décompose en deux grandes étapes. Similairement aux systèmes concurrents, la dynamique de notre système est reproduite par les consommations et productions de ressources de nos activités, qui établissent ainsi des liens de précédence implicites entre elles. En représentant ces liens de précédence à l'aide d'un graphe orienté, nous sommes en mesure d'établir la liste des séquences d'activités respectant ces liens, il s'agit de la sous-étape 1 de la première étape.

La sous-étape 2 consiste justement à établir cette liste. Les séquences ainsi obtenues sont qualifiées d'admissibles car elles peuvent potentiellement permettre la réalisation du scénario. Cependant celles-ci ne sont pas toutes réellement faisables à cause d'une insuffisance des ressources utilisées dans le cadre du scénario. Dans notre approche les propriétés d'interactions sont extraites des activités, par conséquent deux séquences utilisant les mêmes activités agencées différemment fournissent les mêmes interactions (en type et en nombre). Il y a donc une importante redondance d'informations dans les séquences ainsi calculées. Pour réduire la complexité, la troisième sous-étape consiste donc à supprimer toutes ces redondances de la liste des séquences admissibles en ne conservant qu'une seule et unique séquence par combinaison d'activités.

Comme évoqué précédemment, toutes les séquences ne sont pas faisables, la seconde étape du calcul du SA consiste donc à établir la faisabilité de nos séquences. Pour ce faire, notre système cherche à prouver l'existence d'un chemin qui utilise les activités de la séquence et respecte nos liens de précédences tout en permettant de finir le jeu. Si un tel chemin existe, la séquence est jugée « faisable ».

L'ensemble des séquences ainsi validées constitue l'ensemble SA permettant de qualifier un scénario à l'aide de séquences d'activité. Il est important de noter, qu'une seule séquence peut avoir plusieurs chemins faisables utilisant des quantités différentes de propriétés d'interactions. Pour des raisons calculatoires, nous ne construisons qu'un seul chemin faisable par séquence valide. Analyser les propriétés de notre séquence à partir de ce chemin unique n'aurait donc pas de sens et ne serait pas nécessairement représentatif de celle-ci. C'est pourquoi l'analyse des propriétés émergeant des séquences du SA d'un scénario se fait uniquement sur la séquence elle-même, celle-ci étant dépourvue de répétition et étant représentative de tous les chemins qu'elle peut engendrer.

# Section 3
# Évaluations et conclusions

Les chapitres de cette section font le point sur les différentes expérimentations et analyses que nous avons effectuées pour évaluer les contributions de la section 2. Dans le chapitre VI, nous mettons à l'épreuve la modélisation des scénarios et activités décrit dans le chapitre III sur quatre jeux sérieux d'apprentissage afin de mesurer les limitations de notre modèle.

Le chapitre VII présente les différentes analyses et tests menés en rapport avec MP-LOG, un jeu sérieux sur la logique propositionnelle. Deux analyses y sont présentées. Le principe de la première consiste à extraire automatiquement des propriétés d'interactions de traces de joueurs et à les comparer aux résultats d'un sondage sur les interactions auprès des étudiants. L'objectif que nous poursuivons, à travers cette analyse, est d'évaluer la pertinence des propriétés d'interaction proposées en vérifiant que celles-ci soient corrélées aux interactions déclarées par les apprenants et les observations à partir des actions du jeu. La deuxième analyse effectue la comparaison entre les propriétés extraites automatiquement des traces et les propriétés extraites automatiquement du scénario. Cette analyse nous permet ainsi de vérifier si nous sommes en mesure de prévoir les interactions auxquelles auront accès les apprenants dans un scénario de JSMJ.

Pour finir le chapitre VIII présente l'ensemble de nos conclusions sur ce travail de thèse, et présente également les différentes perspectives de nos contributions.

# VI. Mise à l'épreuve des modèles de scénario et d'activité

Pour formaliser les modèles de scénario et d'activité de JSMJ que nous proposons, nous avons créé une ontologie et l'avons implémentée à l'aide du logiciel Protégé[6]. Le schéma de l'ontologie a déjà été présenté (cf. la section III.3 - Figure 8) et s'articule autour des concepts détaillés dans le chapitre III. Une fois l'ontologie mise au point, nous avons cherché à la mettre à l'épreuve sur quatre scénarios de jeux Sérieux différents *Voracy Fish, ByteBattle, ClassCraft* et *Learning Adventure*.

Nous l'avons vu dans le chapitre III, ces quatre jeux possèdent des structures scénaristiques différentes et, de plus, ne traitent pas nécessairement les interactions de la même manière. De fait, nous pensons que la modélisation de ces jeux constitue une base intéressante pour mettre à l'épreuve notre modèle.

Cette mise à l'épreuve constitue le sujet de ce chapitre et est divisée en différentes sections. Chacune d'elles détaille la modélisation d'un de ces jeux en mettant l'accent sur les différentes particularités et difficultés auxquelles nous avons fait face et les solutions que nous avons apportées pour y répondre.

## VI.1. Modélisation d'un scénario de *ByteBattle*

Dans le scénario de *ByteBattle* étudié, les joueurs étaient divisés en deux poules. Le binôme de joueurs ayant obtenu le plus de Victoire dans sa poule de départ était ensuite conduit en demi-finale contre le deuxième de la poule opposée. Les gagnants de chacune des demi-finales s'affrontaient ensuite en finale. Autrement dit, pour un jeu compétitif comme *ByteBattle*, il est nécessaire de distinguer à l'avance le rôle qui gagnera du rôle qui perdra.

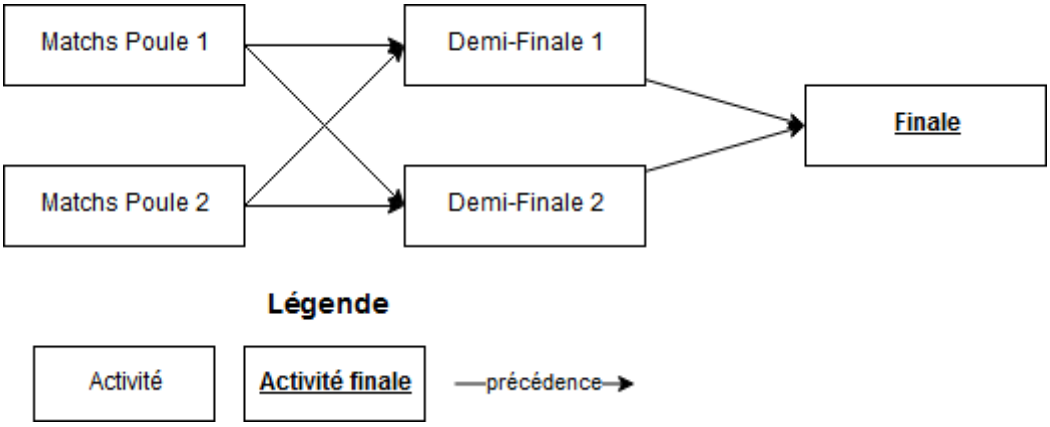


*Figure 15 : Graphe de précédence de ByteBattle*

Dans le cadre d'une utilisation de l'ontologie pour modéliser les actions réalisées par les joueurs, le vainqueur est déjà connu (grâce aux actions effectuées) et peut être associé sans

[6] https://protege.stanford.edu/

problème au rôle prototypique gagnant. Pour ce qui est du chaînage arrière, seules les ressources produites et consommées importent, les rôles prototypiques simulées en chaînage arrière établissant les actions qu'effectuerait les joueurs dans le cas simulé. D'un point de vue scénarisation, cette répartition des rôles a également du sens : un match a lieu entre deux joueurs à l'issue duquel nous avons un perdant et un gagnant.

Comme stipulé dans le scénario, les joueurs sont séparés en deux Poules. Aussi les joueurs de la Poule 1 ne peuvent affronter les joueurs de la Poule 2. Une solution serait de représenter chaque Match par une activité (J1 vs J2, J1 vs J3, etc…) et d'associer aux joueurs des ressources utilisées par le rôle et servant à les identifier (ou à identifier leur équipe d'appartenance). Une telle solution peut cependant vite devenir problématique lorsque le nombre de joueurs augmente. D'autant plus qu'il est alors nécessaire de dédoubler chaque activité pour déterminer le vainqueur.

*Tableau 3 : Activité Match Poule 1*

| Match Poule 1 | | | |
|---|---|---|---|
| Rôle prototypique | Premier | Second | Perdant |
| Description | Le Premier est le joueur ayant obtenu le meilleur score dans les matchs de la poule. | Le Second de la poule obtient le droit de passer en demi-finale mais échoue à être premier. | Les Joueurs ayant échoués au Match de Poule |
| Cardinalité | 1 | 1 | 2 |
| Contraintes | Ø | Ø | Ø |
| Equipes | A | B | Différentes de A et B |
| Consomme | « Poule 1 » | « Poule 1 » | « Poule 1 » |
| Produit | « Demi 1 » | « Demi 2 » | Ø |
| Requiert | Ø | Ø | Ø |
| Connaissances Ciblées | « Mise en pratique d'une bibliothèque de fonctions informatiques » | « Mise en pratique d'une bibliothèque de fonctions informatiques » | Ø |
| Objectifs | Produire « Demi 1 » (objectif réussi) | Produire « Demi 1 » -> (objectif échoué) | Produire « Demi 1 » (objectif échoué) |

L'activité « Match Poule 1 » (cf. Tableau 3) indique la solution que nous avons adoptée, celle-ci consiste à abstraire l'ensemble des matchs de la première Poule en une unique activité.

Seule l'identification de la poule d'appartenance des joueurs est nécessaire. D'où la consommation de la ressource « Poule 1 » présente initialement dans l'inventaire des joueurs. L'activité « Match Poule 2 » suit le même schéma.

La raison pour laquelle il nous est nécessaire de différencier les deux matchs de Poule, est que le deuxième de chaque Poule passe en demi-finale pour affronter le premier de la poule opposée. Il était donc nécessaire de différencier le deuxième de la poule 1 du deuxième de la poule 2 et de différencier le premier de la poule 1 du premier de la poule 2.

*Tableau 4 : Activité Finale*

| Finale | | | |
|---|---|---|---|
| Rôle prototypique | Vainqueur | Second | Perdants |
| Description | Le gagnant de la Finale | Le perdant de la Finale | Ceux ayant été éliminés avant la finale. |
| Cardinalité | 1 | 1 | 6 |
| Contraintes | Ø | Ø | Ø |
| Equipes | A | B | Différentes de A et B |
| Consomme | « Ticket pour la Finale » | « Ticket pour la Finale » | Ø |
| Produit | « Premier » | « Second » | Ø |
| Requiert | « Mise en pratique d'une bibliothèque de fonctions informatiques » | « Mise en pratique d'une bibliothèque de fonctions informatiques » | Ø |
| Connaissances Ciblées | « Application d'une stratégie gagnante à l'aide de fonctions » | Ø | Ø |
| Objectifs | Produire « Premier » (objectif réussi) | Produire « Premier » (objectif échoué) | Produire « Premier » (objectif échoué) |

Selon les principes évoqués plus tôt dans cette section, les activités Demi-Finale 1 et 2, n'ont pas réellement posées de problèmes : deux joueurs identifiés par leurs ressources, un gagnant et un perdant. Le fonctionnement de l'activité « Finale » (cf. Tableau 4) est légèrement différent puisqu'aux rôles de Vainqueur et de Perdant de la Finale (noté Vainqueur et Second), nous ajoutons un rôle de spectateur pour les joueurs déjà éliminés afin de symboliser la fin de

leur scénario et rappeler leur objectif de départ : gagner la finale, autrement dit produire la ressource « Premier ».

*Tableau 5 : Propriétés étant apparues dans la modélisation de ByteBattle*

| Propriétés | Conflit Interne | Conflit Externe | Validation Commune | Construction Commune des Connaissances | Individualisme |
|---|---|---|---|---|---|
| Détection | Non | Oui | Non | Non | Non |

Nous avons appliqué la formalisation du chapitre IV sur les activités que nous venons de décrire et d'autres. Nous résumons dans le Tableau 5 les propriétés étant apparues dans le scénario. Nous constatons donc que *ByteBattle* est un jeu purement compétitif car seul le conflit externe est extrait du scénario tel que nous l'avons modélisé.

# VI.2. Modélisation d'un scénario de *VoracyFish*

Dans *VoracyFish*, les joueurs contrôlent un poisson dans un océan à l'aide d'une *Wii Mote* . Ce jeu a pour objectif d'aider des patients à récupérer la motricité qu'ils ont perdue suite à un AVC. Dans le scénario que nous avons décidé de modéliser, les joueurs doivent s'entre-manger en évitant des obstacles. Le dernier poisson en vie gagne la partie. Ce scénario cible principalement des compétences motrices et les joueurs peuvent réussir à manger leurs adversaires par pure chance sans forcément acquérir dans le jeu ces compétences. Le jeu implique 8 joueurs dont la caractéristique principale au début du jeu est qu'ils sont en vie. Ces 8 joueurs possèdent un inventaire initial identique possédant une unique ressource « Vie » symbolisant cet état de fait.

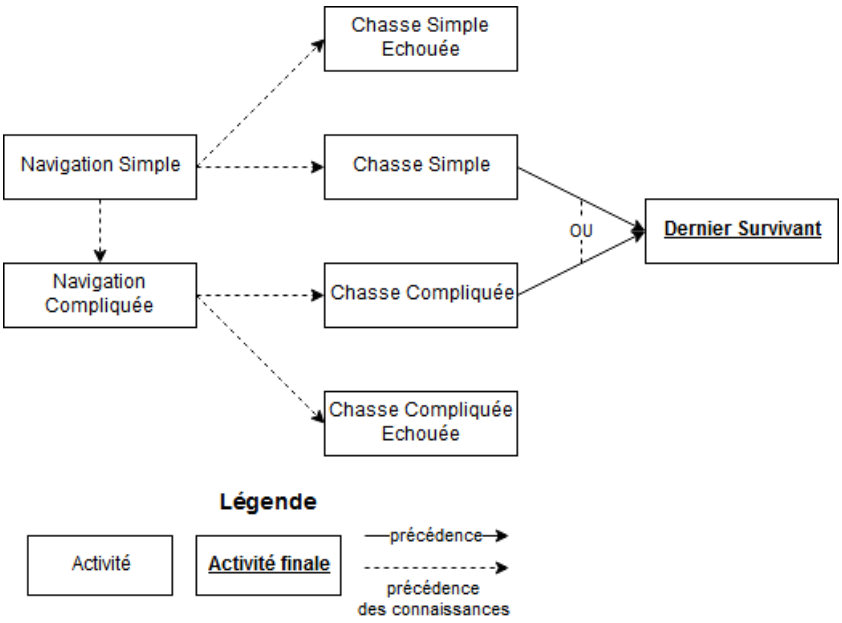


*Figure 16 : Graphe de précédence de VoracyFish*

Pour représenter la différence de difficulté entre le fait de naviguer avec ou sans obstacle, nous avons créé six activités différentes (cf. Figure 16) :

- Navigation Simple : le joueur travaille la motricité basique en naviguant sans obstacles
- Navigation Compliquée : le joueur utilise la motricité basique pour travailler la motricité avancée afin de naviguer au travers d'obstacles.
- Chasse Simple : un joueur mange un autre dans une zone dépourvue d'obstacles
- Chasse Compliquée : un joueur mange un autre dans une zone avec des obstacles
- Chasse Simple Échouée : un joueur pourchasse un autre qui arrive à s'échapper dans une zone dépourvue d'obstacles
- Chasse Compliquée Échouée : un joueur pourchasse un autre qui arrive à s'échapper dans une zone avec des obstacles

Les compétences ne sont pas un frein à la réalisation des activités, mais nous avons trouvé intéressant d'indiquer sur la Figure 16 les liens de précédence qui auraient été créés en prenant en compte les compétences à l'aide d'arcs orientés en pointillés. Les joueurs pouvant rester dans des zones dépourvues d'obstacles ou au contraire ne jamais y aller, il est possible de finir le scénario en utilisant à de multiples reprises une seule de nos activités de chasse. L'activité « Dernier Survivant » permet de caractériser la fin du jeu.

*Tableau 6 : Activité Chasse Simple Échouée*

| Chasse Simple Échouée | | |
|---|---|---|
| Rôle prototypique | Prédateur | Proie |
| Description | Le Prédateur pourchasse la proie dans un environnement sans obstacle. | La Proie s'échappe. |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Vie » | « Vie » |
| Produit | « Vie » | « Vie » |
| Requiert | « Motricité Basique » | « Motricité Basique » |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | La Proie doit produire « Mort » (objectif échoué) | Ne pas produire « Mort » (objectif réussi) |

Contrairement à *ByteBattle* qui nécessitait d'identifier les Joueurs ou tout du moins leurs poules d'appartenance, *Voracy Fish* n'a pas ce problème et traite tous les joueurs sans distinction. Dans le jeu, les poissons ne peuvent manger que des poissons plus petits qu'eux.

Toutefois en suivant un raisonnement similaire aux rôles Vainqueur et Perdant de *ByteBattle*, il est possible d'abstraire cette notion avec le rôle de Proie et de Prédateur (cf. Tableau 6).

*Tableau 7 : Activité Navigation Simple*

| Navigation Simple | |
|---|---|
| Rôle prototypique | Navigateur |
| Description | Le poisson nage librement. |
| Cardinalité | 1 |
| Contraintes | Ø |
| Equipes | Ø |
| Consomme | « Vie » |
| Produit | « Vie » |
| Requiert | Ø |
| Connaissances Ciblées | « Motricité Basique » |
| Objectifs | Connaissances Ciblées « Motricité Basique » (objectif réussi) |

La Tableau 7 présente l'activité Navigation Simple. Cette activité est mono-joueur, la seule propriété pouvant y être détectée est donc « Individualiste ». Ce genre d'activité est intéressant à modéliser à l'échelle du scénario car elles permettent de représenter les démarches individuelles des joueurs dans ce dernier. Ces activités, qui dénotent d'une absence d'interactions, sont parfois indispensables pour faire évoluer le jeu à travers des transformations de ressources et des ciblages de connaissances.

Le dernier Joueur en vie (qui possède la ressource Vie) remporte la partie. Dans le cadre de l'activité « Dernier Survivant », nous symbolisons cela par la consommation de la ressource « Vie » chez le dernier joueur et la ressource « Mort » (identifiant ceux qui ont été proie dans des activités de chasse réussies) chez tous les autres joueurs. La ressource « Mort » n'étant produite qu'une fois la ressource « Vie » d'un joueur consommée, il est impossible d'avoir deux joueurs satisfaisant les conditions du rôle prototypique Gagnant (tous les autres satisfaisant les contraintes du rôle Perdant). La cardinalité de 7 indiqué pour le rôle du Perdant est fixée par rapport au scénario. Si plus ou moins de 8 joueurs était impliqués dans le scénario, il serait nécessaire d'adapter la cardinalité de cette activité.

*Tableau 8 : Activité Dernier Survivant*

| Dernier Survivant | | |
|---|---|---|
| Rôle prototypique | Gagnant | Perdant |
| Description | Le dernier Joueur en vie gagne la partie. | Le Joueur a été mangé plus tôt dans la partie. |
| Cardinalité | 1 | 7 |
| Contraintes | Ø | Ø |
| Equipes | A | Différente de A |
| Consomme | « Vie » | « Mort » |
| Produit | « Victoire » | « Défaite » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Victoire » (objectif réussi) | Produire « Victoire » (objectif échoué) |

L'analyse rapportée dans le Tableau 9 des activités de *Voracy Fish* nous indique que ce jeu est compétitif. En effet seule deux propriétés ont été extraites de ce scénario : le conflit externe et l'individualisme. En effet, le conflit externe est un marqueur de compétition et l'individualisme un marqueur d'absence d'interactions.

*Tableau 9 : Propriétés étant apparues dans la modélisation de Voracy Fish*

| Propriétés | Conflit Interne | Conflit Externe | Validation Commune | Construction Commune des Connaissances | Individualisme |
|---|---|---|---|---|---|
| Détection | Non | Oui | Non | Non | Oui |

# VI.3. Modélisation d'un scénario de *ClassCraft*

Dans *ClassCraft*, les ressources acquises par les joueurs sont les points d'expériences (XP). Pour acquérir ces ressources, les joueurs doivent répondre à des questions de quizz. Ces questions sont posées à des groupes de quatre auxquels les joueurs appartiennent. Les réponses se font individuellement et seul le joueur donnant une bonne réponse est récompensé. Dans le jeu originel, les joueurs peuvent assumer différentes classes leurs donnant accès à des pouvoirs variés spécifiques à leur classe. Ces pouvoirs sont de deux

natures : les pouvoirs individuels et les pouvoirs de groupe. Lors de son utilisation, un pouvoir individuel ne profite qu'au joueur l'ayant utilisé au contraire des pouvoirs de groupe qui profitent au groupe dans son ensemble.

Pour pouvoir utiliser leurs pouvoirs, les joueurs doivent utiliser des points de Mana (MP) qu'ils gagnent en dépensant des points d'XP. Les MP ne sont pas la seule chose achetable à l'aide des XP. En effet, les pouvoirs eux-mêmes nécessitent de dépenser de l'XP pour être achetés, tout comme les augmentations de niveau chez les joueurs (ils partent du niveau 1). Dans le cadre de notre scénario, un classement était effectué à la fin de l'année : le groupe de joueurs ayant le niveau le plus élevé gagnait le jeu (le niveau d'une équipe étant la somme du niveau de ses membres).

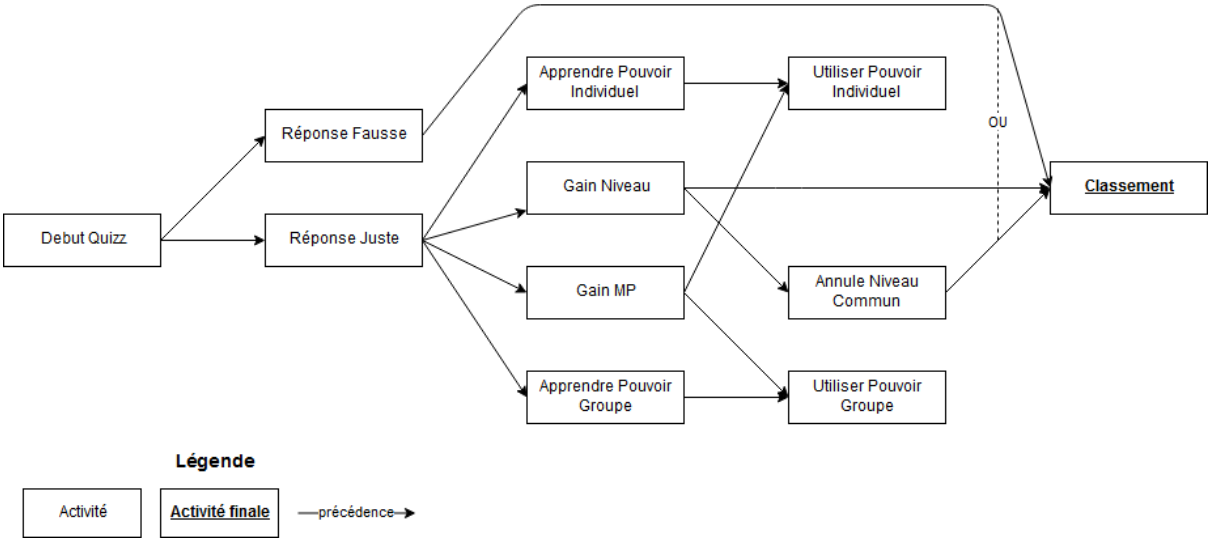


*Figure 17 : Graphe de précédence de ClassCraft*

Rappelons-le, nous nous intéressons aux interactions. La différenciation des divers pouvoirs et classes n'était donc pas intéressante dans notre cas. C'est pourquoi nous avons décidé de les uniformiser selon leurs types (groupe et individuel). Les activités liées à ces pouvoirs sont donc au nombre de quatre (cf. Figure 17) :

- Apprendre un Pouvoir Individuel
- Apprendre un Pouvoir de Groupe
- Utiliser un Pouvoir Individuel
- Utiliser un Pouvoir de Groupe

Dans *ClassCraft*, la quantité d'expérience que les joueurs peuvent gagner peut varier en fonction de la question à laquelle ils répondent. Dans notre cas, seule la symbolique des points de XP était importante, nous avons donc pu abstraire l'ensemble des activités de réponses en deux types : les réponses fausses (sans gain de XP) et les réponses justes (avec gain de XP). Les joueurs ne peuvent répondre qu'aux questions posées à leur groupe, pour modéliser cela, nous avons décidé de dédoubler les activités de quiz et de réponses en fonction du nombre de groupe disponible (ici nous avons deux groupes).

Notre scénario de *ClassCraft* implique 8 joueurs divisés en deux groupes. Chaque Groupe possède une unique ressource Quiz permettant de lancer le jeu pour ce groupe. Ainsi l'état initial du scénario se décrit comme suit :

- Equipe « Groupe 1 » : J1, J2, J3, J4 ; Inventaire partagé : [Quizz]
- Equipe « Groupe 1 » : J5, J6, J7, J8 ; Inventaire partagé : [Quizz]

L'inventaire de nos 8 joueurs est vide à l'état initial.

*Tableau 10 : Activité Classement*

| Classement | | |
|---|---|---|
| Rôle prototypique | Vainqueurs | Perdants |
| Description | Il reste un niveau dans cette équipe après les activités de traitements, ils ont donc gagné. | Il ne reste aucun niveau dans cette équipe après les activités de traitements, ils ont donc perdu. |
| Cardinalité | 4 | 4 |
| Contraintes | Niveau de l'équipe A > Niveau de l'équipe B | Ø |
| Equipes | A | B |
| Consomme | « Question effectuée » * 16 dans l'inventaire de l'équipe A | « Question effectuée » * 16 dans l'inventaire de l'équipe B |
| Produit | « Victoire » | « Défaite » |
| Requiert | « Connaissances sur la littérature française » | « Connaissances sur la littérature française » |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Victoire »<br>(objectif réussi) | Produire « Victoire »<br>(objectif échoué) |

La plupart des activités préalablement évoquées ne posent pas de problème quant à leur modélisation. Il existe toutefois une activité qui pose une problématique majeure quant à notre algorithme, l'activité de Classement. Dans le cadre du scénario de *ClassCraft* que nous avons étudié, le but d'un groupe est d'avoir un plus haut niveau que le groupe auquel il est opposé. Ceci est représenté par la contrainte « Niveau de l'équipe A > Niveau de l'équipe B » dans le Tableau 10. Cependant, notre algorithme en chaînage arrière n'est pas en mesure d'exploiter de telles contraintes. En effet, l'algorithme ne garde pas en mémoire les ressources produites n'impactant pas les inventaires requis (ici les niveaux) et ne peut garantir que l'inventaire B soit inférieure à A dans le chemin testé. En revanche notre algorithme peut

empêcher une ressource d'être présente dans un inventaire en fin de parcours. C'est pourquoi, nous avons mis au point une procédure automatisable permettant de transformer ces conditions. Pour ce faire, nous établissons des activités que nous appelons activités de traitement.

*Tableau 11 : Activité Annule niveau Commun*

| Annule niveau Commun | | |
|---|---|---|
| Rôle prototypique | Groupe 1 | Groupe 2 |
| Description | Traitement : nous retirons un niveau à chaque groupe | Traitement : nous retirons un niveau à chaque groupe |
| Cardinalité | 4 | 4 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Niveau » et « Question effectuée » * 16 dans l'inventaire de l'équipe « A » | « Niveau » et « Question effectuée » * 16 dans l'inventaire de l'équipe « B » |
| Produit | « Question effectuée » * 16 dans l'inventaire de A | « Question effectuée » * 16 dans l'inventaire de B |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Ø | Ø |

L'activité « Annule niveau Commun » (cf. Tableau 11) est une activité de traitement. Elle n'apporte aucune interaction et n'est pas utilisée par les joueurs dans le cadre du Jeu. La condition posant problème se base sur la ressource « Niveau » et Classement n'est accessible qu'une fois les 16 questions répondues par chacun des deux groupes.

Soit a et b les quantités des deux niveaux de nos équipes. Soit c l'écart de niveau entre nos deux équipes. Si a>b nous avons alors c = a-b et donc b+c > b. Le but de l'activité de traitement est de réduire le niveau de l'une de nos deux équipes à 0 afin de pouvoir transformer la contrainte « a>b » en « c>0 ». Cette deuxième contrainte nous donne deux informations : il existe au moins une ressource Niveau dans l'inventaire de l'équipe A et il n'y en a aucune dans l'inventaire de l'équipe B. Or, comme nous l'avons indiqué plus haut, notre algorithme est en mesure de vérifier ces deux informations. L'activité « Annule niveau Commun » enlève une ressource « Niveau » dans l'inventaire des deux équipes à chacune de ses utilisations. Ainsi, en répétant « b » fois cette activité, nous transformons dans les faits notre contrainte « a>b »

en « c>0 » et établissons par la même le niveau maximum de l'équipe Perdante dans l'algorithme au nombre de répétition de l'activité de traitement.

L'activité Classement subit également des modifications en conséquence de la création de l'activité de traitement (cf. Tableau 12). Si c>0, cela signifie que les Vainqueurs ont au moins une ressource « Niveau » dans leur inventaire. « Niveau » est donc ajouté aux consommations de « Vainqueurs ». Pour ce qui est de la contrainte sur les Niveaux, il suffit désormais d'indiquer qu'il ne peut y avoir de « Niveau » dans l'inventaire des « Perdants » en place de la contrainte originelle.

*Tableau 12 : Activité Classement modifiée*

| Classement | | |
|---|---|---|
| Rôle prototypique | Vainqueurs | Perdants |
| Description | Il y a au moins une ressource « Niveau » dans cette équipe après les activités de traitements, ils ont donc gagné. | Pas de ressource « Niveau » dans cette équipe après les activités de traitements, ils ont donc perdu. |
| Cardinalité | 4 | 4 |
| Contraintes | Ø | Pas de « Niveau » dans l'inventaire de l'équipe B |
| Equipes | A | B |
| Consomme | « Niveau » et « Question effectuée » * 16 dans l'inventaire de l'équipe A | « Question effectuée » * 16 dans l'inventaire de l'équipe B |
| Produit | « Victoire » | « Défaite » |
| Requiert | « Connaissances sur la littérature française » | « Connaissances sur la littérature française » |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Victoire » (objectif réussi) | Produire « Victoire » (objectif échoué) |

Notre dernière activité permet de consacrer l'équipe ayant obtenu le plus haut niveau et est donc une activité compétitive se basant sur un conflit Externe.

*Tableau 13 : Propriétés étant apparues dans la modélisation de ClassCraft*

| Propriétés | Conflit Interne | Conflit Externe | Validation Commune | Construction Commune des Connaissances | Individualisme |
|---|---|---|---|---|---|
| Détection ? | Oui | Oui | Oui | Non | Oui |

Le Tableau 13 nous indique que quatre propriétés ont été extraites du scénario de ClassCraft que nous avons étudié. Le Conflit externe vient du classement final entre nos différentes équipes. Ce classement est basé sur le niveau global d'une équipe et les gains de niveaux sont basés sur les points d'expériences remportés par les joueurs. Ainsi, dans l'activité « Apprendre Pouvoir Individuel » et « Utiliser Pouvoir Individuel », le joueur utilise ses points d'expériences pour son seul profit, en lieu et place de l'utiliser pour améliorer le classement global de l'équipe. Nous sommes donc en présence d'un conflit interne. L'individualisme vient des activités gains de MP, Réponse Juste et Réponse Fausse (voir Figure 17) qui n'impliquent chacune qu'un seul joueur sans que cela ait de conséquences sur les autres. L'utilisation d'un pouvoir de groupe révèle la propriété validation commune.

## VI.4. Modélisation d'un scénario de *Learning Adventure*

Dans ce scénario, les joueurs sont divisés en trois groupes qui évoluent en parallèle sur trois ensembles d'activités. Le premier groupe cherche à obtenir les schémas d'un circuit imprimé. Le deuxième groupe cherche les composants dudit circuit. Et enfin le dernier cherche le *SoftWar*e à installer. Le but de ces trois groupes est de construire un Arduino et pour ce faire, chaque groupe doit récupérer, le *SoftWare*, les composants et le schéma de celui-ci. Fait important, il est interdit aux joueurs d'un groupe de commencer l'un des ensembles d'activités préalablement cités, tant que le groupe l'effectuant actuellement n'a pas terminé. Autrement dit, si un groupe a fini de récupérer les schémas et veut récupérer les composants, les joueurs de ce groupe doivent attendre que le groupe qui est en train de récupérer les composants ait fini.

Chacune des tâches confiées aux joueurs se divisent en plusieurs parties :

- La recherche des schémas est initiée par un Personnage Non Joueur (PNJ) appelé Turing, celui-ci indique aux joueurs quel schéma chercher. Ils cherchent alors le schéma dans l'environnement virtuel. Une fois celui-ci trouvé, ils consultent un PNJ robot qui leur indique si leur schéma est valide ou non. Ils doivent par la suite analyser et étudier le schéma sur une table virtuelle dans la salle de classe.
- Les composants nécessaires à la fabrication de l'Arduino sont indiqués par un PNJ appelé Richie. Une fois ces informations collectées, les étudiants cherchent dans l'environnement ludique les composants indiqués pour ensuite y avoir accès dans la salle de classe.

- La récupération du *SoftWare* suit un schéma identique à celui de la recherche des composants, au détail près que le PNJ leur prodiguant les informations voulues s'appelle Bishop.

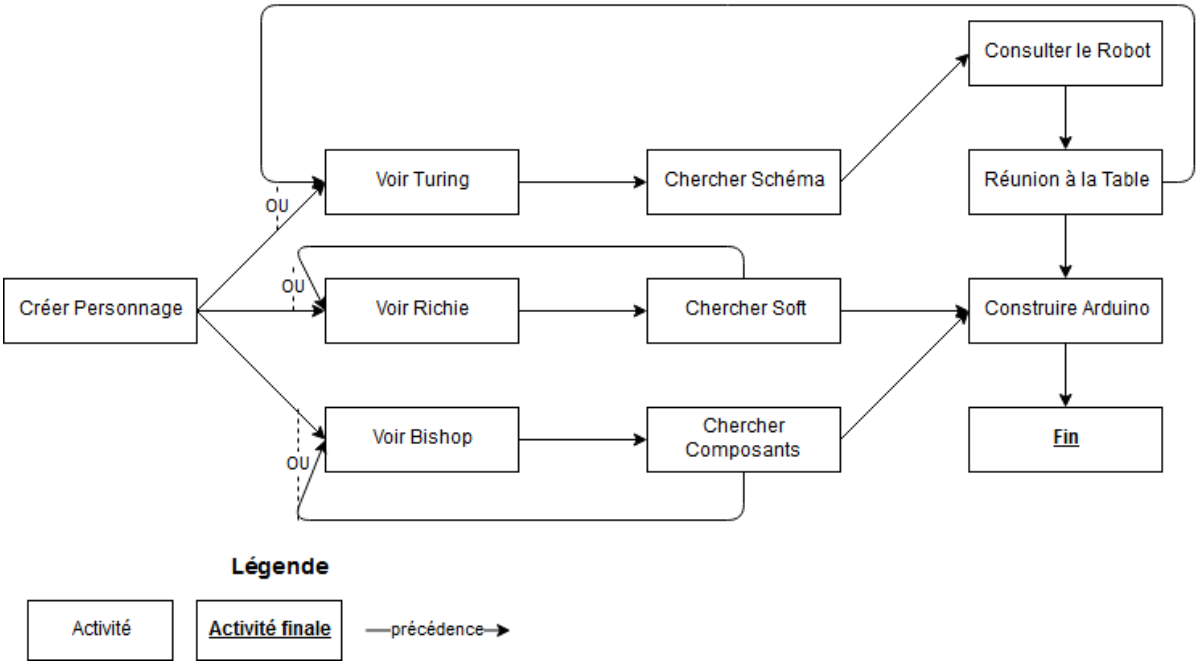


*Figure 18 : Graphe de précédence de LearningAdventure*

Lors de leur entrée en jeu, les joueurs créent chacun leur personnage. L'activité « Créer Personnage » (cf. Figure 18) qui en résulte est la première du scénario. Chaque joueur part ensuite voir le PNJ associé à son groupe grâce aux trois activités :

- Voir Turing
- Voir Richie
- Voir Bishop

Les joueurs suivent ensuite les indications fournies jusqu'à la fin de leur tâche. La première activité d'une Tâche (Voir X) consomme une ressource globale dans un inventaire partagé par tous les joueurs afin d'empêcher qu'un autre groupe n'entreprenne cette tâche. Cette ressource n'est reproduite dans l'inventaire partagé qu'une fois la dernière activité de la tâche concernée effectuée. De fait, il est donc nécessaire d'établir un lien de précédence entre la dernière et la première activité d'une tâche. La présence d'un lien OU conjointement à l'activité « Créer Personnage », permet cependant de traiter tous les problèmes de cycle éventuels qui pourraient apparaître (cf. règles de construction de la structure d'états, section V.1.b ). Une fois que l'ensemble des groupes ont construit leur Arduino, le jeu prend fin.

Le jeu se divise en trois groupes de quatre joueurs. Aucun de ces groupes ne possède de ressource dans l'inventaire partagé. Chaque Joueur possède une ressource « Initialisation » lui permettant de créer un Personnage.

*Tableau 14 : Activité Création de Personnage*

| Créer Personnage | |
|---|---|
| Rôle prototypique | Elève |
| Description | Les élèves créent leurs personnages et apprennent qu'ils vont construire un Arduino |
| Cardinalité | 12 |
| Contraintes | Ø |
| Equipes | Ø |
| Consomme | « Initialisation » |
| Produit | « Turing Disponible », « Richie Disponible », « Bishop Disponible », « Quête Schéma », « Quête Composant », « Quête Soft », « Disponible » |
| Requiert | Ø |
| Connaissances Ciblées | Ø |
| Objectifs | Ø |

L'activité « Créer Personnage » permet d'initialiser le Jeu en donnant à chaque Joueur les ressources dont il aura besoin. Elle permet d'éviter les cycles dans le scénario de *Learning Adventure* causé par la nécessité d'empêcher les joueurs d'accéder à des activités en cours d'utilisation par un autre groupe (voir le lien de précédence entre « Réunion Table » et « Voir Turing » sur la Figure 18 par exemple).

*Tableau 15 : Activité Voir Turing*

| Voir Turing | | |
|---|---|---|
| Rôle prototypique | Elèves | Observateurs |
| Description | Les élèves vont voir Turing, il leur apprend où chercher le schéma. | Les autres élèves ne peuvent aller voir Turing tant que la quête des schémas n'est pas finie |
| Cardinalité | 4 | 8 |
| Contraintes | Ø | Ø |
| Equipe | A | Différent de A |
| Consomme | « Turing Disponible » et « Disponible » et « Quête Schéma » | « Turing Disponible » |
| Produit | « Localisation Schéma » | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Localisation Schéma » (objectif réussi) | Ø |

En allant voir Turing, les joueurs se focalisent sur la quête du schéma et ne sont donc plus disponibles pour effectuer d'autres activités avant d'avoir fini toutes les activités liées au schéma. De plus, il leur est interdit de recommencer ces activités après coup, d'où la consommation de la ressource « Quête Schéma » qui ne leur sera pas rendu par la suite. De plus, Turing est rendu indisponible auprès de tous les autres joueurs. La description des activités « Voir Richie » et « Voir Bishop » est similaire.

*Tableau 16 : Activité Chercher Schéma*

| Chercher Schéma | |
|---|---|
| Rôle prototypique | Elève |
| Description | Les élèves cherchent le schéma |
| Cardinalité | 1 |
| Contraintes | Ø |
| Equipes | Ø |
| Consomme | « Localisation Schéma » |
| Produit | « Schéma non vérifié » |
| Requiert | Ø |
| Connaissances Ciblées | « Identifier un Schéma » |
| Objectifs | Produire « Schéma non vérifié » (objectif réussi) |

*Tableau 17 : Activité Consulter le Robot*

| Consulter le Robot | |
|---|---|
| Rôle prototypique | Elève |
| Description | Le Robot vérifie que 4 schémas aient été trouvé par les élèves |
| Cardinalité | 4 |
| Contraintes | Ø |
| Equipes | Ø |
| Consomme | « Schéma non vérifié » |
| Produit | « Schéma vérifié » |
| Requiert | ø |
| Connaissances Ciblées | ø |
| Objectifs | Produire « Schéma vérifié » (objectif réussi) |

L'activité « Chercher Schéma » demandait au joueur de chercher un schéma dans l'environnement de jeu. Une fois 4 schémas trouvés, l'activité « Consulter le Robot » leur permet d'accéder à la salle contenant la table leur permettant de sélectionner l'unique schéma valide parmi les 4 à travers l'activité « Réunion Table ». Ces activités symbolisent les différentes étapes auxquelles les élèves ont été confrontés. La progression occasionnée par ces activités est linéaire.

*Tableau 18 : Activité Réunion Table*

| Réunion Table | | |
|---|---|---|
| Rôle prototypique | Elèves | Observateurs |
| Description | Les élèves obtiennent leur schéma et libère la quête pour les observateurs | Les autres élèves peuvent de nouveau aller voir Turing |
| Cardinalité | 4 | 8 |
| Contraintes | Ø | Ø |
| Equipe | A | Différent de A |
| Consomme | « Schéma vérifié » | Ø |
| Produit | « Turing Disponible » et « Disponible » et « Schéma valide » | « Turing Disponible » |
| Requiert | « Identifier un Schéma » | Ø |
| Connaissances Ciblées | « Lire et vérifier la validité d'un Schéma » | Ø |
| Objectifs | Produire « Schéma valide » (objectif réussi) | Ø |

La réunion à la table concrétise la construction du schéma et fournit l'un des trois éléments nécessaires à la construction de l'Arduino. De plus, il libère également les activités liées au schéma pour les autres groupes qui peuvent dorénavant accéder à celles-ci à travers l'activité « Voir Turing ». Le fonctionnement est identique pour le Software et les composants. « Construire Arduino » ne peut être effectué qu'une fois les trois éléments composant celui-ci réunis. Cette activité concrétise le scénario pour l'un des trois groupes évoluant dans le jeu. Une fois que nos trois groupes ont créé leur Arduino, il est possible de mettre Fin au Jeu (cf. Tableau 20 ). Dans notre cas, nous avons considéré que les joueurs ne cherchaient pas à finir leur Arduino en premier. Par conséquent l'activité de Fin n'est pas compétitive. Cela aurait été rendu possible en s'inspirant des fins des trois jeux précédemment décrits et en divisant l'activité « Construire Arduino » en trois (un pour chaque rang).

*Tableau 19 : Activité Construire Arduino*

| Construire Arduino | |
|---|---|
| Rôle prototypique | Elève |
| Description | Les élèves ont récupéré tous les éléments de l'Arduino |
| Cardinalité | 4 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Schéma valide » et « Soft valide » et « Composants valides » |
| Produit | « Arduino » |
| Requiert | « Vérifier la validité d'un Schéma » et « Identification d'un Software » et « Identification de Composants » |
| Connaissances Ciblées | « Construction d'une manette de jeu » |
| Objectifs | Produire « Arduino » (objectif réussi) |

*Tableau 20 : Activité Fin*

| Fin | |
|---|---|
| Rôle prototypique | Elève |
| Description | Tous les groupes ont construit leur Arduino |
| Cardinalité | 12 |
| Contraintes | Ø |
| Equipes | Ø |
| Consomme | « Arduino » |
| Produit | Ø |
| Requiert | « Construction d'une manette de jeu » |
| Connaissances Ciblées | Ø |
| Objectifs | Ø |

À travers Learning Adventure, les joueurs doivent collaborer en groupe pour construire un Arduino. Comme précisé lors de l'activité « Fin », nous considérons ici que les différents groupes n'essaient pas de concourir avec les autres pour savoir qui serait le premier à construire l'Arduino. De fait, le Tableau 21 indique trois propriétés apparentes dans le scénario : la Construction Commune des Connaissances, la Validation Commune et l'Individualisme.

*Tableau 21 : Propriétés étant apparues dans la modélisation de Learning Adventure*

| Propriétés | Conflit Interne | Conflit Externe | Validation Commune | Construction Commune des Connaissances | Individualisme |
|---|---|---|---|---|---|
| Détection | Non | Non | Oui | Oui | Oui |

## VI.5. Analyse et Synthèse

Chaque scénario étudié possédait ses propres difficultés. Dans *Learning Adventure*, la structure du scénario interdisait à un groupe de commencer une activité d'une branche du scénario tant qu'un autre groupe y était. Pour résoudre ce problème, nous nous sommes appuyés sur la règle de notre modèle stipulant qu'une activité ne peut être commencée ou effectuée si toutes les ressources consommées ne sont pas présentes dans les inventaires des joueurs endossant les différents rôles prototypiques de l'activité. Notre idée a donc été de fournir une unique ressource permettant de manipuler l'embranchement souhaité, cette ressource n'étant libérée qu'une fois tous l'embranchement fini.

Le problème posé par *ClassCraft* était lui bien plus complexe car il mettait en péril l'algorithme en chaînage arrière pour tous les scénarios nécessitant de comparer deux quantités de ressources pouvant fluctuer au sein du scénario (typiquement des points de vie par exemple). La solution que nous avons apportée a été de combler les lacunes de l'algorithme à l'aide d'une procédure pouvant subdiviser une activité en fonction de la ressource sujette à comparaison.

Pour ce qui est de *ByteBattle*, il est devenu rapidement évident que représenter l'ensemble des matchs individuels par des activités ne pouvait pas être une solution envisageable, bien que la solution apportée par *ClassCraft* ait pu être une piste intéressante pour y pallier. De fait, nous nous sommes reposés sur des activités plus abstraites (mais conservant les propriétés d'interactions soulevées), ainsi que sur des ressources permettant d'identifier les différents groupes des joueurs (similairement aux ressources identifiant la disponibilité des branches pour *Learning Adventure*).

*Voracy Fish* quant à lui n'a pas réellement posé de problème. Nos quatre scénarios ayant pu être modélisés et ces derniers pouvant de plus être exploités par notre algorithme. Toutefois, les différentes difficultés que nous avons rencontrées indiquent clairement que l'ontologie actuelle est difficile à utiliser pour un utilisateur ne la maîtrisant pas complètement.

Nous avons développé une autre version de l'ontologie qui a fait l'objet d'un article court à ECTEL 2017 (Guinebert, Yessad, Muratet, & Luengo, 2017). Cette version de l'ontologie ajoute la possibilité à l'utilisateur d'utiliser une grammaire formelle (Chomsky, 1957) pour exprimer des consommation et production de ressources complexes. Avec ces expressions, il devient en effet possible d'exprimer la quantité des différentes ressources produites en fonction des ressources consommées. Prenons par exemple un jeu où le nombre de points détermine le vainqueur et dans lequel celui-ci remporte 2 fois sa mise de départ. Dans le cadre de cette activité nous aurions deux rôles : « Joueur 1 » et « Joueur 2 » modélisés ainsi avec notre grammaire formelle :

- Joueur 1 : consomme : n*Argent ; m*Points. Produit : Si (m>k) {n*2*Argent} ; Si (m==k) {n*Argent}.
- Joueur 2 : consomme : n*Argent ; k*Points. Produit : Si (k>m) {n*2*Argent} ; Si (m==k) {n*Argent}.

Cette version pallie un certain nombre de problèmes d'abstraction nés de la nature fixe des productions et consommations de l'ontologie présentée ici. Un exemple de ce qu'il est possible d'accomplir avec la grammaire formelle est présent dans le lien en bas de page[7]. Au bout de ce lien se trouve la description avec la version étendue de l'ontologie des quatre jeux modélisés ici.

La grammaire formelle apporte cependant des problèmes liés à la nature fluctuante de ses productions et consommations. En effet, son utilisation en l'état par l'algorithme n'est pas possible, celui-ci demandant de connaitre à l'avance les ressources consommées et produites par les rôles prototypiques. Une perspective intéressante de ce travail serait donc de développer davantage l'algorithme pour qu'il soit en mesure de traiter les jeux sérieux décrits à l'aide de la grammaire formelle.

Dans le format de notre ontologie utilisé dans ce chapitre, la modélisation effectuée pour un scénario de JSMJ, n'est pas nécessairement utilisable en l'état pour un autre scénario du même JSMJ. Un exemple de cela, consiste à faire varier le nombre de joueurs impliqués. Toutes les activités nécessitant l'ensemble des joueurs (initialisations, classements, activités finales, …) doivent subir des modifications pour suivre ces changements de cardinalité. Bien entendu, les activités les plus abstraites demeurent utilisables telles quelles et certaines activités ne nécessiteront que peu de modifications, toutefois cette nécessité de réadapter un certain nombre d'activités clés du scénario est une limite de notre ontologie dont il faut avoir conscience lors de la modélisation d'un scénario de JSMJ.

Lorsque nous avons conçu notre modèle, nous voulions que celui-ci offre le plus de liberté possible à l'utilisateur afin que celui-ci puisse représenter toutes les activités dont il ait besoin pour formaliser son scénario de JSMJ. Cependant, cette liberté apporte avec elle son lot de problème. En effet, l'utilisateur étant libre de ses choix de modélisation et d'abstraction, il peut advenir que les activités modélisées par ce dernier soient trop ou pas assez proche des actions individuelles des joueurs ce qui rend la détection des interactions problématique.

[7] https://nuage.lip6.fr/s/3mj8BMR85cNoa7S

Ces limites et problématiques soulevées par notre ontologie, ainsi que la proposition d'une ontologie s'appuyant sur une grammaire formelle offre de nombreuses perspectives quant au développement de ce travail. Nos conclusions et perspectives sont davantage détaillées dans le chapitre VIII.

# VII. Expérimentation

En Introduction (chapitre I) nous avions établi trois questions de recherche :

1. Quel modèle de scénario adopter pour permettre la description puis la détection d'interactions entre pairs ? (QR1)
2. Comment pouvons-nous modéliser formellement les interactions entre pairs ? (QR2)
3. Est-il possible d'analyser de manière fiable un scénario de jeu à priori dans le but de prévoir les interactions multi-joueurs effectuées par les apprenants ? (QR3)

Afin de vérifier que nos contributions répondent à nos trois questions de recherche, nous avons mené à bien une expérimentation à partir de laquelle nous avons effectué deux analyses. La section VII.4, présente la première analyse visant à vérifier la validité des contributions du chapitre III et IV qui répondent à la QR1 et à la QR2. Le principe de cette analyse est de comparer les propriétés d'interactions extraites automatiquement des traces des joueurs en utilisant le modèle formel proposé, aux interactions que ceux-ci ont déclaré avoir ressenti. Notre ambition à travers cette analyse est de démontrer la pertinence de la formalisation des interactions proposée.

La question de recherche QR3 a quant à elle fait le sujet d'une analyse présentée dans la section VII.5. L'idée de celle-ci est de comparer les propriétés et séquences extraites du scénario à l'aide des algorithmes présentés dans le chapitre V (créé pour répondre à la QR3) aux interactions détectées dans les traces des joueurs dans la section VII.4. En montrant que les interactions prévues par nos algorithmes sont similaires (en types et en nombre) aux interactions effectuées par les apprenants, nous sommes en mesure de répondre à la QR3. Pour ce qui est de la fiabilité évoquée dans la QR3, notre réponse se base à la fois sur les résultats de la section VII.4 (lié à la pertinence de nos propriétés vis-à-vis des interactions ressenties) et sur les séquences extraites par les algorithmes dans la section VII.5 (avons-nous été en mesure de déterminer les séquences d'activités que pourraient réaliser les apprenants ?).

Dans le cadre de l'expérimentation sur laquelle se repose ces deux analyses, nous avons créé notre propre jeu sérieux sur la logique propositionnelle. Celui-ci est présenté en détail dans la section VII.1 et se décompose en deux scénarios différents. La modélisation de celui-ci à l'aide de l'ontologie proposée est quant à elle présentée en section VII.2.

## VII.1. MultiPlayer LOgic Game (MP-LOG)

Dans le cadre de cette thèse, il était nécessaire d'avoir à notre disposition un ou plusieurs jeux sérieux multi-joueurs dédiés à l'apprentissage remplissant un certain nombre de critères. Les critères dont nous avions besoin étaient les suivants :

- Plusieurs interactions disponibles au sein d'un même scénario
- La liberté des joueurs quant aux interactions et actions à effectuer
- La présence d'au moins des interactions suivantes : coopération, collaboration et compétition

- La possibilité de définir ou de modifier les scénarios en fonction de nos besoins
- La possibilité d'avoir accès aux traces des joueurs une fois les sessions de jeu effectuées

N'ayant trouvé aucun jeu ou scénario répondant à l'ensemble de ces critères, nous avons entrepris de développer notre propre jeu pour mener à bien des expérimentations et ainsi fournir une preuve de concept de nos contributions.

Pour ce faire, nous avons créé un jeu de logique appelé *MultiPlayer LOgic Game* (MP-LOG). C'est un jeu sur la logique propositionnelle pour des étudiants en deuxième année de licence universitaire. La construction de ce jeu s'appuie sur le moteur de jeu *Spring*[8] (développé par les *Swedish Yankspankers*[9]) et l'outil auteur dédié *SPRED*[10] (développé par Mathieu Muratet et Benjamin Bontemps). Au-delà du fait que ce jeu répond à l'ensemble de nos critères, sa visée pédagogique était de fournir aux apprenants un environnement ludique dans lequel ils peuvent réutiliser et investir des connaissances vues en cours. Bien entendu, les scénarios de ce jeu sont multi-joueurs et se jouent soit en équipe soit contre d'autres joueurs.

Nous avons développé deux scénarios. Ces deux scénarios s'inspirent du *gameplay* des jeux de type *Tower Defense* dans lequel le joueur doit protéger ses vies face à des vagues d'ennemis. Nous désignons les deux scénarios par TD2 et TD4 en référence à *Tower Defense* (TD2 pour le scénario à deux joueurs et TD4 pour celui à deux équipes de 2 joueurs chacune).

## VII.1.a. Présentation du scénario TD2

Dans le TD2, les deux joueurs jouent contre l'ordinateur qui envoie des vagues d'ennemis de plus en plus puissantes à intervalle de temps régulier. Afin de pouvoir se défendre face à cette menace, les deux joueurs cherchent à obtenir des unités pour pouvoir se défendre. Pour obtenir ces défenseurs, les apprenants doivent alors répondre à des quiz de logique, pour lesquels ils doivent fournir l'ensemble des bonnes réponses. Chaque joueur possède deux stratégies pour soumettre une solution : soit il soumet seul, soit il soumet avec l'autre joueur. Chaque stratégie possède ses avantages et inconvénients. Soumettre seul rapporte 2 points et un défenseur au joueur ayant donné la bonne réponse, l'autre joueur ne gagne rien. Soumettre à deux rapporte 1 point et 1 défenseur à chacun des joueurs. Répondre seul est donc intéressant pour les joueurs ayant une nature compétitive qui désireraient battre les plus hauts scores en termes de points. En revanche, si un joueur cherche à tenir le plus longtemps possible face aux assauts de l'ordinateur, il se doit de répondre collaborativement pour maximiser ses défenses.

[8] http://springrts.fr/
[9] http://www.clan-sy.com/
[10] http://progandplay.lip6.fr/

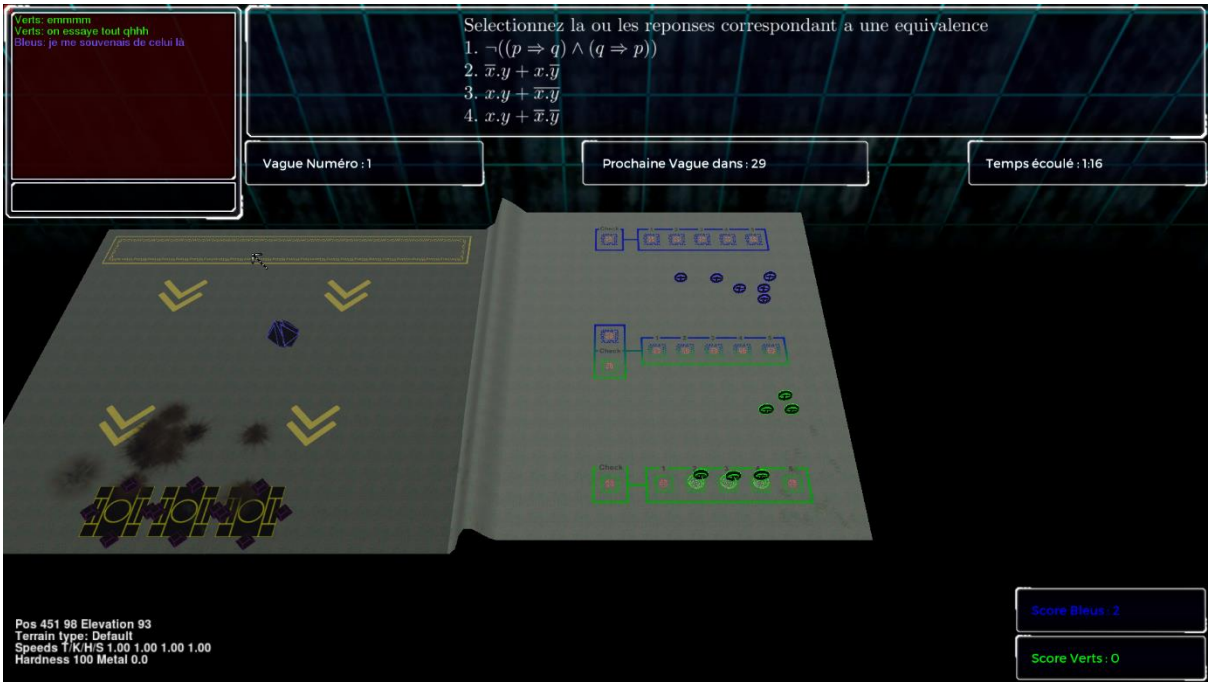


*Figure 19 : Capture d'écran du TD2*

Comme nous pouvons le voir sur la Figure 19, le TD2 se sépare en plusieurs grandes parties. La partie haute de l'écran se divise en trois espaces :

- En haut à gauche, se trouve le chat à travers lequel les deux joueurs peuvent communiquer
- En haut à droite, se trouve le quiz en logique propositionnelle
- Juste en dessous de la zone quiz, le joueur trouve différentes informations sur la vague des ennemis (numéro de la vague, temps avant la prochaine vague, temps écoulé depuis le début du jeu)

Deux grandes zones sont à distinguer dans la partie centrale de l'écran. La zone de gauche correspond au couloir dans lequel les vagues ennemies évoluent. Ces dernières apparaissent en haut de la zone dans l'encadré jaune et, suivant les flèches, se dirigent vers les trois vies des joueurs représentées par les trois grandes étoiles noires cerclées de jaune situées en bas à gauche.

La zone de droite (plus claire) regroupe l'ensemble des éléments nécessaires à la création des solutions et leur validation par les joueurs. Nous y distinguons trois parties :

- En haut, la zone de validation du joueur bleu, accessible uniquement à celui-ci.
- Au milieu, la zone de validation commune aux deux joueurs, c'est cette zone qui leur permet de soumettre une réponse co-construite.
- En bas, la zone de validation du joueur vert, uniquement accessible à celui-ci.

Zoomons maintenant sur les différents éléments du jeu pour mieux les comprendre.


*Figure 20 : Capture d'écran de la zone de réponse du Joueur 1 dans le TD2*

La Figure 20 présente une capture d'écran partielle du jeu. Nous y voyons, dans la partie haute la zone présentant les quiz aux joueurs ainsi que les différentes réponses possibles. Juste en dessous de la question se trouvent deux informations sur le temps dont nous avons parlé précédemment.

En dessous de la zone quiz, se trouve une zone bleue divisée en deux parties et découpée en 6 cases rouges. La partie de gauche, ne comportant qu'une case, sert à valider individuellement la réponse du Joueur Bleu. La partie de droite, permet au Joueur Bleu de sélectionner sa réponse. Pour ce faire, les joueurs ont chacun à leur disposition 6 jetons (un pour chaque case). Le jeton sélectionné par le Joueur Bleu est encadré de vert.

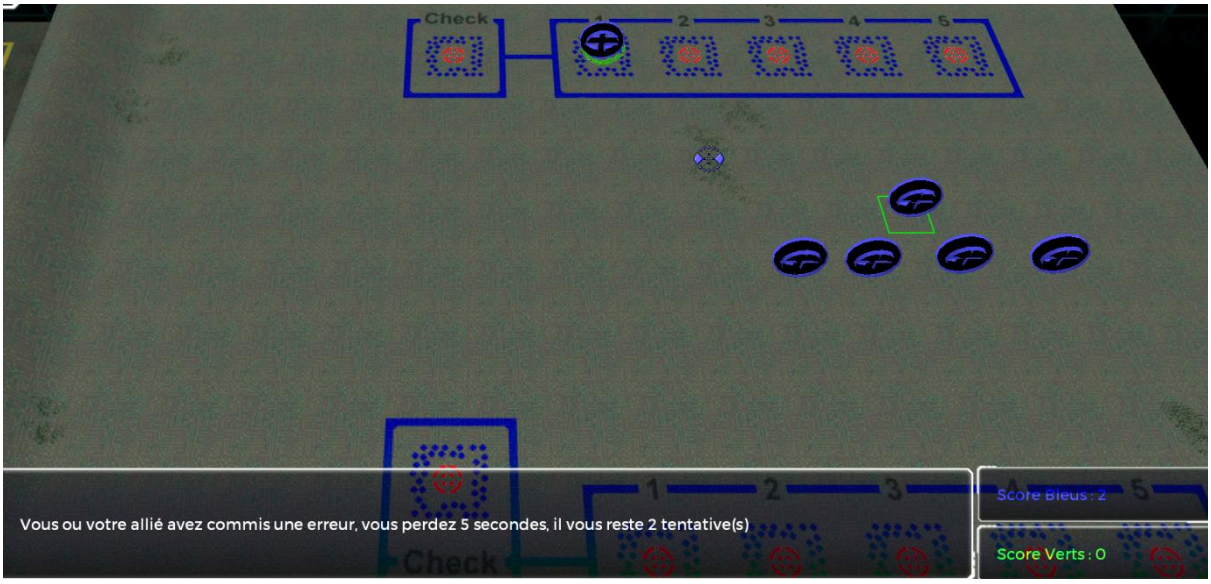


*Figure 21 : Capture d'écran indiquant au joueur qu'il a commis une erreur*

La Figure 21 présente le message que reçoivent les joueurs lorsqu'ils se sont trompés. Le message contient également deux informations. La première rappelle aux joueurs que la prochaine vague d'ennemis arrivera plus rapidement du fait de leur erreur. La deuxième indique aux joueurs le nombre de tentatives leur restant pour donner la réponse correcte. En effet, les joueurs ont à leurs disposition trois tentatives pour répondre correctement à une question du QCM. Une fois ces trois tentatives passées, le QCM passe à la question suivante, sans gain pour les joueurs. Deux autres informations sont disponibles sur cette capture d'écran : en bas à droite de la Figure 21 les joueurs ont accès à leur score ainsi qu'à celui de leur partenaire.

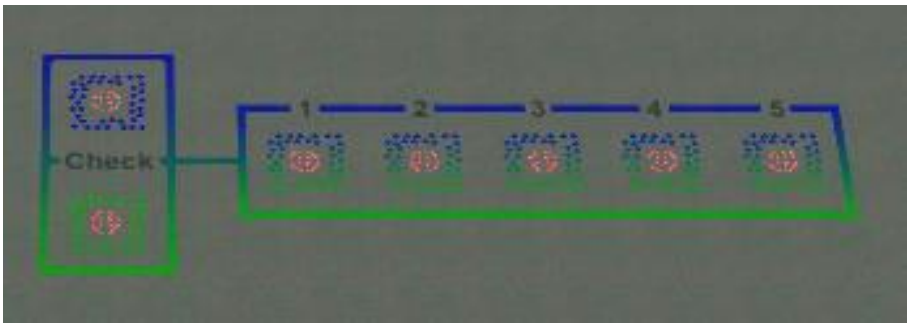


*Figure 22 : Capture d'écran de la zone de validation centrale*

Similairement à la zone de validation du Joueur Bleu, la Zone de validation centrale se décompose en deux parties. À gauche, les joueurs peuvent valider à deux, il est nécessaire que chacun d'eux place un jeton dans sa case de validation pour que la solution soit soumise. À droite, les joueurs peuvent tous deux placer des jetons pour créer leur solution commune.

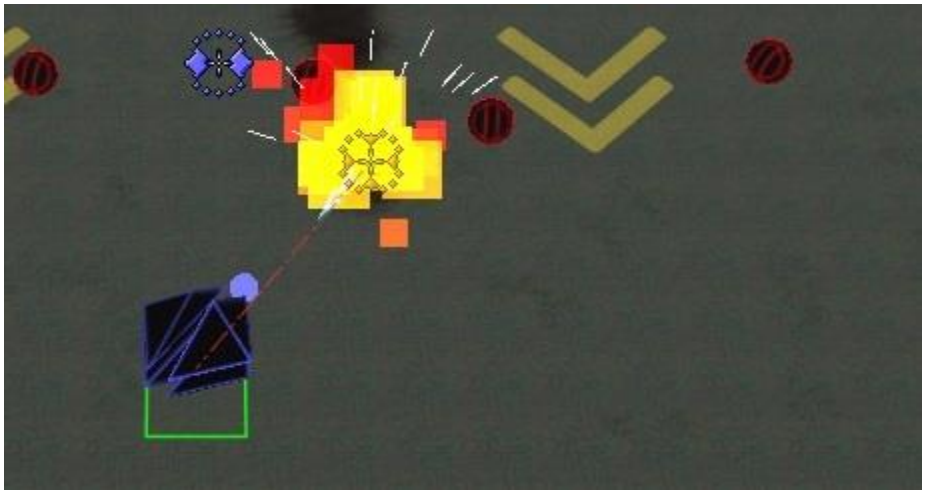

*Figure 23 : Capture d'écran d'une attaque ennemie*

Pour finir notre présentation du TD2, la Figure 23, présente un défenseur (en bleu) actuellement en train de tirer sur la vague des ennemis (en rouge), qui attaquent les vies des joueurs.

## VII.1.b. Présentation du scénario TD4

Dans ce scénario, les joueurs sont divisés en deux équipes de deux joueurs chacune, l'équipe Nord et l'équipe Sud. Chaque équipe possède trois vies situées à la fin d'un long couloir. Le but du jeu pour chacune des deux équipes est de détruire les vies de l'autre avant d'avoir perdues les siennes. Deux actions sont disponibles pour les joueurs : créer un défenseur afin de protéger leurs vies contre une attaque adverse, et créer un attaquant pour détruire les défenseurs et vies de l'équipe ennemie. Chacune de ces deux actions coûtent un jeton au joueur qui l'entreprend. Pour obtenir ces jetons, les apprenants doivent donner une interprétation correcte pour une formule logique qui leur est soumise. Similairement au TD2, les étudiants ont à leur disposition des unités pour construire la solution, ainsi que deux possibilités de validation : une en équipe et une en solo.

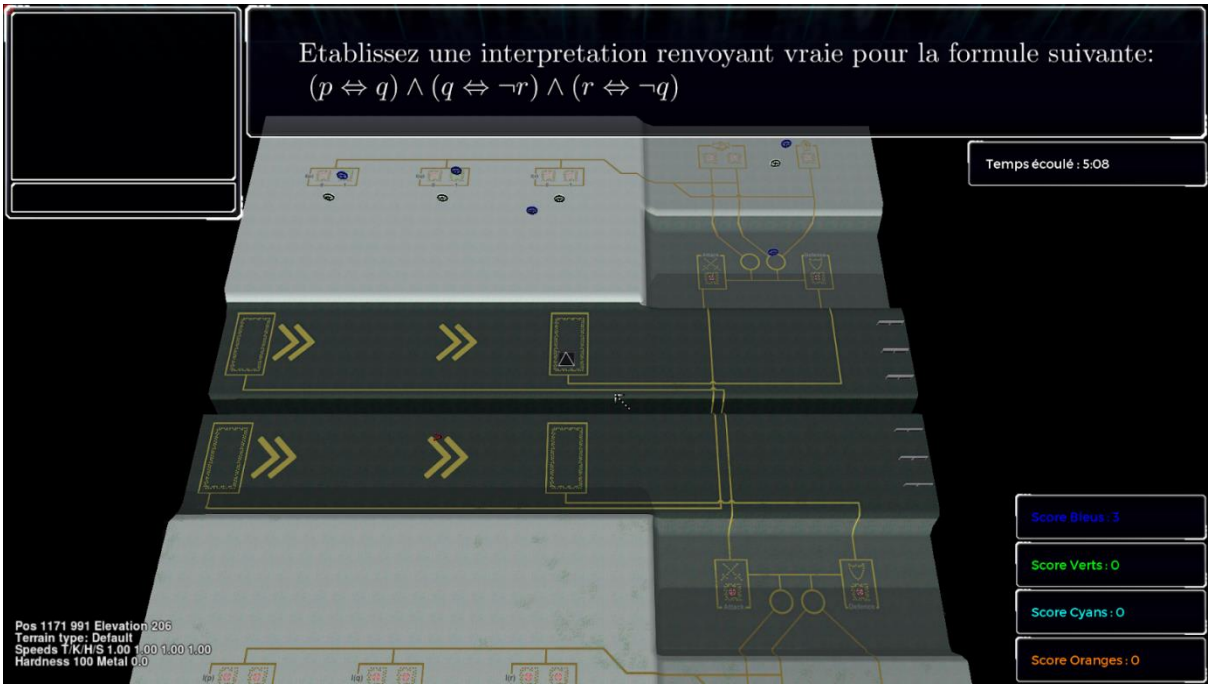


*Figure 24 : Vue d'ensemble du jeu à quatre joueurs*

La Figure 24 permet de visualiser l'environnement de jeu dans le cadre du TD4. La partie haute de la fenêtre suit une division similaire à celle du TD2, c'est à dire :

- En haut à gauche, le chat pour les joueurs d'une même équipe
- En haut à droite, l'énigme posée aux joueurs

L'environnement de jeu au centre se découpe en deux grandes zones symétriques. La zone du haut appartient à l'équipe Nord constituée du Joueur Bleu et du Joueur Vert. La zone du bas appartient à l'équipe Sud constituée du Joueur Orange et du Joueur Cyan. Chacune de ces deux zones se divisent en quatre parties. En prenant pour référence la zone Nord, nous avons :

- En haut à gauche (en blanc), la zone de construction des solutions
- En haut à droite (en gris clair), la zone de validation
- Juste en dessous de la zone de validation (en gris foncé), la zone permettant de lancer attaque et défense
- Au centre (en noir avec les flèches jaunes), le couloir d'attaque au bout duquel se trouve les vies de l'équipe Nord

Zoomons sur ces différentes parties pour en expliquer le fonctionnement.

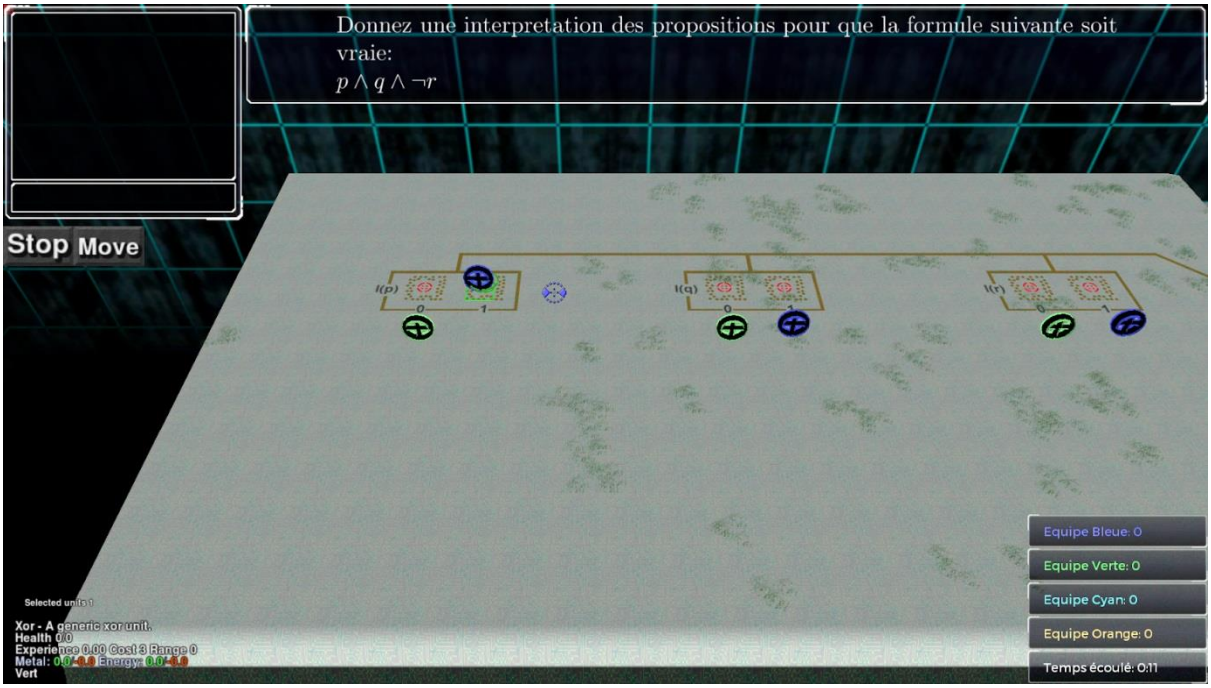


*Figure 25 : Zone pour la construction d'une solution*

Dans le cadre du TD4, les apprenants doivent donner une interprétation de p, q et r de manière à ce que la formule proposée soit vraie. Une interprétation pour chacune de ces trois variables doit être fournies pour être en mesure de valider une solution. Une équipe (composée de deux joueurs) a à sa disposition 6 cases. Les deux cases de gauches permettent de donner l'interprétation de p, les deux du centre celle de q, et enfin celle de droite sont là pour la variable r. La case de gauche de chacun de ces ensembles permet de donner une interprétation à 0 pour la lettre concernée. La case de droite donne une interprétation à 1.

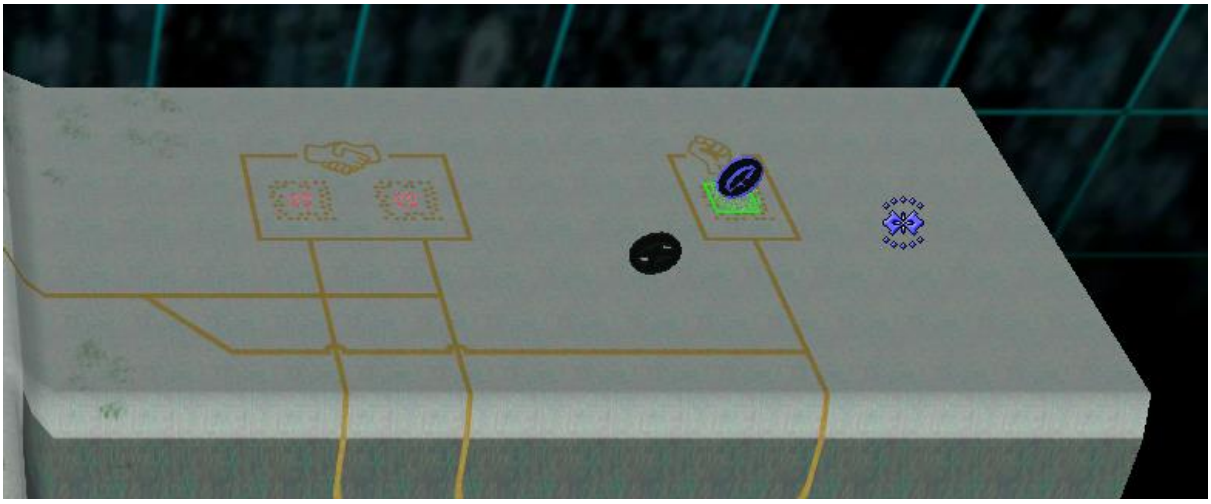

*Figure 26 : Capture d'écran de la zone de validation du TD4*

Comme évoqué précédemment, les joueurs possèdent deux possibilités pour valider leur réponse : seul ou ensemble. La validation seule n'est ici intéressante qu'en cas de désaccord profond avec son coéquipier puisqu'elle ne permet d'obtenir qu'un jeton en cas de réponse correcte, là où la validation en équipe permet d'en obtenir deux. Dans la Figure 26, la zone de

gauche avec les deux cases rouges, permet de valider ensemble, la zone de droite avec une seule case où nous pouvons voir une unité bleue, permet de valider seul. L'erreur dans le TD4 est en revanche beaucoup plus punitive que l'erreur dans le TD2 afin d'éviter que les étudiants ne cherchent l'interprétation correcte par pur hasard. Toute mauvaise réponse fait directement passer à la formule suivante et fournit un jeton bonus à l'équipe adverse.


*Figure 27 : Zone pour dépenser ses Jetons*

La Figure 27 se décompose en trois zones. La case dans la zone de gauche avec les épées croisées permet d'envoyer un attaquant dans le couloir d'attaque de l'équipe adverse. La case dans la zone de droite avec le bouclier permet de générer un défenseur dans son propre couloir d'attaque. Les deux cercles au centre sont l'endroit où les jetons obtenus en récompense d'une bonne formule ou d'une erreur de l'adversaire apparaissent.

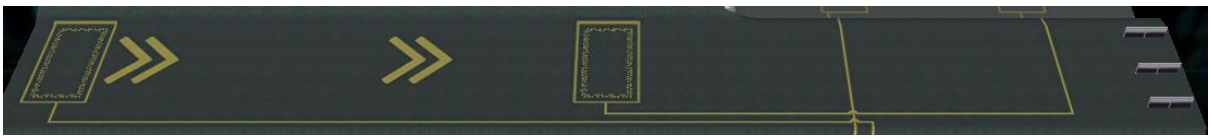

*Figure 28 : Couloir d'attaque dans le TD4*

Les attaquants générés à l'aide des jetons apparaissent dans l'encadré jaune à gauche de la Figure 28. Ils suivent ensuite les flèches au sol avant d'être confrontés aux défenseurs apparaissant dans l'encadré jaune au centre. S'ils parviennent à passer les défenseur (un défenseur permet de détruire un attaquant), ils peuvent atteindre la fin du couloir où se trouvent les vies adverses (les trois lignes blanches). Lorsqu'un attaquant y parvient, une vie est détruite.

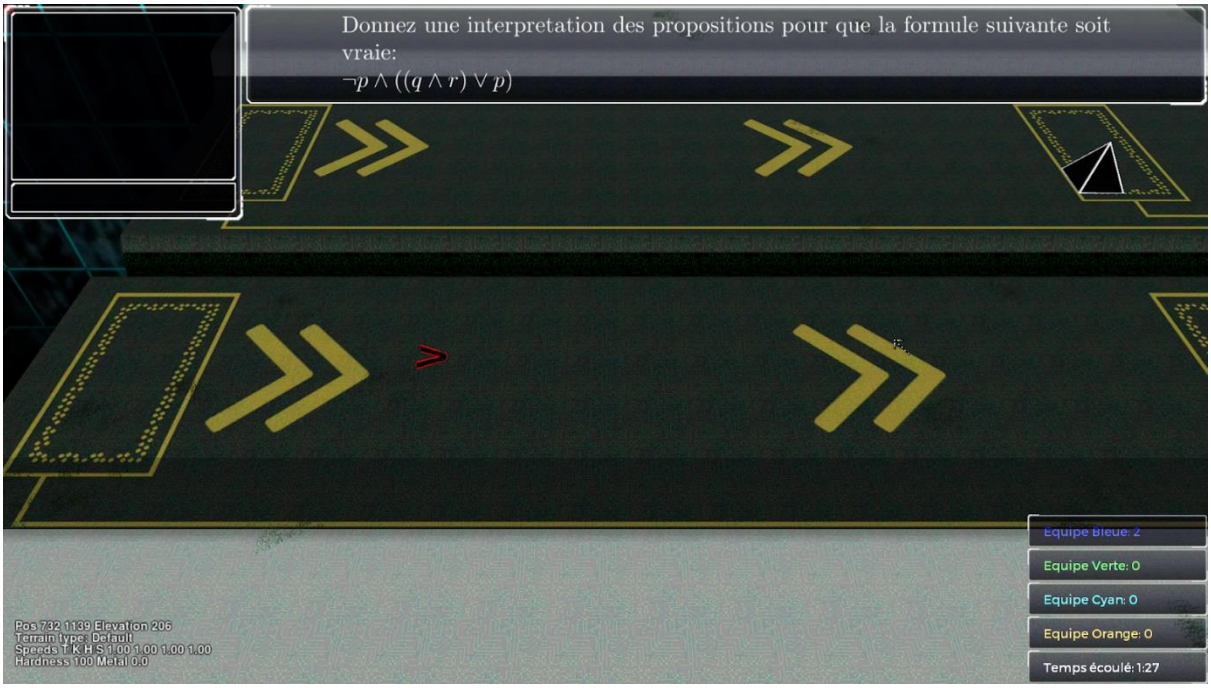


*Figure 29 : Attaquant et Défenseur dans le TD4*

L'unité rouge apparaissant dans la Figure 29 est un attaquant se trouvant actuellement dans le couloir d'attaque de l'équipe Sud. La pyramide blanche en haut à droite de la Figure 29 symbolise quant à elle un défenseur de l'équipe Nord actuellement en train de protéger son couloir.

## VII.2. Modélisation de MP-LOG avec le formalisme MPLGO

Dans le chapitre VI, nous avons mis à l'épreuve MPLGO en modélisant quatre scénarios de jeux sérieux multi-joueurs : *Voracy Fish, LearningAdventure, ByteBattle* et *ClassCraft*. Nous n'avons cependant pas la possibilité de récupérer les traces pour ces jeux, ce qui rend donc difficile l'évaluation de nos différents algorithmes. En effet, nous pourrions utiliser notre algorithme d'analyse pour déterminer les séquences et les propriétés d'interactions émergeant de ces scénarios, mais sans traces, nous ne serions pas en mesure de répondre aux questions de recherche QR2 et QR3. De fait, les traces nous permettent d'évaluer nos contributions en comparant les résultats des différents algorithmes appliqués d'une part sur les descriptions de scénarios et d'autre part sur les traces produites par des joueurs qui jouent ces scénarios.

Tout d'abord ces traces nous permettent d'évaluer la fiabilité des résultats obtenus quant à la détection des propriétés d'interactions en les comparant aux interactions déclarées par les joueurs.

Ensuite, les traces nous permettent également (et surtout) d'évaluer l'algorithme extrayant les interactions à partir de la description du scénario. Les propriétés ainsi extraites en amont de tout utilisation du jeu sont comparées aux propriétés détectées dans les traces.

Ainsi, comme stipulé au cours des différents chapitres, il nous est nécessaire de modéliser avec notre formalisme MPLGO les deux scénarios TD2 et TD4 pour pouvoir évaluer nos algorithmes par la suite. Les sections VII.2.a et VII.2.b présente le détail de ces modélisations.

## VII.2.a. Modélisation du TD2

Le TD2 possède un fonctionnement particulier : le jeu s'arrête lorsque les trois vies des joueurs ont été détruites par l'ordinateur. Or, il est à noter que les vagues d'ennemis sont envoyées sans se soucier du fait que les joueurs aient, ou non, réussi une formule (et donc obtenu des points et des défenseurs). De fait, il est donc possible de finir le scénario sans entreprendre une seule action. Cette stratégie aboutira forcément à une défaite mais constitue tout de même un chemin possible menant à la fin du scénario.

La modélisation d'un scénario doit impérativement commencer par la détermination de ses scénarios élémentaires. Dans le cas du TD2, nous pouvons discerner deux scénarios (voir Figure 30) :

- Les joueurs n'ont pas agi : l'activité « FIN » dépourvue de lien de précédence
- Les joueurs ont agi avant de finir : « FIN_Formule »

Vient ensuite les activités fondamentales du jeu, celles que les joueurs peuvent entreprendre (s'ils y possèdent des rôles actifs). Dans le cas du TD2 nous en discernons trois types :

- Bonne_réponse_Commun : les joueurs ont validé une formule à deux. Cette activité est en réalité divisée en deux autres : Bonne_réponse_VC et Bonne_réponse_CCC caractérisant les propriétés associées.
- Bonne_réponse_Seul : un seul joueur a validé la formule. Celle-ci est séparé en Bonne_réponse_Ind et Bonne_réponse_CI.
- Mauvaise_réponse : les joueurs se sont trompés sur la formule (passage à la suivante). Nous la séparons en 4 : Mauvaise_réponse_Ind, Mauvaise_réponse_VC, Mauvaise_réponse_CCC, Mauvaise_réponse_CI

La différenciation entre ces trois types d'activités a été effectuée par rapport à leurs règles de production et de consommation respectives. Dans le cas de Bonne_réponse_Commun, les deux joueurs produisent des points et des défenseurs, alors que seul le joueur ayant validé la formule en produit dans Bonne_réponse_Ind. Pour ce qui est de Mauvaise_réponse, les joueurs perdent une formule, mais ne sont malheureusement pas en mesure de gagner des Défenseurs ou des Points.

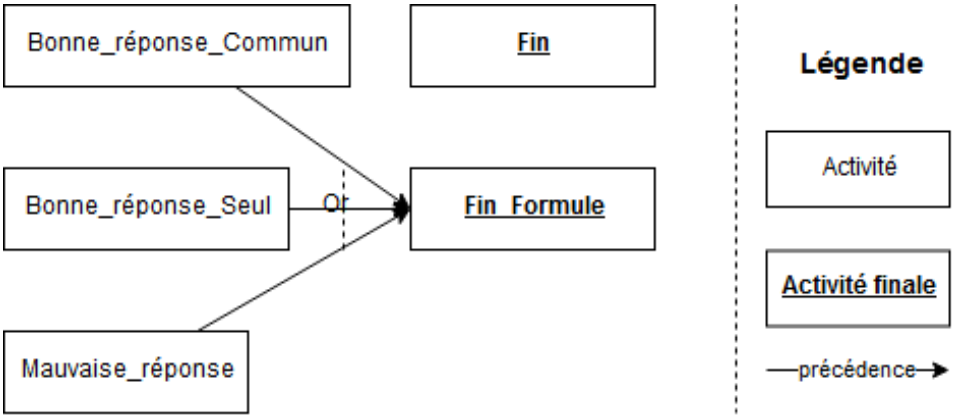


*Figure 30 : Représentation simplifiée du graphe de précédence du TD2*

Les activités de validation du TD2 sont au nombre de 8 divisée en deux groupes de quatre correspondant à une réponse juste à la formule, ou à une réponse fausse. Chacune des quatre activités de chaque groupe correspond à une propriété d'interaction donnée.

*Tableau 22 : Activité Bonne_réponse VC*

| Bonne_réponse_VC | | |
|---|---|---|
| Rôle prototypique | Constructeur | Bénéficiaire |
| Description | Joueur construisant la Formule | Joueur ne faisant que valider |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Défenseur » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé | « Défenseur » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Défenseur » (objectif réussi) | Produire « Défenseur » (objectif réussi) |

*Tableau 23 : Activité Mauvaise_réponse CCC*

| Mauvaise_réponse CCC | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 2 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A |
| Produit | « Formule consommée » dans l'inventaire partagé |
| Requiert | Ø |
| Connaissances Ciblées | « Logique propositionnelle» |
| Objectifs | Produire « Défenseur » (objectif échoué) |

*Tableau 24 : Activité Mauvaise_réponse Ind*

| Mauvaise_réponse_Ind | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 1 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A |
| Produit | « Formule consommée » dans l'inventaire partagé |
| Requiert | Ø |
| Connaissances Ciblées | « Logique propositionnelle» |
| Objectifs | Produire « Défenseur » (objectif échoué) |

L'activité Bonne_réponse VC correspond à la propriété Validation Commune. Les deux joueurs ont donné une réponse correcte et obtenu un Défenseur chacun. En revanche, seul l'un d'eux a participé à la construction de la solution, il est donc le seul à cibler la connaissance « Logique » et il n'y a donc pas de Construction Commune de Connaissance au contraire de l'activité Mauvaise_réponse CCC (cf. Tableau 23). Dans cette dernière activité, les deux joueurs n'ont pas réussi à créer la bonne formule, ils n'obtiennent donc pas de Défenseur. En revanche, ils ont construit et validé la solution ensemble.

Dans le cadre de Bonne_réponse_CI, un joueur a validé seul sa réponse et obtenu un défenseur. L'autre joueur avait participé et tenté de créer une formule, mais l'action de son coéquipier l'a empêché d'obtenir un Défenseur. Il y a donc un Conflit Interne entre les deux joueurs. A l'inverse, dans l'activité Mauvaise_réponse_Ind (cf. Tableau 24), seul un joueur a été actif (bien qu'il se soit trompé). Sa validation seule n'entraîne donc pas de conflit interne.

*Tableau 25 : Activité Bonne_réponse_CI*

| Bonne_réponse_CI | | |
|---|---|---|
| Rôle prototypique | Constructeur | Lésé |
| Description | Joueur construisant la Formule | Joueur n'ayant pu valider en dépit de sa participation |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Défenseur » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Défenseur » (objectif réussi) | Produire « Défenseur » (objectif échoué) |

*Tableau 26 : Activité Fin*

| Fin | |
|---|---|
| Rôle prototypique | Joueur |
| Description | Activité de Fin |
| Cardinalité | 2 |
| Contraintes | Ø |
| Equipes | Ø |
| Consomme | Ø |
| Produit | « Fin » |
| Requiert | Ø |
| Connaissances Ciblées | Ø |
| Objectifs | Ø |

*Tableau 27 : Activité Fin Formule*

| Fin Formule | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 2 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule consommée » dans l'inventaire partagé de l'équipe A |
| Produit | « Fin » |
| Requiert | Ø |
| Connaissances Ciblées | Ø |
| Objectifs | Ø |

Les quatre autres activités de validation non décrites ici sont disponibles en Annexes (section X.1). À nos 8 activités nous ajoutons les deux activités finales « Fin » et « Fin Formule ». La première peut être accédée sans contrainte, elle n'exige aucune ressource et permet de symboliser un scénario où les vies des joueurs sont détruites sans effectuer de Formules. « Fin Formule » en revanche nécessite l'utilisation d'une activité de validation (présence de « Formule consommée » dans l'inventaire partagé).

Les propriétés détectées au sein de chaque activité du TD2 sont disponibles en annexe section X.3.

## VII.2.b. Modélisation du TD4

La représentation du TD4 est à la fois plus riche et plus complexe que celle du TD2. Elle comporte les trois activités de Validation de formule Bonne_réponse_Commun, Bonne_réponse_Seul et Mauvaise_réponse comme le TD2 (avec des productions différentes cependant). En plus de celles-ci, le TD4 comporte 3 autres activités fondamentales :

- Créer Défense : Un joueur dépense un Jeton pour créer un défenseur.
- Créer Attaque : Un joueur dépense un Jeton pour créer un attaquant.
- Perdre Vie : Un attaquant ôte une vie à l'équipe adverse.

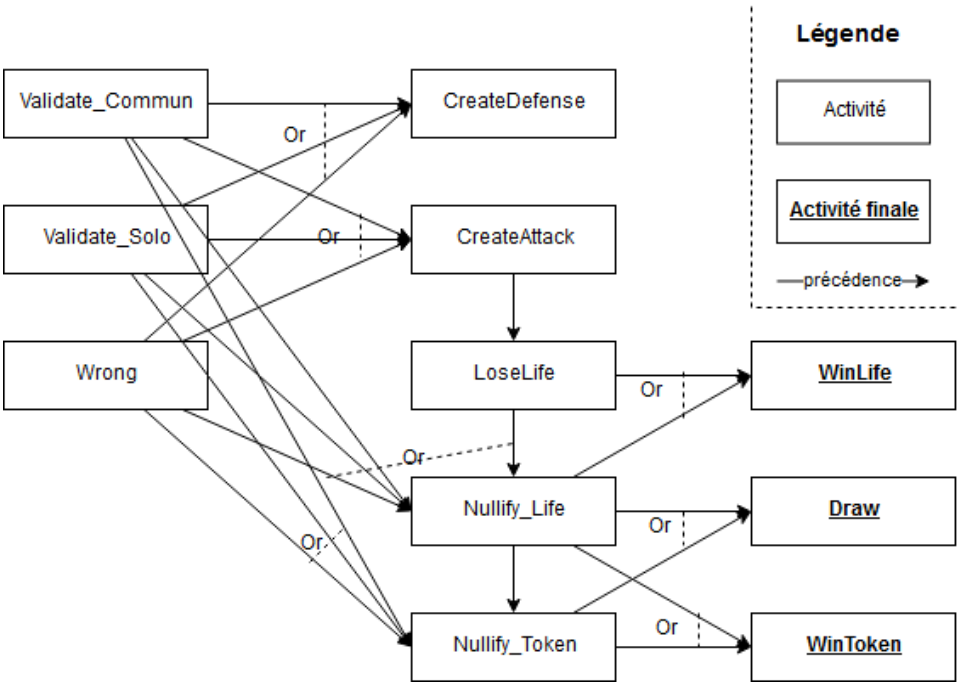


*Figure 31 : Visualisation simplifiée du TD4*

Dans le TD4, trois fins différentes peuvent être atteintes, la victoire par nombre de vies restantes, la victoire par jetons et l'égalité. Il y a deux manières de mettre fin au jeu, soit lorsque l'une des deux équipes a perdu toutes ses vies, soit lorsque l'une des deux équipes a utilisé toutes ses formules. Dans le premier cas, il s'agit bien évidemment d'une victoire par point de vies. Dans le deuxième, il y a trois dénouements possibles. Si une équipe a plus de points de vie, elle gagne. Si les deux équipes ont le même nombre de points de vie, l'équipe

avec le plus de jetons remporte la victoire. Dans le cas où les deux équipes auraient le même nombre de points de vies et de jetons, nous avons une égalité. Nous obtenons donc les trois activités finales suivantes :

- Victoire Vie : Il s'agit de la victoire par point de vie.
- Victoire Jeton : Il s'agit de la victoire par jetons.
- Egalité : Symbolise l'égalité.

*Tableau 28 : Activité Mauvaise_réponse VC*

| Mauvaise_réponse_VC | | | |
|---|---|---|---|
| Rôle prototypique | Constructeur | Bénéficiaire | Adversaire |
| Description | Joueur construisant la Formule | Joueur ne faisant que valider | Joueur adverse obtenant un Jeton à cause de l'échec du Constructeur. |
| Cardinalité | 1 | 1 | 1 |
| Contraintes | Ø | Ø | Ø |
| Equipes | A | A | B |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø | Ø |
| Produit | « Formule consommée » dans l'inventaire partagé | | « Jeton » |
| Requiert | Ø | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø | Ø |
| Objectifs | Produire « Jeton » (objectif échoué) | Produire « Jeton » (objectif échoué) | Ø |

Contrairement au TD2, qui ne possède que des activités de jeu et des activités finales, le TD4 possède également des activités de traitement (cf. section VI.3). Ces activités ne comportent aucun rôle actif et sont chargées de représenter les modifications que le jeu opère en fin de partie pour établir quelle activité finale choisir. Ces activités n'engendrent aucune interaction et sont au nombre de deux :

- Annuler Vie : enlèvera une vie à chaque équipe tant que l'une d'elle n'a pas atteint 0 points de vie. En cas de fin par formules, elle permet de déterminer si des joueurs ont

gagnés grâce aux points de vie ou non (si une équipe a plus de points de vie, il lui en restera au moins 1 quand l'autre sera à 0)

- Annuler Jeton : arriver à cette activité signifie que les deux équipes sont à 0 points de vie grâce à Annuler Vie. Cette activité consomme un jeton dans chaque équipe jusqu'à ce qu'il n'en reste plus dans l'inventaire d'au moins l'une des deux équipes.

Le fonctionnement des activités de validation du TD4 est similaire à celle du TD2 au détail près que celles du TD4 produisent des Jetons en lieu et place des Défenseurs du TD2. Un autre aspect divergeant vient du fait qu'en cas d'échec, l'équipe adverse gagne le Jeton que les joueurs auraient dû obtenir. L'activité Mauvaise_réponse VC détaillé dans le Tableau 28 est un exemple d'une activité de validation erronée dans le TD4. Les 7 autres activités de validation étant similaires aux TD2, elles sont présentées en annexe (section X.2).

Comme stipulé plus haut, les joueurs peuvent créer des défenseurs ou des attaquants en dépensant leurs Jetons. Pour ce faire les deux activités Créer Attaque et Créer Défense (cf. Tableau 29 et Tableau 30) sont à leur disposition. Ces activités sont compétitives par nature puisque la création de ces unités est contraire aux intérêts des joueurs opposés. Les adversaires endossent donc un rôle spécifique pour montrer le conflit externe créé (Attaquant dans Créer Défense et Défenseur dans Créer Attaque).

*Tableau 29 : Activité Créer Attaque*

| Créer Attaque | | |
|---|---|---|
| Rôle prototypique | Attaque | Défenseur |
| Description | Joueur dépensant un Jeton pour créer un Attaquant | Joueurs adverses |
| Cardinalité | 1 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Jeton » | Ø |
| Produit | « Attaquant » | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Attaquant »<br>(objectif réussi) | Attaque ne doit pas produire de « Attaquant »<br>(objectif échoué) |

*Tableau 30 : Activité Créer Défense*

| Créer Défense | | |
|---|---|---|
| Rôle prototypique | Défense | Attaquant |
| Description | Joueur dépensant un Jeton pour créer un Défenseur | Joueurs adverses |
| Cardinalité | 1 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Jeton » | Ø |
| Produit | « Défenseur » | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Défenseur » (objectif réussi) | Défense ne doit pas produire de « Défenseur » (objectif échoué) |

Lorsqu'une attaque est réussie l'activité Perdre Vie est appelée. Elle consomme un Attaquant pour enlever une vie dans l'inventaire partagé de l'équipe adverse. Elle est donc nécessairement une activité se reposant sur le conflit externe.

*Tableau 31 : Activité Perdre Vie*

| Perdre Vie | | |
|---|---|---|
| Rôle prototypique | Attaque | Défenseur |
| Description | Joueur utilisant son attaquant pour détruire les points de vies | Joueurs adverses |
| Cardinalité | 1 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Attaquant » | « Vie » dans l'inventaire partagé de l'équipe B |
| Produit | Ø | « Mort » dans l'inventaire partagé de l'équipe B |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Défenseur doit consommer « Vie » (objectif réussi) | Ne pas consommer « Vie » (objectif échoué) |

Un certain nombre de fins sont possibles dans le cadre du TD4. Il est possible d'atteindre la fin du jeu en détruisant tous les points de vie adverses, ou lorsqu'une équipe n'a plus de formule à résoudre. Dans le deuxième cas, nous discernons deux possibilités, soit une équipe à plus de points de vie que l'autre, soit elles en ont autant. Si elles en ont autant, le nombre de Jetons des différentes équipes permet de déterminer le vainqueur. En cas de nombre de jetons identiques, les deux équipes sont à égalité. Pour traiter ces différents cas de figure, notre modélisation propose 5 activités dont 2 activités de traitement (cf. section VI.3 sur *ClassCraft*).

*Tableau 32 : Activité Victoire Vie*

| Victoire Vie | | |
|---|---|---|
| Rôle prototypique | Vainqueur | Perdant |
| Description | Équipe encore en vie | Équipe n'ayant plus de vies |
| Cardinalité | 2 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Vie » dans l'inventaire partagé de l'équipe A | « Mort » * 3 dans l'inventaire partagé de l'équipe B |
| Produit | « Victoire » | « Défaite » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Victoire » (objectif réussi) | Produire « Victoire » (objectif échoué) |

L'activité Victoire Vie consomme une Vie dans une des deux équipes et trois Morts dans l'autre, garantissant ainsi que la première équipe n'est pas à égalité en nombre de vies avec la deuxième. Dans le cas où la fin du jeu a été amenée par le nombre de Formule résolue, l'utilisation de l'activité Annuler Vie (cf. Tableau 33) réduit de 1 le nombre de points de vie de deux équipes jusqu'à ce que l'une des deux soit à zéro. Si les deux équipes ont le même nombre de points de vie lors de l'utilisation d'Annuler Vie, elles finiront toutes les deux à zéros et ne pourront effectuer Victoire Vie. Dans le cas contraire, l'une des deux équipes aura toujours une vie en réserve lorsque l'autre aura trois « Mort ».

*Tableau 33 : Activité Annuler Vie*

| Annuler Vie | | |
|---|---|---|
| Rôle Prototypique | Equipe 1 | Equipe 2 |
| Description | Traitement : nous enlevons une Vie à chaque équipe | Traitement : nous enlevons une Vie à chaque équipe |
| Cardinalité | 2 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Vie » et « Formule consommée » * 8 dans l'inventaire partagé de l'équipe A | « Vie » dans l'inventaire partagé de l'équipe B |
| Produit | « Mort », » Formule consommée » * 8 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | « Mort » et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Ø | Ø |

Annuler Jeton ne peut pas être exécutée tant que les deux équipes ont encore des vies en réserve. Deux activités permettent de produire la ressource Mort nécessaire à Annuler Jeton : Perdre Vie et Annuler Vie. Cependant, si une équipe venait à perdre toutes ses vies avec Perdre Vie, elle aurait dû perdre avec l'activité Victoire Vie. Ainsi, Annuler Jeton utilise une ressource « Signal d'Annulation » produite uniquement par Annuler Vie et imposant ainsi une égalité en nombre de vies chez les deux équipes. Le fonctionnement d'Annuler Jeton est identique à celui d'Annuler Vie : nous retirons un Jeton à chacune des deux équipes.

*Tableau 34 : Activité Annuler Jeton*

| Annuler Jeton | | |
|---|---|---|
| Rôle prototypique | Equipe 1 | Equipe 2 |
| Description | Traitement : nous enlevons une Vie à chaque équipe | Traitement : nous enlevons une Vie à chaque équipe |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Jeton » dans l'inventaire personnel, « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | « Token » dans l'inventaire personnel, « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Produit | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Ø | Ø |

Grâce aux contraintes d'équipes, il est possible de spécifier qu'une équipe ne peut être sélectionnée pour un rôle que si celle-ci ne possède pas une ressource donnée. Victoire Jeton permet donc de différencier deux équipes : une équipe A avec un Jeton et une équipe B dont aucun des membres n'a de Jeton. La même règle permet d'obtenir l'égalité avec l'activité Egalité (cf. Tableau 36).

*Tableau 35 : Activité Victoire Jeton*

| Victoire Jeton | | | |
|---|---|---|---|
| Rôle prototypique | Vainqueur | Vainqueur_Bis | Perdant |
| Description | Joueur ayant encore un Jeton | Coéquipier du vainqueur | Équipe n'ayant plus de Jeton |
| Cardinalité | 1 | 1 | 2 |
| Contraintes | Ø | Ø | « Jeton » = 0 |
| Equipes | A | A | B |
| Consomme | « Jeton » dans l'inventaire personnel, « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | Ø | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Produit | « Victoire » | « Victoire » | « Défaite » |
| Requiert | Ø | Ø | Ø |
| Connaissances Ciblées | Ø | Ø | Ø |
| Objectifs | Produire « Victoire » (objectif réussi) | Produire « Victoire » (objectif réussi) | Produire « Victoire » (objectif échoué) |

*Tableau 36 : Activité Egalité*

| Egalité | | |
|---|---|---|
| Rôle prototypique | Equipe A | Equipe B |
| Description | Egalité | Egalité |
| Cardinalité | 2 | 2 |
| Contraintes | « Jeton » = 0 | « Jeton » = 0 |
| Equipes | A | B |
| Consomme | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Produit | « Egalité » | « Egalité » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Victoire » (objectif échoué) | Produire « Victoire » (objectif échoué) |

Les propriétés détectées au sein de chaque activité du TD4 sont disponibles en annexe section X.4.

## VII.3. Transformation des logs en Chemins Tracés

Comme évoqué en chapitre IV sur la modélisation et la détection des interactions dans des activités, notre approche nécessite des séquences d'activités pour être appliquée. Or les traces sauvegardées par le jeu dans ce que nous appelons les logs ne sont pas modélisées comme des séquences d'activités. Il est donc nécessaire de les transformer. Par ailleurs, lors de la présentation de notre ontologie en chapitre III, nous avons montré l'intérêt d'agréger des actions ensemble sous forme d'activité.

Les logs enregistrent chronologiquement les actions des joueurs et contiennent trois informations pour chacune d'elles : la nature des actions (déplacer un jeton dans une zone de validation, lancer une attaque, valider une réponse, etc…), le sujet des actions (quel joueur / équipe effectue l'action) et les ressources affectées par l'action (Jeton utilisé, réponse sélectionnée, formule testée, etc…). La chronologie des actions et les informations qui leur sont associées sont critiques pour permettre l'agrégation des logs sous forme de séquences

d'activités. Pour ce qui est de MP-LOG, l'ensemble des actions et informations qui leur sont associées sont disponibles en section X.6.

```
TIME_STAMP    12/19/17 11:38:02    USER_ID    3521617
TRACE_CONTENT    Incorrect_Proposition    Both    3    4
State {Blue Units:  Both_3 Blue_1 Blue_3 Both_4 Both_1 Green Units:}
```

*Figure 32 : Exemple d'action  dans les logs des joueurs*

La Figure 32 présente un exemple de log issu du TD2. Grâce à cette ligne, nous savons que les deux joueurs ont tenté ensemble les réponses 3 et 4 et ont échoué (indiqué par *Incorrect_Proposition* suivi de *Both*). Il est cependant intéressant de noter que le joueur Bleu a posé des jetons dans sa zone de réponse propre (*Blue_1* et *Blue_3*) en plus des réponses communes (*Both_3* et *Both_4*) et qu'en revanche, le joueur vert n'a placé aucune des réponses (pas d'informations après *Green Units*). Dans ce cas particulier, en regardant les actions précédentes du log, il s'avère que le joueur Vert n'a pas cherché à construire la solution, ni même à communiquer avec le joueur Bleu.

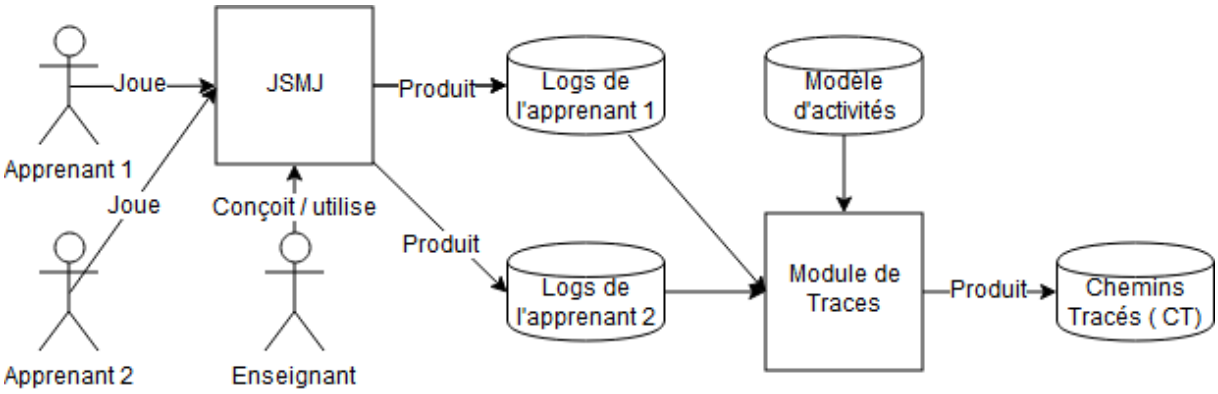


*Figure 33 : Transformations des logs en chemins tracés*

La transformation des logs des joueurs, présenté dans la  Figure 33 s'opère en trois étapes. La première étape consiste à rassembler les logs individuels des joueurs en un seul fichier cohérent. Avec les informations fournies par les logs, il est à priori possible de déterminer les rôles assumés par les joueurs (comme nous l'avons montré avec notre exemple d'action un peu plus haut). Notre deuxième étape nécessite une modélisation explicite des relations entre activités et actions. Elle nécessite également de bien comprendre comment conserver une chronologie dans nos activités une fois les actions agrégées.

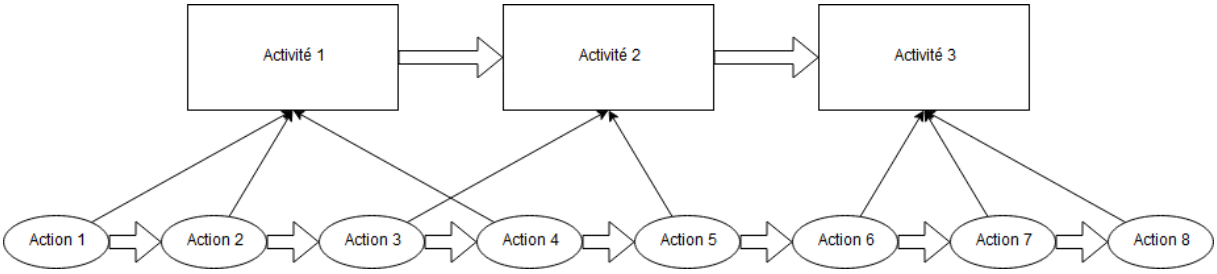


*Figure 34 : Exemple de factorisation d'actions et représentation de la chronologie obtenue*

La Figure 34 permet de mettre en lumière une problématique posée par l'agrégation des actions. En effet, nous pouvons constater que l'Action 3 est agrégée dans l'Activité 2 alors que l'Activité 1 n'est pas encore finie. Les intervalles de temps dans lesquels nos activités s'inscrivent peuvent donc se chevaucher (Activité 1 et 2 dans notre exemple). Nous considérons alors que l'activité A précède l'activité B si la dernière action de l'activité A précède la dernière action de l'activité B. Si un conflit devait apparaître entre les activités (utilisation de ressources et d'individus rendus disponibles par les actions dans la chronologie mais pas par l'activité par exemple), cela signifie que le niveau de granularité choisi pour les activités n'est pas pertinent et que celles-ci doivent probablement être subdivisées en de nouvelles activités de granularité inférieure.

Cette interface entre action et activité est dépendante du JSMJ et ne peut être généralisée. Voici toutefois un exemple de son fonctionnement. Désignons par $a_i$ avec $i \in \mathbb{N}$ les actions effectuées par les joueurs dans le cadre d'un JSMJ. Désignons par $J_p$ avec $p \in \mathbb{N}$ les joueurs prenant part au JSMJ. Enfin désignons par $A_k$ avec $k \in \mathbb{N}$ les activités du JSMJ.

- Soit $A_1$ une activité composée de deux rôles prototypiques $R_1$ et $R_2$. Le rôle prototypique $R_1$ sera détecté lorsque le joueur réalisera la séquence d'action $<a_1, a_2, a_3>$. Le rôle prototypique $R_2$ sera détecté lorsque le joueur réalisera la séquence d'action $<a_2, a_4>$.
- Soit $A_2$ une activité composée d'un rôle prototypique unique $R_3$. Ce rôle prototypique $R_3$ correspond à la séquence d'action $<a_3, a_4>$.

Considérons la séquence d'actions $< (J_1, a_1), (J_2, a_2), (J_1, a_2), (J_2, a_3), (J_2, a_4), (J_1, a_3)>$ où $(J_p, a_i)$ désigne que l'action $a_i$ a été réalisée par le joueur $J_p$. Si nous mettons en gras les actions que nous pouvons agréger en $A_1$, et en souligné les actions que nous pouvons agréger en $A_2$, nous obtenons : $< \mathbf{(J_1, a_1)}, \mathbf{(J_2, a_2)}, \mathbf{(J_1, a_2)}, \mathbf{\underline{(J_2, a_3)}}, \underline{(J_2, a_4)}, \mathbf{(J_1, a_3)}>$. L'activité $A_2$ finit donc avant l'activité $A_1$. Dans l'exemple l'action a3 peut être utilisée par nos deux activités, la séquence obtenue est donc $A_2$ -> $A_1$.

L'exemple donné en Figure 35 représente une séquence d'activité réellement effectués par les joueurs lors des expérimentations avec MP-LOG dans le cadre du TD2. Pour chaque activité de la séquence, nous précisons la nature de celle-ci (ex : Mauvaise_réponse2_Solo, Game_Over) suivi des joueurs impliqués et de leurs rôles.

```
['Mauvaise_réponse2_Solo', ['Bleus', 'Solver']] ->
['Mauvaise_réponse2_Solo', ['Bleus', 'Solver']] -> ['Mauvaise_réponse2_CA',
['Bleus', 'Solver'], ['Verts', 'Beneficiary']] -> ['Game_Over', ['Bleus',
'Players'], ['Verts', 'Players']]
```

*Figure 35 : Exemple de séquence d'activité produite par agrégation des actions des logs*

Une fois l'agrégation des actions en activités effectuée, nous sommes donc en présence d'une séquence (voir exemple au-dessus) chronologique des activités effectuées par les joueurs. Cette séquence peut cependant contenir des répétitions si une même activité a été effectuée plusieurs fois. De telles séquences (que nous qualifions de chemins tracés avec répétitions) ne peuvent être comparées aux séquences admissibles (voir chapitre V) obtenues par l'analyse du scénario qui elles sont dépourvues de répétitions. La dernière et troisième étape consiste donc à supprimer les répétitions d'activités des chemins tracés afin de ne conserver qu'une seule occurrence de chaque activité utilisée dans le chemin étudié.

Les spécifications permettant d'agréger les différentes actions liées à la validation pour le TD2 et le TD4 sont schématisées dans la Figure 36 qui se trouve en annexe section X.5.

## VII.4. Évaluation de la pertinence des propriétés détectées automatiquement au sein des logs des joueurs

Une fois que la modélisation de MP-LOG à l'aide de l'ontologie MPLGO est réalisée, il est possible d'appliquer l'algorithme de détection des propriétés d'interactions pouvant émerger des scénarios MP-LOG.

Un apport direct de cette recherche et qui n'est pas le sujet de cette thèse est de fournir aux enseignants, à posteriori, des informations sur les interactions qu'ont réellement effectuées les élèves, dans un scénario de JSMJ à partir de leurs traces d'interactions. L'analyse à posteriori suit les mêmes étapes de détection de propriétés que celles de l'analyse à priori d'un scénario mais appliqué à des chemins tracés.

Nous avons mené une expérimentation avec des étudiants de deuxième année de licence. Cette expérimentation avait deux objectifs :

1. Évaluer la pertinence des propriétés extraites par l'algorithme de détection automatique des propriétés d'interactions et leur degré de dépendances. Sont-elles vraiment différentes les unes des autres ?
2. Évaluer l'algorithme de calcul du sous-ensemble des séquences admissibles (SA) dans le but de vérifier que les séquences d'activités de SA contiennent l'ensemble des informations relatives aux interactions.

La section VII.4 présente l'analyse menée pour répondre au premier objectif. L'idée de cette analyse est de comparer les propriétés détectées automatiquement dans les logs des joueurs aux interactions déclarées par les joueurs eux-mêmes.

L'objectif 2 est traité en section VII.5 dans laquelle l'analyse menée consistait à comparer l'ensemble calculé SA aux chemins tracés des joueurs. Un chemin tracé correspond aux logs générés par les joueurs dans une session de jeu, c'est-à-dire, une suite ordonnée d'activités lors de la réalisation effective d'un scénario par les joueurs. Nous désignons un chemin tracé par CT. Deux critères ont été définis pour évaluer l'algorithme : (1) Le taux de couverture de SA : tous les chemins tracés générés par les joueurs sont-ils inclus dans les séquences de SA ? et (2) Les propriétés calculées à partir des CT sont-elles identiques et similairement réparties à celles calculées à partir des séquences de SA, c'est-à-dire, le SA révèle-t-il les mêmes interactions (en nature et en distribution) que l'analyse des CT ?

## VII.4.a. Protocole de l'expérimentation

Nous avons mené l'expérimentation avec 6 groupes d'étudiants en deuxième année de Licence Informatique. 114 étudiants : 89 hommes et 35 femmes, ont participé à l'expérimentation. Elle s'est déroulée sur une séance de 1h45 où les étudiants ont joué aux deux scénarios de MP-LOG. Les parties pouvant être relativement courtes, il était demandé aux joueurs de jouer plusieurs fois aux deux scénarios. Nous espérions que les étudiants adoptent différentes stratégies d'interactions d'une session à l'autre.

La session de 1h45 a été divisée en plusieurs périodes. Lors de la première, nous leur avons fourni une description complète de MP-LOG et de son fonctionnement pendant 10~15 min. Il est important de noter, qu'aucune consigne n'a été fournie aux étudiants quant à la manière de gagner ou aux différentes stratégies de jeu possibles. Le regroupement des étudiants pour jouer ensemble fut effectué de manière aléatoire, et nous leur avons demandé de ne pas communiquer oralement mais uniquement à travers le chat.

Le temps restant fut divisé en deux : la moitié des étudiants a joué sur le scénario TD2 pendant la première période, et l'autre moitié sur le scénario TD4. Pendant la seconde période, il fut demandé aux joueurs de changer de scénario. A la fin de chaque partie de jeu, que ce soit pour TD2 ou pour TD4, les étudiants devaient répondre à un questionnaire court pour donner leur ressenti par rapport aux interactions multi-joueurs qu'ils avaient pu expérimenter à travers la partie. Ce questionnaire se présentait sous la forme de trois items auxquels il fallait fournir une réponse comprise entre 1 et 5 (5 indiquant que l'item est important). Voici les questions posées aux étudiants avec trois opinions révélant le Conflit, le travail Ensemble ou la Coordination) :

1. Avez-vous été en conflit avec d'autres étudiants ? (Opinion Conflit)
2. Avez-vous travaillé ensemble pour résoudre les problèmes ? (Opinion Ensemble)
3. Avez-vous coordonné votre stratégie de jeu avec d'autres étudiants ? (Opinion Coordination)

Ces trois questions visaient à recueillir le ressenti des étudiants vis-à-vis de la partie qu'ils venaient d'effectuer. La question 1 visait à obtenir le ressenti des étudiants par rapport à la compétition (avec son partenaire ou avec ses adversaires). Les questions 2 et 3 portaient davantage sur des notions de coopération et collaboration, la question 2 étant davantage

centrée sur la collaboration verbale (résolution du problème) et la question 3 sur la collaboration ludique (déplacement des unités en jeu, etc…).

## VII.4.b. Données

Grâce aux logs récupérés lors de l'expérimentation, nous obtenons ainsi trois ensembles d'informations :

1. les Chemins Tracés (CT) effectués par les étudiants,
2. les propriétés détectées automatiquement sur les Chemins Tracés,
3. les opinions des joueurs .

Ce sont ces trois ensembles qui seront utilisés pour effectuer notre analyse.

Toutes les parties effectuées par les joueurs ont été tracées en temps réels (les actions des joueurs ont été instantanément inscrites dans les logs du jeu avec le *TimeStamp* correspondant). Nous avons d'abord nettoyé les traces (parties incomplètes, erreur de connexion, etc…). Chaque trace individuelle obtenue a ensuite été fusionnée aux traces des autres joueurs impliqués dans la partie. Nous avons ainsi obtenu une seule séquence d'actions chronologique par partie. Ces séquences incluent les actions de jeu des joueurs, leurs échanges sur le chat et les réponses aux questionnaires pour une partie (voir logs section VII.3). Au total, les apprenants ont effectué 132 séquences dans le TD2 et 62 dans le TD4. Nous désignerons ces séquences par le terme parties dans la suite du chapitre.

En plus de la modélisation des deux scénarios de MPLGO avec les concepts de l'ontologie, nous avons également nettoyé et transformé les logs des joueurs lors des sessions de jeu en chemins tracés. Les chemins tracés sont modélisés, à l'instar des séquences d'activités, avec les concepts de l'ontologie MPLGO. Ce travail nous a permis d'appliquer l'algorithme d'extraction de propriétés aux chemins tracés au même titre qu'aux séquences d'activités.

L'objectif du questionnaire posé à la fin de chaque partie était d'obtenir le ressenti des joueurs par rapport à la collaboration, la coopération et les différents conflits ayant eu lieu au cours de la partie. Or une partie de TD4 regroupe l'ensemble des activités effectuées par les quatre joueurs. Pour comparer la collaboration et la coopération ressenti par les deux joueurs d'une même équipe, il est nécessaire d'extraire les activités collaboratives et coopératives qu'ils ont effectués de la partie concernée (pour ne pas comparer leur ressenti avec des activités auxquelles ils n'ont pas pris part). Chaque partie issue des logs est transformée en chemin tracé (CT) conformément à la méthode décrite en section VII.3. Pour que les propriétés issues des CT du TD4 soit plus significative, nous avons donc divisé chacune des CT du TD4 en fonction de nos deux équipes, montant donc le total de CT à 256 (1 chemin tracé par partie de TD2 et 2 par partie de TD4). L'algorithme d'analyse des interactions fut ensuite appliqué sur chacun de ces CT.

Conformément à la méthodologie décrite section IV.3, nous avons sommé pour chaque CT les différentes propriétés détectées automatiquement et avons obtenu ainsi 256 vecteurs de taille 5. En faisant la somme de ces différents vecteurs nous obtenons ainsi le nombre total d'occurrences de chacune des propriétés pour le TD2 et le TD4 (cf. Tableau 37).

*Tableau 37 : Nombre d'occurrences des différentes propriétés détectées automatiquement au sein de MP-LOG*

| | Validation Commune | Construction Commune des Connaissances | Conflit Externe | Conflit Interne | Individualisme |
|---|---|---|---|---|---|
| TD2 | 653 | 543 | 0 | 180 | 386 |
| TD4 | 378 | 293 | 1065 | 161 | 46 |
| Total | 1031 | 836 | 1065 | 341 | 432 |

L'idée de cette expérimentation est de vérifier la cohérence des propriétés que nous avons détecté vis-à-vis des opinions recueillies auprès des joueurs dont la répartition des réponses est résumée dans le Tableau 38.

*Tableau 38 : Nombre de réponses associées à chacune des réponses du sondage sur les interactions*

| Réponse | Opinion Conflit | Opinion Ensemble | Opinion Coordination |
|---|---|---|---|
| 1 | 83 | 39 | 51 |
| 2 | 62 | 76 | 67 |
| 3 | 55 | 68 | 67 |
| 4 | 17 | 21 | 12 |
| 5 | 19 | 28 | 29 |

## VII.4.c. Méthode d'analyse

Pour chacune de nos 5 propriétés, nous avons calculé son nombre moyen d'occurrences au cours d'une session de jeu en fonction des opinions fournies par les étudiants. Grâce à ces résultats, nous devrions être en mesure de vérifier la cohérence de nos propriétés par rapport aux opinions exprimées par les étudiants dans leurs sondages.

## VII.4.d. Résultats

Le Tableau 39 résume les différentes informations concernant la propriété Validation Commune que nous avons associé à l'idée de coopération et de collaboration. Les différentes valeurs en gras représentent le maximum de chaque colonne d'opinion et les valeurs en italique représentent les valeurs minimales.

Les résultats obtenus dans le cadre de cette propriété semblent confirmer que nous pouvons associer la propriété Validation Commune aux notions de travailler ensemble et de coordination. En effet, les valeurs minimales obtenues dans les deux colonnes correspondantes sont associées aux réponses « 1 » et les réponses maximales sont associées

aux réponses « 4 » ou « 5 ». Cela signifie que les étudiants ayant indiqué avoir peu coordonnés leurs efforts ou peu travaillés ensemble ont eu peu d'occurrences de la propriété « Validation Commune » au contraire de ceux ayant indiqués avoir travailler ensemble et s'être coordonnés.

*Tableau 39 : Nombre moyen d'occurrences de la propriété Validation Commune pour chacune des réponses d'opinions*

| | Opinion Conflit | Opinion Ensemble | Opinion Coordination |
|---|---|---|---|
| 1 | 4,04819277 | *1,87179487* | *2,25490196* |
| 2 | *3,80645161* | 4,03947368 | 4,74626866 |
| 3 | 4,18181818 | 4,33823529 | 4,53731343 |
| 4 | **6** | 4,0952381 | **5,41666667** |
| 5 | 3,89473684 | **6,46428571** | 5,20689655 |

La Construction Commune des Connaissances telle que nous l'avons conçue est plus exigeante que la Validation Commune et dénote, selon nous, d'un marqueur collaboratif important. Dans le cadre de cette propriété, l'Opinion Coordination semble corroborer notre approche pour les réponses basses (« 1 ») mais pas pour les réponses hautes étant donné la proximité des occurrences moyennes associées aux réponses « 4 » et « 5 » avec la réponse « 2 ». En revanche, l'Opinion Ensemble nous fournit une claire distinction qui tend à valider notre approche dans le cadre de cette expérimentation.

*Tableau 40 : Nombre moyen d'occurrences de la propriété Construction Commune des Connaissances pour chacune des réponses d'opinions*

| | Opinion Conflit | Opinion Ensemble | Opinion Coordination |
|---|---|---|---|
| 1 | 3,54216867 | *1,41025641* | *1,98039216* |
| 2 | 3,43548387 | 3,44736842 | 4,29850746 |
| 3 | 3,69090909 | 3,73529412 | 3,86567164 |
| 4 | **4,76470588** | 3,47619048 | **4,58333333** |
| 5 | *2,68421053* | **5,92857143** | 4,13793103 |

L'Opinion Conflit peut être associé à un conflit interne ou externe à une équipe. La propriété Conflit Externe (Tableau 41) tel que nous l'avons conçue devrait nous permettre de détecter des conflits entre des étudiants appartenant à des équipes différentes. La réponse « 3 » pour l'Opinion Conflit est légèrement plus élevé que la réponse « 5 ». Il est cependant important de noter que la réponse « 1 » a de loin le nombre d'occurrence moyen le plus bas pour cette Opinion. Ainsi, bien que les résultats soient ici moins marqués qu'avec la Validation Commune et la Construction Commune des Connaissances, il semblerait que l'Opinion des joueurs corroborent le lien entre Conflit Externe et Conflit ressenti par les joueurs.

Nous noterons par ailleurs que les réponses associées à Opinion Ensemble et Opinion Coordination semble indiquer que les joueurs ayant produit du Conflit Externe se sont

coordonnés et ont travaillé ensemble. Le Conflit Externe n'était produit que dans le TD4 opposant deux équipes de deux joueurs, ce résultat semble cohérent (« nous » contre « eux »).

*Tableau 41 : Nombre moyen d'occurrences de la propriété Conflit Externe pour chacune des réponses d'opinions*

| | Opinion Conflit | Opinion Ensemble | Opinion Coordination |
|---|---|---|---|
| 1 | *1,98795181* | *4,53846154* | *2,80392157* |
| 2 | 6 | 6,21052632 | 5,3880597 |
| 3 | **9,27272727** | 5,14705882 | 6,71641791 |
| 4 | 7,64705882 | **7,14285714** | 5,16666667 |
| 5 | **9,26315789** | 6,28571429 | **8,34482759** |

Les résultats associés aux Conflit Interne (Tableau 42) sont particulièrement intéressants. L'Opinion Conflit semble ici ne pas du tout aller dans notre sens au contraire des Opinions Ensemble et Coordination. Ces deux dernières Opinions indiquent clairement que les joueurs ne se sont pas coordonnés et n'ont pas travaillé ensemble, ce qui serait logique dans le cas d'un conflit interne important entre les joueurs. Pourtant, l'Opinion Conflit semble indiquer que les joueurs considèrent ne pas avoir été en conflit avec d'autres étudiants. L'une de nos hypothèses pour expliquer ce phénomène résiderait dans l'interprétation qu'aurait eu les étudiants de la question sur le Conflit en comparaison avec le Conflit Externe du TD4.

Les résultats que nous avons obtenus concernant le Conflit Interne sont donc mitigés. D'un côté l'absence de coopération/collaboration attendu est bien représenté dans les réponses des joueurs, de l'autre le conflit lui-même semble absent.

*Tableau 42 : Nombre moyen d'occurrences de la propriété Conflit Interne pour chacune des réponses d'opinions*

| | Opinion Conflit | Opinion Ensemble | Opinion Coordination |
|---|---|---|---|
| 1 | **1,55421687** | **2,07692308** | **1,90196078** |
| 2 | 1,25806452 | 1,17105263 | 1,31343284 |
| 3 | 1,30909091 | 1,26470588 | 1,31343284 |
| 4 | 1,35294118 | 1,33333333 | 1,66666667 |
| 5 | *1,10526316* | *1,10714286* | *0,51724138* |

La propriété Individualiste dénote d'une absence d'interactions. Pour que nos Opinions corroborent celle-ci, il faudrait donc que les valeurs maximales correspondent aux réponses « 1 » (voir « 2 ») et les minimales aux valeurs « 4 » ou « 5 ». Or ce sont précisément les résultats que nous obtenons dans le Tableau 43. Ainsi donc les réponses obtenues dans nos sondages semblent indiquer que les joueurs ayant produit la propriété Individualiste n'ont pas été Conflit, n'ont pas travailler ensemble et n'ont pas coordonné leurs actions.

*Tableau 43 : Nombre moyen d'occurrences de la propriété Individualiste pour chacune des réponses d'opinions*

| | Opinion Conflit | Opinion Ensemble | Opinion Coordination |
|---|---|---|---|
| 1 | 2,01204819 | **3,58974359** | **3,37254902** |
| 2 | **3,43548387** | 1,60526316 | 1,59701493 |
| 3 | 0,21818182 | 1,88235294 | 0,62686567 |
| 4 | *0,88235294* | 0,66666667 | 2,25 |
| 5 | 1,36842105 | *0,25* | *0,68965517* |

### VII.4.e. Discussion des résultats

Les résultats que nous avons obtenus dans le cadre de cette expérimentation ne nous permettent pas d'affirmer de façon certaine que nos propriétés d'interactions soient bel et bien en accord avec les opinions exprimées par les joueurs sur les interactions. Toutefois, nous notons que dans l'ensemble ceux-ci corroborent notre vision (hormis pour le Conflit Interne).

Nous pensons donc que si les résultats obtenus sont encourageant et tendent à appuyer notre point de vue, il est nécessaire de mener à bien d'autres expérimentations. Nous pensons également que celles-ci devraient être menées avec une variété plus importante de propriétés et une formulation plus spécifique des questions liées au conflit.

## VII.5. Évaluation de l'algorithme de calcul du sous ensemble de séquences d'activités SA

En reprenant les traces obtenues dans l'expérimentation précédente (sans diviser les Traces du TD4 cette fois-ci), nous avons mené à bien une étude sur l'algorithme d'analyse à priori, cherchant à répondre à trois questions différentes et permettant de répondre à la QR3 :

1. Les séquences issues de l'analyse à priori d'un scénario de jeu sérieux multi-joueurs couvrent-elles bien les chemins tracés réels des joueurs ? Existent-ils des chemins tracés non-couverts ? Comment ces chemins tracés se répartissent-ils sur l'ensemble des séquences admissibles par le scénario ?
2. Quelles propriétés d'interactions apparaissent dans les séquences dérivées du scénario ?
3. Les propriétés obtenues dans les traces des joueurs sont-elles incluses dans celles trouvées par les séquences ? Leur répartition (type et nombre) est-elle similaire ?

### VII.5.a. Données

En plus des Chemins Tracés (CT) obtenus en agrégeant les actions des joueurs sous forme de séquences d'activités (cf. section VII.3), nous manipulons dans cette expérimentation l'ensemble des séquences du sous-ensemble (SA) obtenues à travers l'algorithme détaillé en

chapitre V pour les scénarios TD2 et TD4 décrit avec MPLGO. Le sous-ensemble de séquence (SA) ainsi obtenu regroupe 510 séquences valides pour le TD4 et 256 pour le TD2.

## VII.5.b. Résultats

*Tableau 44 : Nombre de séquences utilisées et CT non-couverts pour le TD2 et le TD4*

| | Nombre de séquences de SA valides | Nombre de CT | Nombre de séquences du SA utilisées | Nombre de CT non-couverts par SA |
|---|---|---|---|---|
| Scénario TD4 | 510 | 62 | 38 | 2 |
| Scénario TD2 | 256 | 132 | 35 | 0 |

Nous considérons un CT comme non-couvert si aucune des séquences valides du SA ne lui correspond. De plus, nous considérons une séquence du SA comme étant utilisée, si au moins un CT lui correspond. Le Tableau 44 résume les résultats obtenus qui montrent que l'algorithme proposé est capable de couvrir la majorité des CT effectués par les joueurs. Seuls 2 CT sur les 62 du TD4 n'ont pas été couverts soit 3%. Ce résultat est encourageant mais nous invite à chercher pourquoi ces 2 CT n'ont pas été identifiés. Dans (Guinebert et al., 2017), il est défini qu'un scénario peut permettre la réalisation de plusieurs objectifs apprenants. Après analyse le de ces 2 CT, il s'avère que ces deux sessions de jeu ont donné lieu à une égalité entre les joueurs. Or cet état final n'ayant pas été anticipé lors de la modélisation du scénario du TD4, des séquences permettant de réaliser cet « objectif » n'ont pas été générées par l'algorithme de calcul. Ce résultat nous a permis de réitérer sur la modélisation du scénario pour ajouter ce nouvel état final. Ces deux CT sont maintenant couverts par le nouvel SA.

Un deuxième résultat montre que de nombreuses séquences n'avaient pas été utilisées par les apprenants alors que certaines l'ont été de nombreuses fois (pour le TD4 par exemple, 8 CT correspondent à une seule séquence). Ce dernier point explique pourquoi le nombre de CT est supérieur au nombre de séquences utilisées du SA. Les joueurs ont suivi des stratégies similaires en dépit de la liberté qui leur était accordée (même constat pour le TD2). Enfin, cela montre qu'une analyse basée uniquement sur les traces ne permet pas nécessairement d'extraire toutes les interactions et stratégies possibles dans un scénario (sauf peut-être dans le cas de données importantes).

*Tableau 45 : Pourcentages de répartition des propriétés d'interactions dans les séquences du SA, dans les séquences utilisées du SA et dans les CT des joueurs*

| Propriétés (%) | TD2 | | | TD4 | | |
|---|---|---|---|---|---|---|
| | Séquences du SA | Séquences du SA utilisées | CT | Séquences du SA | Séquences du SA utilisées | CT |
| CE | 0 | 0 | 0 | 47.32 | 48.4 | 48.02 |
| CI | 20 | 17.34 | 15.14 | 10.53 | 10.56 | 10.96 |
| Ind | 20 | 16 | 22.69 | 10.53 | 6.15 | 4.24 |
| VC | 40 | 41.33 | 36.08 | 21.08 | 21.11 | 21.26 |
| CCC | 20 | 25.33 | 26.09 | 10.53 | 13.78 | 15.52 |

Aux résultats présentés dans le Tableau 37 (cf. section VII.4.b), nous avons appliqué le processus de détection de propriétés d'interactions sur les séquences du SA. Le Tableau 45 synthétise nos résultats. Nous pouvons observer, dans les deux scénarios TD2 et TD4, que les CT des apprenants ont une répartition de propriétés qui est différente de la répartition des mêmes propriétés dans les séquences du SA. Ceci montre que les étudiants ont privilégié certaines interactions. Par exemple pour le TD4, les propriétés Ind (Individualiste) et CI (Conflit interne) sont équi-présentes parmi les séquences du SA (10.53%) alors que dans les CT, les étudiants ont plutôt été en conflit interne (10.96%) qu'individualiste (4.24%).

Nous avons appliqué le test du khi2 pour vérifier l'indépendance des répartitions de propriétés, celle du SA et celle des CT des joueurs. L'idée étant alors la suivante : les apprenants sont libres de leurs actions et peuvent réutiliser plusieurs fois une même séquence du SA (comme indiqué plus haut). Ainsi, même si CT et SA utilisent la même modélisation, la répartition de leurs propriétés n'est pas obligatoirement similaire ou dépendante. Or, les résultats du test montrent clairement une dépendance entre les deux répartitions, et ceci dans le cas des deux scénarios TD2 et TD4. Ce résultat montre, dans le cas de cette étude, que les propriétés extraites à partir des séquences du SA sont similaires (en type mais aussi en distribution) aux propriétés que nous pouvons extraire des CT des joueurs. L'analyse, à priori, d'un scénario JSMJ permet donc de fournir des informations pertinentes sur les interactions. Ce résultat est important car il permet aux enseignants et aux concepteurs de scénarios de détecter les interactions de leurs scénarios, en amont de leur utilisation par des apprenants, et donc de pouvoir les modifier / corriger si ces interactions ne répondent pas à leurs attentes. Toutefois, ces résultats sont propres au contexte de l'étude présentée ici, d'autres études devraient être menées dans d'autres contextes pour les valider.

## VII.5.c. Analyse des résultats

Cette dernière analyse était de loin la plus importante de toutes car il s'agissait de montrer que les interactions détectées par notre algorithme à partir du scénario étaient valides et avaient du sens par rapport à ce que nos apprenants étaient en mesure de faire dans le scénario.

Les résultats que nous avons obtenus sont de trois ordres :

1. Nous avons été en mesure de couvrir presque tous les CTs des joueurs avec une modélisation incomplète, puis tous les CTs avec une modélisation corrigée.
2. Il existe une dépendance entre les répartitions d'interactions issues du SA et celles des CTs.
3. Il existe des divergences sur les proportions d'interactions entre SA et CT.

Notre **résultat 1** (cf. Tableau 44), nous indique que lorsque le scénario est correctement modélisé, tous les chemins effectués par les apprenants sont calculés par l'algorithme. Autrement dit, l'ensemble des combinaisons d'interactions que peuvent rencontrer les apprenants ont été envisagées par l'algorithme. Evidemment, nous pouvons arguer que l'expérimentation n'ayant porté que sur MP-LOG, ces résultats sont à nuancer. Toutefois, ils restent très encourageants.

Le fait que nous n'ayons pas été en mesure de couvrir l'ensemble des CT dans un premier temps montre les limites de ce travail. Cette limite est directement liée au fait que nous n'avons pas, à ce jour, de moyen de vérifier que le scénario soit correctement modélisé. De fait, le développement ou la conception d'un tel moyen / méthode nous semble être une perspective intéressante à ce travail.

Le **résultat numéro 2** (cf. Tableau 45), quant à lui, établit que les proportions d'interactions des CTs et SA suivent des règles de répartitions similaires. Nous pouvons donc en déduire que les informations relatives aux interactions fournies par le SA, permettent d'entrevoir les possibilités d'interactions dans les CTs (bien que nous ne puissions pas les prévoir).

Dans la section V.1.c, nous avons procédé à une réduction importante du nombre de séquences étudiées et avions émis l'hypothèse que la perte d'information sur les interactions occasionnée par cette réduction serait limitée. Les **résultats 1 et 2** indique que, dans le cas de notre exemple, cette hypothèse est vérifiée. Bien sûr cette hypothèse nécessite d'être confrontée à d'autres situations expérimentales pour être pleinement vérifiée.

Pour finir, les divergences que nous avons obtenues entre SA et CT sont particulièrement intéressantes. Le SA est issue d'une heuristique se basant uniquement sur la faisabilité des séquences et non sur les interactions utilisées par celle-ci. De fait, cela suggère que les propriétés issues de SA peuvent indiquer les proportions que nous obtiendrions chez les apprenants si ceux-ci n'étaient pas influencés par certaines interactions. Or les divergences que nous obtenons nous indiquent les interactions que les apprenants ont volontairement évité ou recherché dans le scénario. L'exemple Validation Commune (VC) et Construction Commune des Connaissances (CCC) est particulièrement parlant. Ces deux propriétés sont liées : nous ne pouvons avoir de Construction Commune des Connaissances sans Validation Commune. De fait, puisqu'il existe dans nos scénarios exactement le même nombre de chemins permettant d'atteindre CCC+VC que de chemin permettant d'atteindre Validation Commune seul. Nous avons deux fois plus de Validation Commune que de Construction Commune des Connaissances dans les SA du TD2 et du TD4. Ces deux proportions ne sont pas respectées dans les CT, nous y voyons en effet un clair favoritisme de Construction Commune des Connaissances. Or Construction Commune des Connaissances est un composant fondamental de la collaboration, nous pouvons donc supposer que les apprenants ont favorisé la collaboration à la coopération lorsqu'ils en avaient l'opportunité. Deux hypothèses que

nous avons émises pour expliquer cette supposition sont que la structure du scénario pousserait les joueurs à certains types d'interactions, ou bien que leurs préférences personnelles les conduiraient à choisir un comportement plutôt qu'un autre. En absence de données complémentaires, il ne nous est pas possible de confirmer ou d'infirmer ces hypothèses.

# VIII. Conclusions et Perspectives

## VIII.1. Conclusions sur MPLGO

Il nous est apparu, que pour être en mesure de détecter automatiquement des interactions multi-joueurs, nous allions devoir réfléchir à plusieurs questions de recherche dont la question suivante : Quel modèle de scénario adopter pour permettre la description puis la détection d'interactions entre pairs ? Nous avions émis l'hypothèse qu'il était possible d'analyser les interactions effectuées par les joueurs en se basant sur les éléments de *game design* caractérisant le scénario. Aussi, nous nous sommes mis en quête d'un modèle à même de formaliser des scénarios relativement complexes et dont le fonctionnement serait centré autour des actions du jeu.

Aucune des solutions présentées dans la littérature ne semble répondre aux contraintes que nous avons identifiées et nous avons par conséquent décidé de proposer un modèle. Celui-ci utilise quelques concepts liés à la théorie de l'activité et des systèmes concurrents et centre son fonctionnement sur la description des ressources produites et consommées par les joueurs (ainsi que les connaissances / compétences travaillées) lorsqu'ils participent à l'activité. Pour représenter cela, nous avons amené la notion de rôle prototypique correspondant à un comportement type d'un joueur dans le cadre d'un scénario de JSMJ.

Afin d'évaluer notre modèle, nous l'avons utilisé pour formaliser quatre scénarios de jeux sérieux basés sur *Voracy Fish, Learning Adventure, ByteBattle* et *ClassCraft*. Les résultats que nous avons alors obtenus étaient satisfaisants bien qu'ayant montré des limitations (chapitre VI). Une version de l'ontologie palliant à certaines de ces limitations en utilisant une grammaire formelle a été présentée dans l'article (Guinebert et al., 2017). Cependant, cette solution rend variables les consommations et les productions de ressources au sein des activités (elles ne sont plus nécessairement fixées à l'avance). Nos algorithmes présentés dans le chapitre V, nécessitent de connaitre à l'avance ces productions et consommations. Le développement de nouveaux algorithmes capables de prendre en compte cette grammaire formelle constitue donc une perspective intéressante à ce travail.

L'évaluation de l'ontologie proposée s'est poursuivie avec les scénarios de *MultiPlayer LOgic Game* (MP-LOG), un jeu que nous avons développé dans le cadre de cette thèse pour répondre à nos besoins expérimentaux. En effet, pour évaluer l'ensemble de nos contributions, nous avions besoin d'un jeu répondant à plusieurs critères :

- Le jeu doit offrir une liberté d'actions importante permettant des interactions de différentes natures.
- Le jeu doit fournir des traces des actions des joueurs.
- Le jeu doit être open source.
- Le jeu doit être utilisable dans le cadre universitaire.

Or n'ayant trouvé aucun jeu satisfaisant ces quatre critères, nous avons dû créer le nôtre.

Nous avons été en mesure de représenter l'ensemble des scénarios testés avec notre modèle (*Voracy Fish, Learning Adventure, ByteBattle, ClassCraft et MP-LOG)* et ce en dépit des

limitations relevées plus tôt, en dépit de la variété structurelle des scénarios et en dépit de la variabilité des scénarios testés quant aux interactions.

Il est cependant important de noter que ces tests ne garantissent pas la réutilisabilité aisée de cette ontologie pour d'autres scénarios et pour des utilisateurs n'ayant pas participé à la conception de cette dernière. Des tests plus poussés allant dans ce sens sont donc nécessaires pour garantir la validité de ce modèle.

## VIII.2. Conclusions sur le *framework* pour la modélisation des interactions

Une des questions que nous nous étions posés, était : Qu'est-ce qu'une interaction ? Cette question s'est rapidement transformée en « Comment pouvons-nous modéliser formellement les interactions entre pairs ? ». L'une des problématiques liées à cette question vient du fait qu'il n'existe pas de définitions consensuelles pour les différentes interactions. En effet, en fonction du domaine ou de l'individu, certaines interactions ne se reposent pas sur les mêmes critères. La notion de gain et de perte par exemple, est centrale en théorie des jeux mais pas dans le domaine des sciences de l'éducation. Ainsi, les définitions de collaboration de ces deux domaines s'en trouvent rigoureusement divergentes.

Plutôt que de chercher à recueillir l'ensemble des définitions et visions existantes pour nos différentes interactions, nous avons choisi d'attaquer le problème par un angle différent. Il nous est apparu, que les différentes définitions auxquelles nous étions confrontés pouvaient être décomposées en propriétés de granularité plus fines. Certaines de ces propriétés sont communes aux différentes définitions, d'autres sont propres à une définition donnée.

La formalisation de nos propriétés s'appuie sur le modèle de scénarisation MPLGO. Les propriétés d'interaction peuvent donc ainsi être détectées automatiquement sur des activités respectant l'ontologie que nous avons créée. Nous avons alors mis en place une expérimentation afin de vérifier que les propriétés d'interactions que nous détections soient en accord avec les interactions déclarées par les apprenants.

Cette expérimentation a été menée auprès de 114 étudiants avec MP-LOG. Elle nous a fourni des résultats quant aux interactions ressenties par les étudiants (à travers un sondage de fin de partie) et quant aux propriétés d'interactions détectées dans les chemins tracés (séquences d'activités reproduites à partir des traces) des étudiants. La comparaison effectuée nous a donné des résultats mitigés. En effet, nous avons pu lier efficacement les opinions des joueurs à 4 de nos propriétés. Nous avons en revanche obtenu des résultats contradictoires pour la propriété Conflit Interne.

Au vu de nos résultats, nous pensons que de plus amples expérimentations sont nécessaires pour déterminer la fiabilité de notre *framework* à détecter des interactions cohérentes vis-à-vis du ressenti des apprenants malgré des résultats allant en partie dans ce sens. Selon nous, deux pistes intéressantes à suivre pour mener à bien ces expérimentations serait d'augmenter le nombre de propriétés différentes à étudier, et de poser des questions plus spécifiques aux apprenants au sujet de leurs opinions sur les interactions (en particulier le conflit).

# VIII.3. Conclusions sur l'analyse de scénarios

Une hypothèse que nous avons émise, indiquait qu'il était possible de déterminer les interactions pouvant avoir lieu dans un scénario à condition de pouvoir en analyser sa description. En effet, certains scénarios contraignent les joueurs à employer des ressources spécifiques pour finir le jeu. Or, si l'obtention de ces ressources ne peut se faire qu'avec un effort commun, nous créerons de la collaboration ou de la coopération, et si cette même ressource est en quantité insuffisante pour que tous les joueurs gagnent, nous créerons de la compétition.

Avec nos deux précédentes contributions, nous étions à même de détecter automatiquement des interactions au sein d'une séquence d'activité, ainsi que de décrire formellement un scénario de jeu sérieux. L'idée que nous avons alors eu a été d'analyser en amont le scénario en extrayant de celui-ci les différentes séquences d'activités faisables pour atteindre la fin du jeu. L'algorithme résultant de l'approche que nous avons choisie, a nécessité de nombreux élagages et simplifications pour en permettre une utilisation efficace d'un point de vue calculatoire sur des scénarios complexes offrant une grande liberté aux joueurs.

En dépit de ces simplifications, les résultats que nous avons obtenus en comparant l'analyse des séquences extraites du jeux MP-LOG (cf. section VIII.1) aux chemins tracés issus de l'expérimentation avec les étudiants (cf. section VIII.2), sont particulièrement encourageants. En effet, nous avons été en mesure de couvrir l'ensemble des chemins tracés des étudiants. De plus, les propriétés d'interactions extraites du sous-ensemble suivent une distribution similaire à celles extraites des chemins tracés. Autrement dit, notre algorithme a été en mesure de trouver l'ensemble des interactions pouvant être effectuées par les joueurs et aussi d'obtenir des distributions de propriétés d'interaction similaires, indiquant une capacité à partiellement prédire les interactions pouvant émerger du scénario.

Par ailleurs, les différentes divergences de distributions obtenues sur nos deux ensembles, permettent d'inférer les comportements des apprenants vis-à-vis des interactions. En effet, les séquences calculées par l'algorithme ne sont pas influencées par les stratégies adoptées ou préférées des apprenants, contrairement aux chemins tracés. Un exemple parlant, étant le rapport entre les deux propriétés Validation Commune (VC) et Construction Commune des Connaissances (CCC). En effet, dans le cadre de ces propriétés, il est impossible d'obtenir Construction Commune des Connaissances sans Validation Commune, mais il est possible d'avoir l'inverse. Lors de la conception de notre jeu, nous avons donné la possibilité aux joueurs de faire Validation Commune sans Construction Commune des Connaissances, ou de faire Construction Commune des Connaissances (et donc nécessairement Validation Commune). Pour ce qui est de l'analyse de nos scénarios, cela s'est traduit par une proportion de Validation Commune égale à exactement deux fois la proportion de Construction Commune des Connaissances (Construction Commune des Connaissances impliquant Validation Commune, il est normal d'avoir VC = 2*CCC). Or cette proportion n'est pas respectée chez les apprenants, le rapport obtenu entre Validation Commune et Construction Commune des Connaissances étant d'environ 1.4. Cette divergence de résultat indique chez

les apprenants une tendance à privilégier la collaboration (marquée par Construction Commune des Connaissances) à la coopération (marquée par Validation Commune seule).

En définitive, les résultats de notre expérimentation vont dans notre sens et appuient nos différentes hypothèses. Cependant celle-ci souffre d'un certain nombre de biais dont le principal étant que la formalisation du scénario, la formalisation des propriétés et l'algorithme ont été effectués par la même personne. Ainsi, d'autres expérimentations sont nécessaires pour garantir de la pertinence de nos résultats.

## VIII.4. Perspectives

L'ensemble de nos contributions ont permis d'avancer sur les différentes questions que nous nous étions posées en vue de détecter automatiquement des interactions entre pairs au sein d'un JSM. Toutefois ces contributions nécessitent davantage d'évaluations et une perspective serait donc de réutiliser les modèles de scénario et d'activités et d'appliquer les algorithmes avec d'autres scénarios de JSMJ.

L'ontologie a montré de sérieuses limites tant au niveau de sa prise en main que de son potentiel. Une perspective intéressante serait de rendre celle-ci plus accessible ou de garantir que sa version utilisant une grammaire formelle puisse être employée conjointement à l'algorithme d'extraction des séquences d'un scénario.

Une critique similaire peut-être émise concernant la description des propriétés. Leur fonctionnement est relativement simple, toutefois leur formalisation repose entièrement sur l'ontologie dont nous avons soulevé les problèmes de prise en main. Rendre la description des propriétés indépendantes de l'ontologie ou la rendre plus accessible à des utilisateurs non informaticiens semble une perspective intéressante.

Lors de l'expérimentation, une partie des traces des étudiants n'avait pas été couverte par l'algorithme de calcul du sous-ensemble de séquences à cause d'une modélisation imparfaite du jeu. Il serait intéressant de réfléchir à une méthodologie garantissant que la description du jeu à laquelle nous sommes parvenus couvre bien toutes les sorties du jeu.

La détection en amont des interactions émergeant d'un scénario fournit des informations inédites sur le jeu. Les informations apportées sont complémentaires aux informations obtenues à partir des traces des joueurs. Ces deux groupes d'informations n'ont ni la même signification ni le même impact sur l'utilisation ou le développement du jeu. Aussi il nous semblerait intéressant de travailler à la création d'un processus itératif se reposant sur ces deux approches complémentaires pour faire progresser et améliorer l'identification des différentes interactions chez les joueurs.

# IX. Bibliographie


Alvarez, J. (2007). *Du jeu vidéo au serious game : Approches culturelle, pragmatique et formelle* (PhD Thesis). Toulouse 2.

Alvarez, J., Rampnoux, O., Jessel, J.-P., & Methel, G. (2007). Serious Game : Just a question of posture. *Artificial & Ambient Intelligence, AISB*, *7*, 420–423.

Avouris, N., Fiotakis, G., Kahrimanis, G., Margaritis, M., & Komis, V. (2007). Beyond logging of fingertip actions : Analysis of collaborative learning using multiple sources of data. *Journal of Interactive Learning Research*, *18*(2), 231–250.

Beaudouin-Lafon, M., Mackay, W. E., Andersen, P., Janecek, P., Jensen, M., Lassen, M., … Ratzer, A. (2001). CPN/Tools : A post-WIMP interface for editing and simulating coloured Petri nets. *International Conference on Application and Theory of Petri Nets*, 71–80. Springer.

Bell, B. S., & Kozlowski, S. W. (2007). Advances in technology-based training. *Managing human resources in North America*, 27–42.

Brightman, H. J. (2006). Mentoring faculty to improve teaching and student learning. *Issues in Accounting Education*, *21*(2), 127–146.

Burguillo, J. C. (2010). Using game theory and competition-based learning to stimulate student motivation and performance. *Computers & Education*, *55*(2), 566–575.

Carron, T., Marty, J.-C., & Heraud, J.-M. (2008). Teaching with game-based learning management systems : Exploring a pedagogical dungeon. *Simulation & Gaming*, *39*(3), 353–378.

Chomsky, N. (1957). *The structure of language*.

Corbel, A., Jaillon, P., Serpaggi, X., Baker, M., Quignard, M., Lund, K., & Séjourné, A. (2003). DREW : Un outil Internet pour créer des situations d'apprentissage coopérant. *Actes de la conférence EIAH 2003*, 109–113.

David, J.-P., George, S., Godinet, H., & Villiot-Leclercq, E. (2008). Scénariser une situation d'apprentissage collective instrumentée : Réalités, méthodes et modèles, quelques pistes. *International journal of Technologies in Higher Education*, *4*(2), p–72.

Derntl, M., Neumann, S., Griffiths, D., & Oberhuemer, P. (2012). The conceptual structure of IMS learning design does not impede its use for authoring. *IEEE Transactions on Learning Technologies*, *5*(1), 74–86.

Deutsch, M. (1949). A theory of co-operation and competition. *Human relations*, *2*(2), 129–152.

Dillenbourg, P. (1999). *What do you mean by collaborative learning?* Oxford: Elsevier.

Dillenbourg, P., Baker, M. J., Blaye, A., & O'Malley, C. (1995). *The evolution of research on collaborative learning*. Elsevier, Oxford.

Djaouti, D., Alvarez, J., & Jessel, J.-P. (2011). Classifying serious games : The G/P/S model. In *Handbook of research on improving learning and motivation through educational games : Multidisciplinary approaches* (p. 118–136). IGI Global.

Doignon, J.-P., & Falmagne, J.-C. (1985). Spaces for the assessment of knowledge. *International journal of man-machine studies*, *23*(2), 175–196.

Dyke, G., Girardot, J.-J., Lund, K., & Corbel, A. (2007). Analysing face to face computer-mediated interactions. *EARLI (European Association for Research, Learning and Instruction), 12th Biennial International Conference*.

Edwards, A., & Westgate, D. P. (2005). *Investigating classroom talk*. Routledge.

Engeström, Y. (2001). Expansive learning at work : Toward an activity theoretical reconceptualization. *Journal of education and work*, *14*(1), 133–156.

Engeström, Y. (2014). *Learning by expanding*. Cambridge University Press.

Eseryel, D., Law, V., Ifenthaler, D., Ge, X., & Miller, R. (2013). An investigation of the interrelationships between motivation, engagement, and complex problem solving in game-based learning. *Educational technology & society*, *17*(1), 42–53.

Fabricatore, C. (2000). *Learning and videogames : An unexploited synergy*.

Falmagne, J.-C., Cosyn, E., Doignon, J.-P., & Thiéry, N. (2006). The assessment of knowledge, in theory and in practice. In *Formal concept analysis* (p. 61–79). Springer.

Fikes, R. E., & Nilsson, N. J. (1971). STRIPS : A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, *2*(3‑4), 189–208.

Gendron, E., Carron, T., & Marty, J.-C. (2008). Collaborative Indicators in Learning Games : An immersive factor. *2nd European Conference on Games Based Learning, Barcelona, Spain*, 16–17.

Georgeon, O., Mathern, B., Mille, A., Bellet, T., Bonnard, A., Henning, N., & TRÉMAUX, J. (2007). *Abstract Analysis of Behavior and Situation for menTal Representation Assessment and Cognitive acTivity modelling*.

Guinebert, M., Yessad, A., Muratet, M., & Luengo, V. (2017). An ontology for describing scenarios of multi-players learning games : Toward an automatic detection of group interactions. *European Conference on Technology Enhanced Learning*, 410–415. Springer.

Hermans, H., Janssen, J., & Koper, R. (2016). Flexible authoring and delivery of online courses using IMS Learning Design. *Interactive Learning Environments*, *24*(6), 1265–1279.

Hummel, H., Manderveld, J., Tattersall, C., & Koper, R. (2004). Educational modelling language and learning design : New opportunities for instructional reusability and personalised learning. *International Journal of Learning Technology*, *1*(1), 111–126.

Interactive, G. (2012). *Voracy Fish : New multiplayer serious game for physical rehabilitation (2012)*.

Jean-Paul Doignon, J., & Falmagne, J. (1999). Knowledge Spaces. *Berlin: Heidelberg. Germay: Springer-Verlag*.

Johnson, R. T., & Johnson, D. W. (1988). Cooperative learning : Two heads learn better than one. *Transforming education*, *18*, 34.

Kozar, O. (2010). Towards Better Group Work : Seeing the Difference between Cooperation and Collaboration. *English Teaching Forum*, *48*, 16–23. ERIC.

Kreijns, K., Kirschner, P. A., & Jochems, W. (2003). Identifying the pitfalls for social interaction in computer-supported collaborative learning environments : A review of the research. *Computers in human behavior*, *19*(3), 335–353.

Kumpulainen, K., & Mutanen, M. (1999). The situated dynamics of peer group interaction : An introduction to an analytic framework. *Learning and instruction*, *9*(5), 449–473.

Magee, J., & Kramer, J. (1999). *State models and java programs*. wiley New York.

Maharg, P., & Nicol, E. (2009). Cyberdam and SIMPLE : A study in divergent developments and convergent aims. In *Learning in a virtual world : Reflections on the Cyberdam Research and Development Project*. The Australian National University.

Manna, Z., & Pnueli, A. (2012). *The temporal logic of reactive and concurrent systems : Specification*. Springer Science & Business Media.

Marne, B. (2014). *Modèles et outils pour la conception de jeux sérieux : Une approche meta-design* (PhD Thesis). Université Pierre et Marie Curie-Paris VI.

Marty, J., & Carron, T. (2011). Observation of Collaborative Activities in a Game-Based Learning Platform. *IEEE Transactions on Learning Technologies*, *4*(1), 98 - 110. https://doi.org/10.1109/TLT.2011.1

McAndrew, P., Goodyear, P., & Dalziel, J. (2006). Patterns, designs and activities : Unifying descriptions of learning structures. *International Journal of Learning Technology*, *2*(2-3), 216–242.

Michel, C., Garrot, E., & George, S. (2007). Situations d'apprentissage collectives instrument$\backslash$'ees:$\backslash$'etude de pratiques dans l'enseignement sup$\backslash$'erieur. *arXiv preprint arXiv:0707.3014*.

Moore, M. G. (1989). *Three types of interaction*. Taylor & Francis.

Morrison, A., Tennent, P., & Chalmers, M. (2006). Coordinated visualisation of video and system log data. *Fourth International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV'06)*, 91–102. IEEE.

Mortara, M., Catalano, C. E., Bellotti, F., Fiucci, G., Houry-Panchetti, M., & Petridis, P. (2014). Learning cultural heritage by serious games. *Journal of Cultural Heritage*, *15*(3), 318–325.

Muratet, M., Delozanne, E., Torguet, P., & Viallet, F. (2012). Serious game and students' learning motivation : Effect of context using prog&play. *international conference on Intelligent Tutoring Systems*, 123–128. Springer.

Muratet, M., Yessad, A., & Carron, T. (2016). Framework for learner assessment in learning games. *European Conference on Technology Enhanced Learning*, 622–626. Springer.

Nüssli, M.-A., Jermann, P., Sangin, M., & Dillenbourg, P. (2009). Collaboration and abstract representations : Towards predictive models based on raw speech and eye-tracking data. *Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1*, 78–82. International Society of the Learning Sciences.

Panitz, T. (1999). *Collaborative versus Cooperative Learning : A Comparison of the Two Concepts Which Will Help Us Understand the Underlying Nature of Interactive Learning.*

Paraskeva, F., Mysirlaki, S., & Papagianni, A. (2010). Multiplayer online games as educational tools : Facing new challenges in learning. *Computers & Education*, *54*(2), 498–505.

Portugal, J. N. (2006). Le Rapprochement du Jeu et de l'Apprentissage. *Serious Games Summit Europe 2006*.

Rapoport, A., Chammah, A. M., & Orwant, C. J. (1965). *Prisoner's dilemma : A study in conflict and cooperation* (Vol. 165). University of Michigan press.

Reisig, W. (2012). *Petri nets : An introduction* (Vol. 4). Springer Science & Business Media.

Roberts, T. S. (2004). *Online collaborative learning : Theory and practice*. IGI Global.

Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. *Computer supported collaborative learning*, 69–97. Springer.

Roth, W.-M., & Lee, Y.-J. (2007). “Vygotsky’s neglected legacy” : Cultural-historical activity theory. *Review of educational research*, *77*(2), 186–232.

Salen, K., Tekinbaş, K. S., & Zimmerman, E. (2004). *Rules of play : Game design fundamentals*. MIT press.

Sanchez, E., Young, S., & Jouneau-Sion, C. (2017). Classcraft : From gamification to ludicization of classroom management. *Education and Information Technologies*, *22*(2), 497–513.

Slootmaker, A., Kurvers, H., Hummel, H. G., & Koper, R. (2014). Developing Scenario-based Serious Games for Complex Cognitive Skills Acquisition : Design, Development and Evaluation of the EMERGO Platform. *J. UCS*, *20*(4), 561–582.

Sourmelis, T., Ioannou, A., & Zaphiris, P. (2017). Massively Multiplayer Online Role Playing Games (MMORPGs) and the 21st century skills : A comprehensive research review from 2010 to 2016. *Computers in Human Behavior*, *67*, 41–48.

Turkay, S., Hoffman, D., Kinzer, C. K., Chantes, P., & Vicari, C. (2014). Toward understanding the potential of games for learning : Learning theory, game design characteristics, and situating video games in classrooms. *Computers in the Schools*, *31*(1‑2), 2–22.

Villiot-Leclercq, E. (2007). La méthode des Pléiades : Un formalisme pour favoriser la transférabilité et l’instrumentation des scénarios pédagogiques. *Sciences et Technologies de l’Information et de la Communication pour l’Éducation et la Formation*, *14*(1), 117–154.

Villiot-Leclercq, E., & David, J.-P. (2007). Le formalisme des Pléiades pour la conception et l’adaptation de patrons de scénarios pédagogiques. *Actes de la conférence EIAH 2007*. INRP.

Vygotsky, L. (1978). Interaction between learning and development. *Readings on the development of children*, *23*(3), 34–41.

Wendel, V., Gutjahr, M., Göbel, S., & Steinmetz, R. (2013). Designing collaborative multiplayer serious games : Escape from Wilson Island—A multiplayer 3D serious game for collaborative learning in teams. *Education and Information Technologies*, *18*(2), 287-308. https://doi.org/10.1007/s10639-012-9244-6

Wouters, P., Van Nimwegen, C., Van Oostendorp, H., & Van Der Spek, E. D. (2013). A meta-analysis of the cognitive and motivational effects of serious games. *Journal of educational psychology*, *105*(2), 249.

Zagal, J. P., Rick, J., & Hsi, I. (2006). Collaborative games : Lessons learned from board games. *Simulation & Gaming*, *37*(1), 24–40.

Zyda, M. (2005). From visual simulation to virtual reality to games. *Computer*, *38*(9), 25–32.

# X. Annexes

## X.1. Modélisation du TD2

| Bonne_réponse_VC | | |
|---|---|---|
| Rôle prototypique | Constructeur | Bénéficiaire |
| Description | Joueur construisant la Formule | Joueur ne faisant que valider |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Défenseur » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé | « Défenseur » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Défenseur »<br>(objectif réussi) | Produire<br>« Défenseur »<br>(objectif réussi) |

| Bonne_réponse CCC | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 2 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A |
| Produit | « Défenseur » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé |
| Requiert | Ø |
| Connaissances Ciblées | « Logique propositionnelle» |
| Objectifs | Produire « Défenseur »<br>(objectif réussi) |

| Bonne_réponse Ind | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 1 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A |
| Produit | « Défenseur » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé |
| Requiert | Ø |
| Connaissances Ciblées | « Logique propositionnelle» |
| Objectifs | Produire « Défenseur »<br>(objectif réussi) |

| Bonne_réponse_CI | | |
|---|---|---|
| Rôle prototypique | Constructeur | Lésé |
| Description | Joueur construisant la Formule | Joueur n'ayant pu valider en dépit de sa participation |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Défenseur » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Défenseur » (objectif réussi) | Produire « Défenseur » (objectif échoué) |

| Mauvaise_réponse CCC | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 2 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A |
| Produit | « Formule consommée » dans l'inventaire partagé |
| Requiert | Ø |
| Connaissances Ciblées | « Logique propositionnelle» |
| Objectifs | Produire « Défenseur » (objectif échoué) |

| Mauvaise_réponse_VC | | |
|---|---|---|
| Rôle prototypique | Constructeur | Bénéficiaire |
| Description | Joueur construisant la Formule | Joueur ne faisant que valider |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Formule consommée » dans l'inventaire partagé | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Défenseur » (objectif échoué) | Produire « Défenseur » (objectif échoué) |

| Mauvaise_réponse Ind | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 1 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A |
| Produit | « Formule consommée » dans l'inventaire partagé |
| Requiert | Ø |
| Connaissances Ciblées | « Logique propositionnelle» |
| Objectifs | Produire « Défenseur » (objectif échoué) |

| Mauvaise_réponse_CI | | |
|---|---|---|
| Rôle prototypique | Constructeur | Lésé |
| Description | Joueur construisant la Formule | Joueur n'ayant pu valider en dépit de sa participation |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Formule consommée » dans l'inventaire partagé | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Défenseur » (objectif réussi) | Connaissances Ciblées « Logique propositionnelle » (objectif échoué) |

| Fin | |
|---|---|
| Rôle prototypique | Joueur |
| Description | Activité de Fin |
| Cardinalité | 2 |
| Contraintes | Ø |
| Equipes | Ø |
| Consomme | Ø |
| Produit | « Fin » |
| Requiert | Ø |
| Connaissances Ciblées | Ø |
| Objectifs | Ø |

| Fin Formule | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 2 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule consommée » dans l'inventaire partagé de l'équipe A |
| Produit | « Fin » |
| Requiert | Ø |
| Connaissances Ciblées | Ø |
| Objectifs | Ø |

# X.2. Modélisation du TD4

| Bonne_réponse CCC | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 2 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A |
| Produit | « Jeton » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé |
| Requiert | Ø |
| Connaissances Ciblées | « Logique propositionnelle» |
| Objectifs | Produire « Jeton »<br>(objectif réussi) |

| Bonne_réponse_VC | | |
|---|---|---|
| Rôle prototypique | Constructeur | Bénéficiaire |
| Description | Joueur construisant la Formule | Joueur ne faisant que valider |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Jeton » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé | « Jeton » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Jeton »<br>(objectif réussi) | Produire « Jeton »<br>(objectif réussi) |

| Bonne_réponse Ind | |
|---|---|
| Rôle prototypique | Constructeur |
| Description | Joueur construisant la Formule |
| Cardinalité | 1 |
| Contraintes | Ø |
| Equipes | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A |
| Produit | « Jeton » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé |
| Requiert | Ø |
| Connaissances Ciblées | « Logique propositionnelle» |
| Objectifs | Produire « Jeton »<br>(objectif réussi) |

| Bonne_réponse_CI | | |
|---|---|---|
| Rôle prototypique | Constructeur | Lésé |
| Description | Joueur construisant la Formule | Joueur n'ayant pu valider en dépit de sa participation |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | A |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Jeton » dans son inventaire personnel et « Formule consommée » dans l'inventaire partagé | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Jeton »<br>(objectif réussi) | Produire « Jeton »<br>(objectif échoué) |

| Mauvaise_réponse_VC | | | |
|---|---|---|---|
| Rôle prototypique | Constructeur | Bénéficiaire | Adversaire |
| Description | Joueur construisant la Formule | Joueur ne faisant que valider | Joueur adverse obtenant un Jeton à cause de l'échec du Constructeur. |
| Cardinalité | 1 | 1 | 1 |
| Contraintes | Ø | Ø | Ø |
| Equipes | A | A | B |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø | Ø |
| Produit | « Formule consommée » dans l'inventaire partagé | | « Jeton » |
| Requiert | Ø | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø | Ø |
| Objectifs | Produire « Jeton »<br>(objectif échoué) | Produire « Jeton »<br>(objectif échoué) | Ø |

| Mauvaise_réponse CCC | | |
|---|---|---|
| Rôle prototypique | Constructeur | Adversaire |
| Description | Joueur construisant la Formule | Joueur adverse obtenant un Jeton à cause de l'échec du Constructeur. |
| Cardinalité | 2 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Formule consommée » dans l'inventaire partagé | « Jeton » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Jeton » (objectif échoué) | Ø |

| Mauvaise_réponse Ind | | |
|---|---|---|
| Rôle prototypique | Constructeur | Adversaire |
| Description | Joueur construisant la Formule | Joueur adverse obtenant un Jeton à cause de l'échec du Constructeur. |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø |
| Produit | « Formule consommée » dans l'inventaire partagé | « Jeton » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø |
| Objectifs | Produire « Jeton » (objectif échoué) | Ø |

| Mauvaise_réponse_CI | | | |
|---|---|---|---|
| Rôle prototypique | Constructeur | Lésé | Adversaire |
| Description | Joueur construisant la Formule | Joueur n'ayant pu valider en dépit de sa participation | Joueur adverse obtenant un Jeton à cause de l'échec du Constructeur. |
| Cardinalité | 1 | 1 | 1 |
| Contraintes | Ø | Ø | Ø |
| Equipes | A | A | B |
| Consomme | « Formule » dans l'inventaire partagé de l'équipe A | Ø | Ø |
| Produit | « Formule consommée » dans l'inventaire partagé | Ø | « Jeton » |
| Requiert | Ø | Ø | Ø |
| Connaissances Ciblées | « Logique propositionnelle» | Ø | Ø |
| Objectifs | Produire « Jeton » (objectif échoué) | Connaissances Ciblées « Logique propositionnelle» (objectif échoué) | Ø |

| Créer Défense | | |
|---|---|---|
| Rôle prototypique | Défense | Attaquant |
| Description | Joueur dépensant un Jeton pour créer un Défenseur | Joueurs adverses |
| Cardinalité | 1 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Jeton » | Ø |
| Produit | « Défenseur » | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Défenseur » (objectif réussi) | Défense ne doit pas produire de « Défenseur ». (objectif échoué) |

| Créer Attaque | | |
|---|---|---|
| Rôle prototypique | Attaque | Défenseur |
| Description | Joueur dépensant un Jeton pour créer un Attaquant | Joueurs adverses |
| Cardinalité | 1 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Jeton » | Ø |
| Produit | « Attaquant » | Ø |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Attaquant » (objectif réussi) | Attaque ne doit pas produire de « Attaquant ». (objectif échoué) |

| Perdre Vie | | |
|---|---|---|
| Rôle prototypique | Attaque | Défenseur |
| Description | Joueur utilisant son attaquant pour détruire les points de vies | Joueurs adverses |
| Cardinalité | 1 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Attaquant » | « Vie » dans l'inventaire partagé de l'équipe B |
| Produit | Ø | « Mort » dans l'inventaire partagé de l'équipe B |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Défenseur doit consommer « Vie » (objectif réussi) | Ne pas consommer « Vie » (objectif échoué) |

| Annuler Vie | | |
|---|---|---|
| Rôle prototypique | Equipe 1 | Equipe 2 |
| Description | Traitement : nous enlevons une Vie à chaque équipe | Traitement : nous enlevons une Vie à chaque équipe |
| Cardinalité | 2 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Vie » et « Formule consommée » * 8 dans l'inventaire partagé de l'équipe A | « Vie » dans l'inventaire partagé de l'équipe B |
| Produit | « Mort », » Formule consommée » * 8 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | « Mort » et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Ø | Ø |

| Annuler Jeton | | |
|---|---|---|
| Rôle prototypique | Equipe 1 | Equipe 2 |
| Description | Traitement : nous enlève un Jeton à chaque équipe | Traitement : nous enlevons un Jeton à chaque équipe |
| Cardinalité | 1 | 1 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Jeton » dans l'inventaire personnel, « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | « Token » dans l'inventaire personnel, « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Produit | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Ø | Ø |

| Victoire Vie | | |
|---|---|---|
| Rôle prototypique | Vainqueur | Perdant |
| Description | Équipe encore en vie | Équipe n'ayant plus de vies |
| Cardinalité | 2 | 2 |
| Contraintes | Ø | Ø |
| Equipes | A | B |
| Consomme | « Vie » dans l'inventaire partagé de l'équipe A | « Mort » * 3 dans l'inventaire partagé de l'équipe B |
| Produit | « Victoire » | « Défaite » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Victoire » (objectif réussi) | Produire « Victoire » (objectif échoué) |

| Victoire Jeton | | | |
|---|---|---|---|
| Rôle prototypique | Vainqueur | Vainqueur_Bis | Perdant |
| Description | Joueur ayant encore un Jeton | Coéquipier du vainqueur | Équipe n'ayant plus de Jeton |
| Cardinalité | 1 | 1 | 2 |
| Contraintes | Ø | Ø | « Jeton » = 0 |
| Equipes | A | A | B |
| Consomme | « Jeton » dans l'inventaire personnel, « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | Ø | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Produit | « Victoire » | « Victoire » | « Défaite » |
| Requiert | Ø | Ø | Ø |
| Connaissances Ciblées | Ø | Ø | Ø |
| Objectifs | Produire « Victoire » (objectif réussi) | Produire « Victoire » (objectif réussi) | Produire « Victoire » (objectif échoué) |

| Egalité | | |
|---|---|---|
| Rôle prototypique | Equipe A | Equipe B |
| Description | Egalité | Egalité |
| Cardinalité | 2 | 2 |
| Contraintes | « Jeton » = 0 | « Jeton » = 0 |
| Equipes | A | B |
| Consomme | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe A | « Mort » * 3 et « Signal d'Annulation » dans l'inventaire partagé de l'équipe B |
| Produit | « Egalité » | « Egalité » |
| Requiert | Ø | Ø |
| Connaissances Ciblées | Ø | Ø |
| Objectifs | Produire « Victoire »<br>(objectif échoué) | Produire « Victoire »<br>(objectif échoué) |

# X.3. Propriétés détectées au sein des activités du TD2

| | Ind | CCC | VC | CI | CE |
|---|---|---|---|---|---|
| Bonne_réponse CCC | Non | Oui | Oui | Non | Non |
| Bonne_réponse VC | Non | Non | Oui | Non | Non |
| Bonne_réponse Ind | Oui | Non | Non | Non | Non |
| Bonne_réponse CI | Non | Non | Non | Oui | Non |
| Mauvaise_réponse CCC | Non | Oui | Oui | Non | Non |
| Mauvaise_réponse VC | Non | Non | Oui | Non | Non |
| Mauvaise_réponse Ind | Oui | Non | Non | Non | Non |
| Mauvaise_réponse CI | Non | Non | Non | Oui | Non |
| Fin | Non | Non | Non | Non | Non |
| Fin Formule | Non | Non | Non | Non | Non |

# X.4. Propriétés détectées au sein des activités du TD4

| | Ind | CCC | VC | CI | CE |
|---|---|---|---|---|---|
| Bonne_réponse CCC | Non | Oui | Oui | Non | Non |
| Bonne_réponse VC | Non | Non | Oui | Non | Non |
| Bonne_réponse Ind | Oui | Non | Non | Non | Non |
| Bonne_réponse CI | Non | Non | Non | Oui | Non |
| Mauvaise_réponse CCC | Non | Oui | Oui | Non | Oui |
| Mauvaise_réponse VC | Non | Non | Oui | Non | Oui |
| Mauvaise_réponse Ind | Oui | Non | Non | Non | Oui |
| Mauvaise_réponse CI | Non | Non | Non | Oui | Oui |
| Créer Attaque | Non | Non | Non | Non | Oui |
| Créer Défense | Non | Non | Non | Non | Oui |
| Perdre Vie | Non | Non | Non | Non | Oui |
| Annuler Vie | Non | Non | Non | Non | Non |
| Annuler Jeton | Non | Non | Non | Non | Non |
| Victoire Vie | Non | Non | Oui | Non | Oui |
| Victoire Jeton | Non | Non | Oui | Non | Oui |
| Egalité | Non | Non | Non | Non | Non |

# X.5. Spécification pour agréger les actions en Activités.

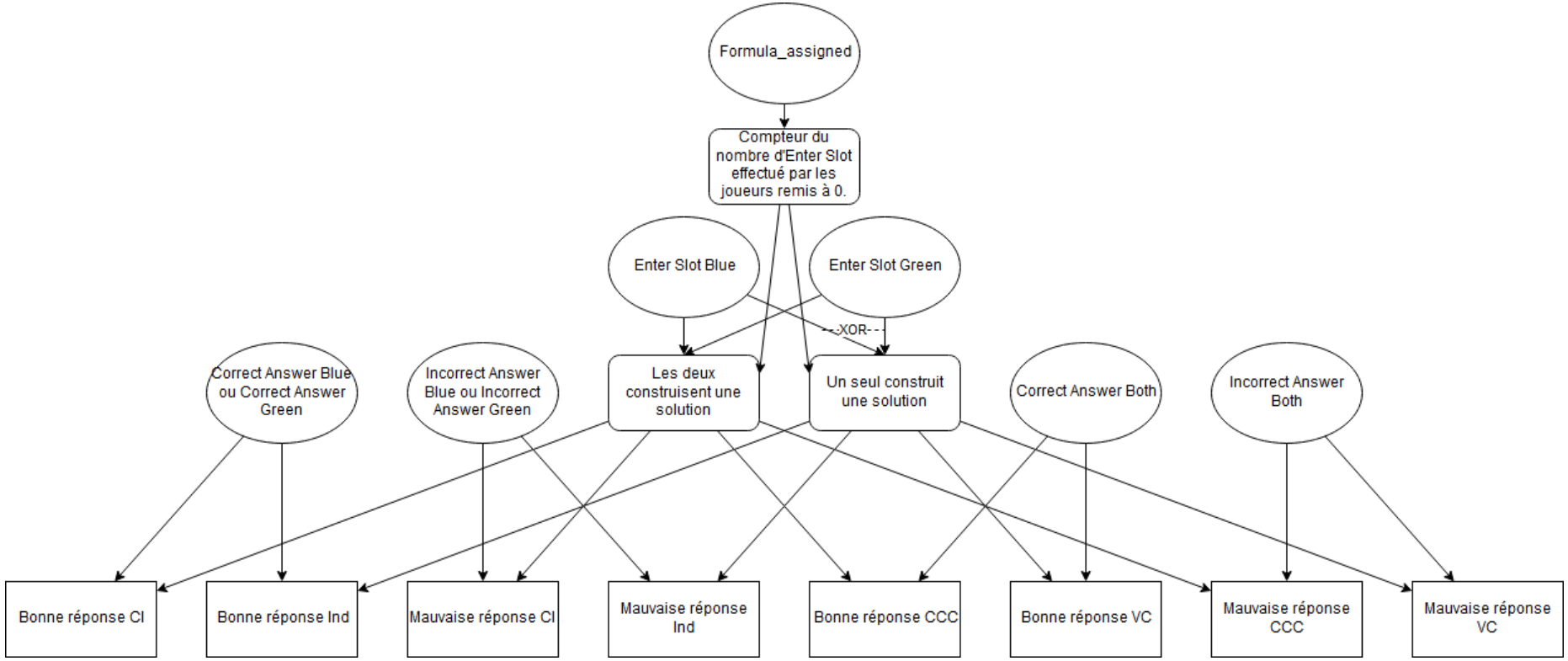


*Figure 36 : Agrégations des actions de validation (TD2 et TD4)*

# X.6. Modèle pour les Traces