Séquencement d'actions en environnement virtuel collaboratif

Guillaume Claude

Thèse

UNIVERSITE
BRETAGNE
LOIRE

THESE INSA Rennes
sous le sceau de l'Université Bretagne Loire
pour obtenir le titre de
DOCTEUR DE L'INSA RENNES
Spécialité : Informatique

présentée par

Guillaume CLAUDE

ECOLE DOCTORALE : *MATISSE*
LABORATOIRE : *IRISA*

Séquencement d'actions en environnement virtuel collaboratif

Thèse soutenue le 12.07.2016
devant le jury composé de :

Pierre Jannin
Directeur de recherche INSERM,LTSI Rennes / Président
Indira Thouvenin
Professeure, UTC Compiègne/ Rapporteure
Pierre Chevaillier
Professeur, ENIB Brest / Rapporteur
Pascal Guitton
Professeur Université de Bordeaux 1 / Examinateur
Valérie Gouranton
Maitre de Conférences, INSA de Rennes / Co-encadrante de thèse
Bruno Arnaldi
Professeur, INSA de Rennes / Directeur de thèse

Séquencement d’actions en environnement virtuel collaboratif

Guillaume CLAUDE

En partenariat avec

"I saw my earlier selves as different people, acquaintances I had outgrown. I wondered how I could ever have been some of them."

- *Corwin*, in **The Courts of Chaos** (Roger Zelazny, 1978)

Remerciements

Merci tout d'abord aux membres du jury de cette thèse : Indira Thouvenin et Pierre Chevaillier pour leurs rôles de rapporteurs et leurs retours plus que pertinents sur ce manuscrit ainsi que Pascal Guitton et Pierre Jannin pour leurs rôles d'examinateurs. Merci à tous les quatre pour les échanges et discussions que nous avons pu avoir. Je remercie énormément Bruno Arnaldi et Valérie Gouranton pour leur encadrement, pas toujours facile, durant ces trois ans. Je vais fait mon possible pour faire les choses dans le bon sens. Je remercie également l'ensemble des membres du projet S3PM de m'avoir accueilli et de m'accueillir de nouveau au sein de SUNSET. C'est un projet passionnant autant scientifiquement qu' humainement.

Merci à Florian, Thomas "papa", Rozenn, Julian, Cedric et Thomas L. pour leur travail sur FIVE, SEVEN et les démos des différentes publications liées de près ou de loin à ce manuscrit, mais également pour les moments de tous les jours. Mention spéciale pour Quentin "QPE" petit.

Merci à Fraçois pour ces trois ans de bureau. Merci aussi à Tristan, Kevin, Andéol, Gwendal, Anne-Solène et Benoit et Ludovic ainsi qu'à tous les membres d'Hybrid et de Mimetic qui font la bonne humeur de tous les jours dans ce couloir.

Merci à nos assistantes : Nathalie, Marie-Françoise et Aurore parce que personne n'arriverait à rien sans elles.

Merci à Cédric, Antony et Bertrand qui m'ont sorti de mon canapé et qui m'ont permis de décompresser quand c'était (souvent) nécessaire. Merci aux potes de Wong Fei et surtout à l'équipe de compétition pour la bonne humeur et la motivation.

Merci à mes parents de m'avoir laissé faire les études que je voulais comme je voulais, même si ça n'a pas dû toujours être facile.

Merci à Marie-Claude pour les coups de main sur ces trois ans, notamment pour le pot, mais pas que.

Et surtout, merci à ma petite femme, Déborah, pour son soutient et qui m'a laissé quitter un CDI d'ingénieur avec des astreintes pour une paye de doctorant, qui a accepté les rédactions de papiers, les horaires, les pas-weekends, les pas-vacances et pour toute l'aide qu'elle m'a apportée sur la rédaction de ce manuscrit.

Merci à toutes les personnes que j'ai pu oublier, car il y en a surement plein.

Enfin, une pensée pour les CTM de Saint-Malo : Merci, nous sommes prévenus !

Table des matières

Introduction

La réalité virtuelle est actuellement en plein essor, portée par le battage médiatique de l'arrivée, cette année (2016), des différents *Head Mounted Display* HMD, "*Casques de réalité virtuelle*", pour le grand public. Pourtant la réalité virtuelle n'est pas nouvelle (à l'échelle de l'informatique). En 1965 Sutherland [Sutherland, 1965] imagine une pièce dans laquelle l'existence de la matière est contrôlée par ordinateur. Peu après (1968) le premier HMD voit le jour (c.f. Figure 1), toujours dans les travaux de Sutherland [Sutherland, 1968].

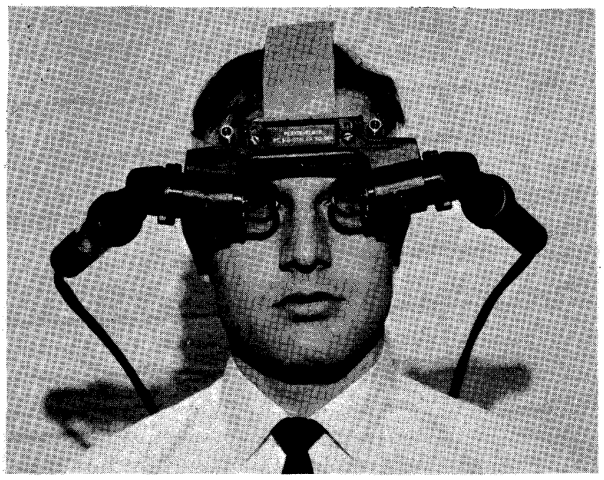


Figure 1 – Le *Head Mounted Display* de Sutherland, 1968 (origine : [Sutherland, 1968]

Depuis, sans atteindre la vision de Sutherland, la réalité virtuelle a évolué et permet maintenant la mise en place d'environnements virtuels pour diverses utilisations en loisir comme dans le monde professionnel. Une de ces utilisations est de simuler des situations pour permettre à une ou plusieurs personnes d'acquérir ou de renforcer des compétences. Réaliser ces environnements virtuels pour la formation demande actuellement beaucoup de travail et des compétences avancées en développement logiciel. Une des difficultés majeures est de définir ce que les personnages (virtuels ou réels) peuvent ou doivent faire dans la simulation. Pour cela, les environnements virtuels reposent sur des modèles de scénarios.

Les travaux présentés dans ce manuscrit sont inscrits dans le projet de recherche S3PM, qui cherche à automatiser la spécification de scénarios de formations aux procédures de neurochirurgie à partir de l'observation de cas réels. Nos travaux visent à proposer un modèle

Figure 2 – La plateforme de réalité virtuelle Immersia (Rennes, France)

de scénarios permettant une spécification et une exécution de scénarios pour environnements virtuels, compatibles avec les besoins du projet, mais permettant également une utilisation plus classique par un développeur spécialisé. Pour cela, nous nous intéressons à deux problèmes distincts : la description des actions possibles dans l'environnement et la distribution de ces actions entre les différents personnages (réels ou virtuels).

Ce manuscrit est découpé en 7 chapitres :

- Le Chapitre 1 présente le projet S3PM.
- Le Chapitre 2 propose une étude des travaux existants en modélisation des environnements virtuels, en modélisation des scénarios et en distribution des actions.
- Le Chapitre 3 présente en détail #SEVEN, notre modèle générique de scénarios.
- Le Chapitre 4 détaille l'utilisation de #SEVEN pour spécifier puis exécuter des scénarios en environnements virtuels
- Le Chapitre 5 propose un nouveau modèle pour gérer la distribution des actions entre les personnages.
- Le Chapitre 6 détaille l'utilisation de nos modèles dans le cadre du projet S3PM pour réaliser un environnement virtuel pour la formation à une procédure de neurochirurgie.

Ces chapitres sont suivis d'une conclusion et de propositions de perspectives de recherches.

Contexte : Le projet S3PM

1

S3PM (Synthesis and Simulation of Surgical Process Models) est un projet de CominLabs (Comunication and Information Sciences Laboratories), une initiative financée par le ministère français de la Recherche dans le cadre du programme "*Laboratoires d'Excellence*", elle-même partie du programme "*Investissements d'Avenir*". l'objectif de S3PM est de mettre en place des technologies pour permettre la réalisation de système de réalité virtuelle de formation à des procédures chirurgicales à partir d'observations de cas réels.

Le projet réunit les membres de plusieurs équipes de recherches issues de domaines et de laboratoires différents. MediCIS est une équipe de l'INSERM (Institut Nationnal de la Santé et de la Recherche Médicale). Elle s'intéresse aux systèmes d'assistance à la prise de décision, à la formation et à l'évaluation des compétences en chirurgie. HYCOMES est une équipe d'Inria (Institut National de la Recherche en Informatique et en Automatique) et de l'IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires). Son expertise touche notamment la modélisation de systèmes hybrides et le process-mining. Hybrid qui a encadré les travaux de ce manuscrit, est également une équipe Inria/IRISA. Elle s'intéresse aux technologies liées à la Réalité virtuelle et à l'interaction des utilisateurs avec des environnements virtuels. Enfin, ces travaux sont réalisés avec la participation du département de neurochirurgie du Centre Hospitalier Universitaire de Rennes.

S3PM ne s'intéresse pas aux gestes techniques, mais à la procédure de l'opération et à la collaboration entre les différents membres de l'équipe médicale. Ce choix est guidé par le fait que seulement 25% de la qualité d'une opération chirurgicale dépend de la technique. Les 75% restants sont liés à des compétences non techniques ou à de la connaissance [Hall et al., 2003].

L'objectif principal de S3PM est de fournir un modèle de la procédure (c.-à-d. une représentation des déroulements possibles) exécutable par un Système de réalité virtuelle. Cette représentation est obtenue à partir de données issues de l'observation de plusieurs cas réels d'opération chirurgicale. La Figure 4.2 donne un aperçu des étapes de génération et d'utilisation d'un modèle. L'observation de cas réels (1) permet d'obtenir des exemples concrets d'exécution de procédures chirurgicales. Ces données sont utilisées pour générer un modèle général de la procédure (2) décrivant l'ensemble des scénarios possibles. Ce modèle est intégré dans une application de réalité virtuelle (3) pour la formation collaborative du personnel médical. Dans ce Chapitre, nous détaillons les deux premières étapes du processus. La troisième partie est liée

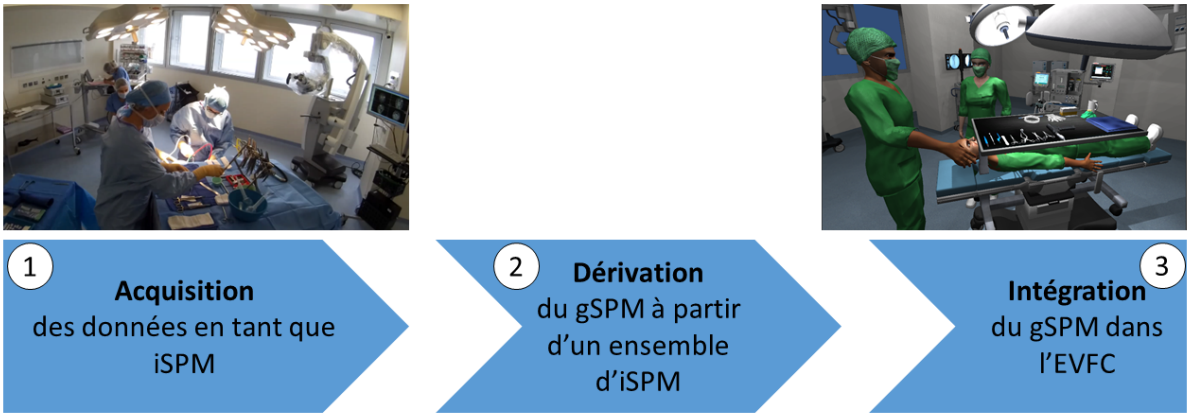


Figure 1.1 – Workflow général du projet S3PM.

aux contributions de ce manuscrit et sera donc traitée dans les chapitres suivants.

1 Acquisition des données

La première étape consiste à obtenir des données de cas réels. L'objectif est de recueillir plusieurs instances d'une même procédure (ou partie de procédure) chirurgicale. Par exemple une opération de la cataracte ou une craniotomie (étape d'ouverture du crane en neurochirurgie). Pour ce faire, l'opération est filmée suivant plusieurs points de vue. Les vidéos sont ensuite utilisées en parallèle de SurgeTrack [Garraud et al., 2014], une suite logicielle de saisie de données chirurgicales (c.f. Section 1.2) qui repose sur le modèle ontologique OntoSPM [Gibaud et al., 2014] (c.f. Section 1.1).

1.1 OntoSPM

OntoSPM [Gibaud et al., 2014] est construit sur l'ontologie fondatrice BFO (Basic Formal Ontology) [Smith and Grenon, 2002]. OntoSPM fournit le vocabulaire permettant la description détaillée de procédures chirurgicales. Il permet par exemple de décrire les étapes, les actions réalisées, les acteurs ou les instruments utilisés. Le modèle reprend des termes d'ontologies existantes comme IAO (Information Artifact ontology [Ceusters, 2012]), PATO (Phenotype And Trait Ontology [Xiang et al., 2011]) ou FMA (Foundational Model of Anatomy [Rosse et al., 2003]). Il contient actuellement 262 classes et 78 propriétés d'objets. La Figure 1.2 montre un extrait d'OntoSPM : L'action "*Trépaner le crane*" et une spécialisation de l'action "*Couper une partie du corps*", qui ne s'applique qu'à "*un crane*" et qui n'est réalisable que par "*un chirurgien*" à l'aide d'un "*foret haute vitesse*". OntoSPM est open source et vise à être utilisé et étendu. Une extension de OntoSPM spécifique pour S3PM a été définie. Elle contient actuellement 101 classes qui s'intéressent spécifiquement à la phase de craniotomie.

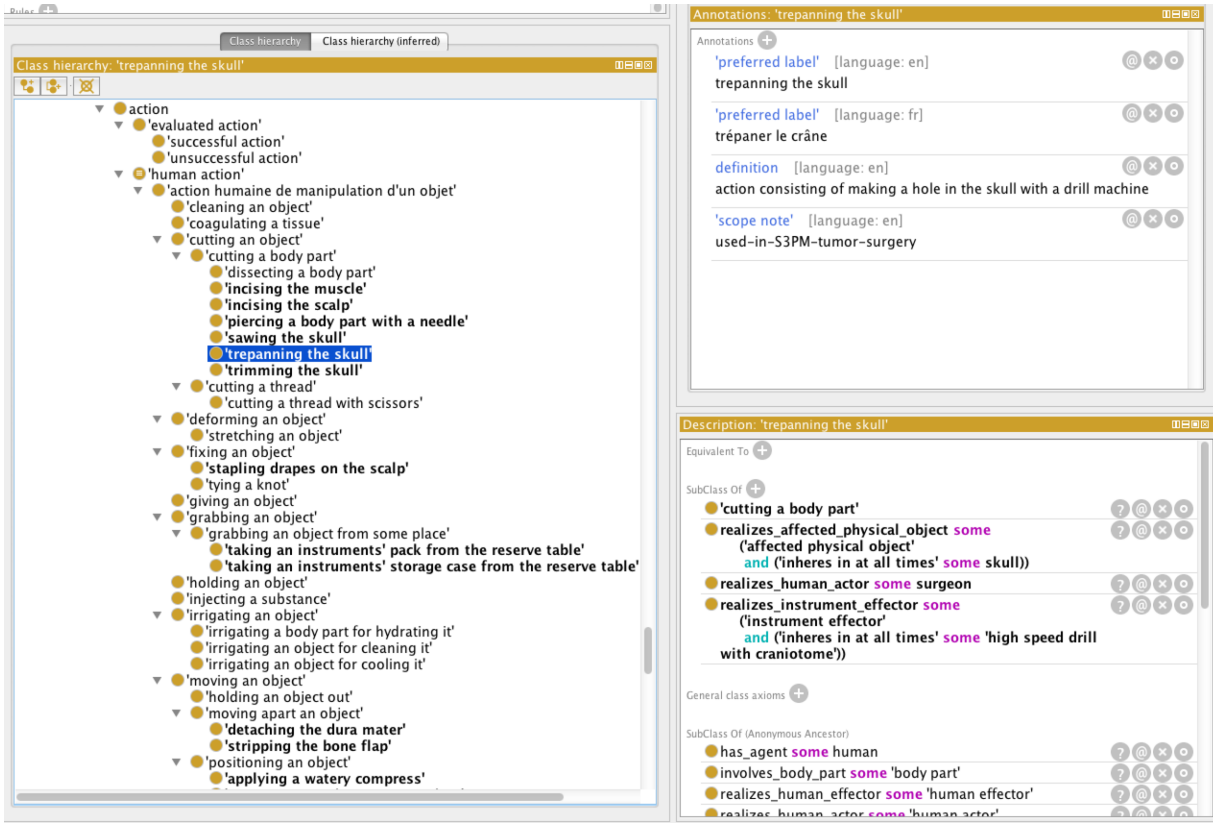


Figure 1.2 – OntoSPM permet de décrire de manière détaillée des procédures chirurgicales.

1.2 SurgeTrack

SurgeTrack [Garraud et al., 2014] est une suite logicielle qui repose sur OntoSPM (voir Section 1.1) pour la représentation des données liées aux procédures de chirurgie. Elle fournit notamment des outils pour noter les observations d'une procédure. Il s'agit d'une description des actions des différents acteurs et des évènements ayant lieu lors de la procédure, positionnés dans le temps les uns par rapport aux autres. La Figure 1.3 présente l'interface principale du logiciel. La fenêtre de gauche contient la vidéo capturée lors de la procédure. La fenêtre de droite permet la visualisation et la saisie séquentielle des actions et autres évènements ayant lieu lors de la procédure.

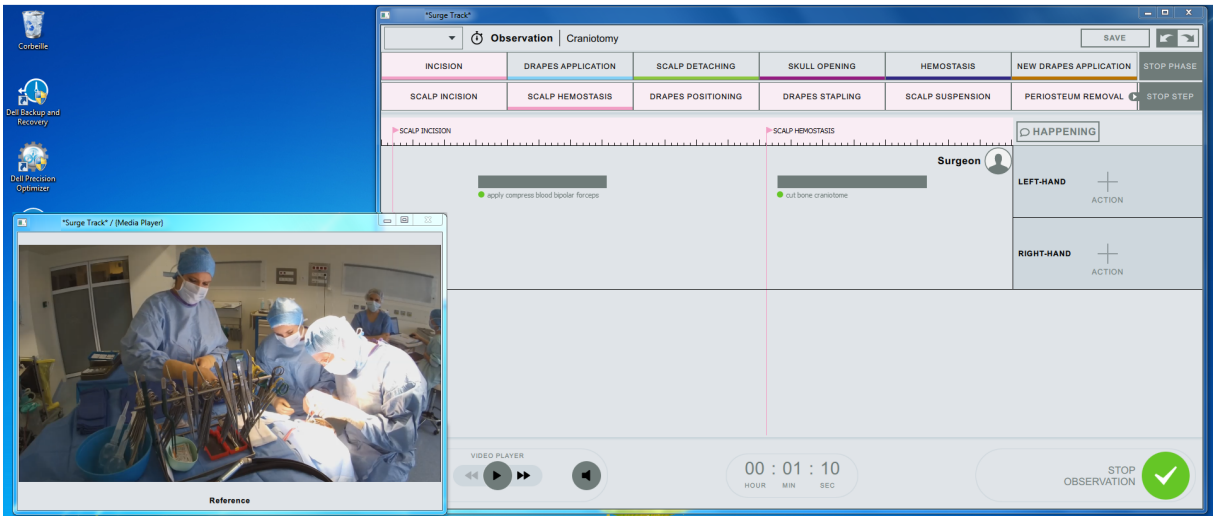


Figure 1.3 – SurgeTrack permet de décrire une procédure en suivant les actions des différents acteurs.

2 Génération du modèle de procédure générique

Les données des observations sont ensuite utilisées, en conjonction avec l'ontologie, par un système qui synthétise un réseau Test and Flip [Caillaud, 2013] (c.f. Section 2.1). Ce réseau propose un modèle générique de la procédure en généralisant les observations (c.f. Section 2.1.1).

2.1 Les Réseaux Test and Flip

Un réseau Test and Flip est un réseau place-transition binaire (possédant 0 ou 1 jeton). Chaque place peut-être reliée à des transitions par des arcs. Ces arcs portent un opérateur. Cet opérateur définit deux choses : une condition sur la valeur de la place pour que la transition soit déclenchable et le nouvel état de la place si la transition est déclenchée. Le tableau 1.1 résume ces opérateurs. De par sa nature binaire, un réseau Test and Flip ne peut pas tenir un décompte du nombre d'occurrence d'un évènement.

2.1.1 Synthèse du modèle de procédure

La synthèse de réseau [Caillaud, 2013] prend en compte la concurrence, les conflits et les relations de causalités entre les évènements observés. La synthèse de réseau Test and Flip utilisée dans le projet S3PM n'est pas statistique. Elle repose sur la résolution d'un système d'équations linéaires. Ainsi, elle permet d'obtenir un réseau qui généralise la procédure et propose des exécutions qui n'ont pas forcément été observées. La Figure 1.4 donne un exemple théorique de synthèse. Les enregistrements (1) sont utilisés pour générer un réseau Test and Flip (2).

Opérateur	Condition	Effet
[0]	Test à 0	aucun
[1]	Test à 1	aucun
[-]	Test à 1	mise à 0
[+]	Test à 0	mise à 1
[¬]	aucune	inverse
[⊥]	aucune	aucun

Table 1.1 – Opérateurs des réseaux Test and Flip. La condition et l'effet s'appliquent à l'état de la place associée

.

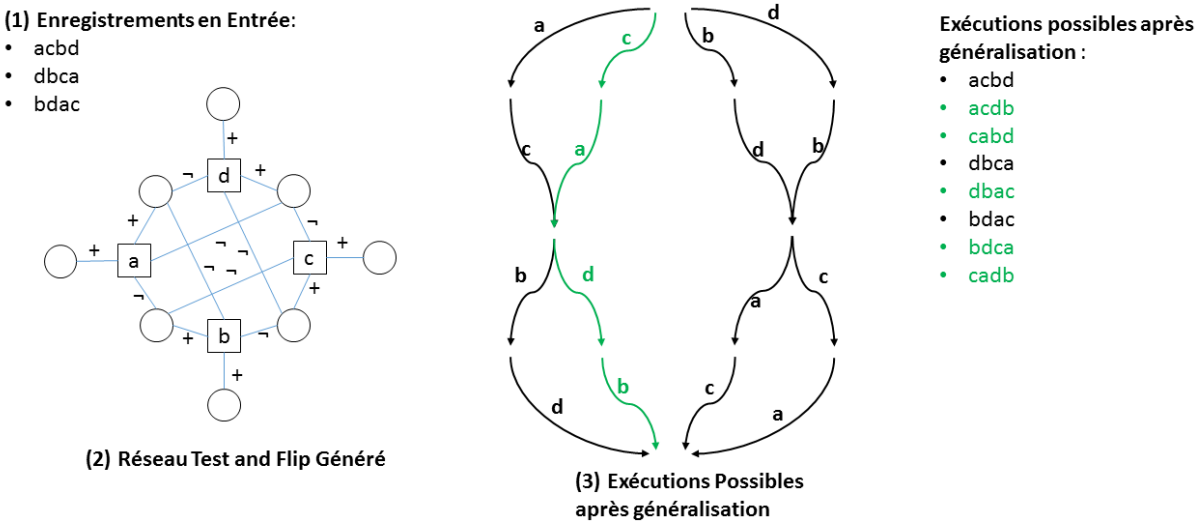


Figure 1.4 – Exemple de synthèse de réseau à partir d'enregistrements.

Ce dernier propose une généralisation des enregistrements et propose de nouvelles exécutions possibles qui n'ont pas été observées en vert (3).

3 Environnements virtuels pour la formation collaborative

Le réseau Test and Flip généré fournit une description des agencements possibles des actions lors d'une procédure chirurgicale. Cependant, ce dernier ne propose actuellement pas les propriétés nécessaires à une intégration dans un système de réalité virtuelle. Il lui faut un moteur d'exécution compatible avec un environnement virtuel collaboratif. De plus, il faut faire correspondre les actions observées à des entités de l'environnement virtuel, et prendre en compte les actions des différents personnages par rapport à leur implication prévue dans la réalité. Enfin, il faut également contrôler l'état de l'environnement virtuel pour qu'il réagisse de

la même manière que la salle d'opération réelle aux actions des acteurs et permette de recréer des situations de formation.

Les travaux présentés dans ce manuscrit cherchent à offrir des modèles permettant de décrire des agencements plus ou moins complexes d'actions d'acteurs et d'autres évènements pouvant avoir lieu en environnement virtuel. Deux aspects sont étudiés : la description de la procédure et le travail en équipe. Le premier s'intéresse à la description de l'agencement, temporel et causal, des actions indépendamment de la personne qui les réalise. Ce modèle doit pouvoir être utilisé par lui-même ou en combinaison avec les outils de génération et de descriptions de procédure du projet S3PM que sont SurgeTrack et la synthèse de réseaux Test and Flip. Le travail en équipe passe, entre autres, par une distribution des actions entre les intervenants. Cette distribution suit des règles définies et connues par les différents membres de l'équipe. Nous cherchons donc un modèle permettant de décrire la distribution des actions à partir de l'organisation de l'équipe qui puisse être utilisé conjointement avec le modèle de description de procédure.

État de l'art

2

Ce chapitre propose une analyse des travaux existants portant sur l'agencement des actions dans les environnements virtuels. La section 1 introduit l'architecture des environnements virtuels. La section 2 s'intéresse spécifiquement aux modèles de scénarios pour environnements virtuels. Enfin, la section 3 se concentre sur les modèles de distribution des actions entre les personnages du monde virtuel.

1 Architecture des environnements virtuels

Dans cette section nous nous intéressons aux principes généraux liés aux environnements virtuels. Dans la section 1.1 nous proposons une introduction à la réalité virtuelle et dans la section 1.2 nous proposons une étude des modèles de spécification de comportements pour les environnements virtuels.

1.1 Introduction à la réalité virtuelle

Il existe différentes définitions de la réalité virtuelle [Steuer, 1992]. Dans ce manuscrit, nous utiliserons celle d'Arnaldi et al.[Arnaldi et al., 2006]:

"*La réalité virtuelle est un domaine scientifique et technique exploitant l'informatique et des interfaces comportementales en vue de simuler dans un monde virtuel le comportement d'entités 3D, qui sont en interaction en temps réel entre elles et avec un ou des utilisateurs en immersion pseudonaturelle par l'intermédiaire de canaux sensorimoteurs.* "

La Figure 2.1 illustre cette définition. Nous distinguons deux grandes familles de travaux liés à la réalité virtuelle. La première (1) s'intéresse à la création et à l'utilisation des interfaces qui permettent à l'utilisateur d'interagir et de percevoir l'environnement virtuel. Il s'agit de matériel comme des contrôleurs spécialisés ou des dispositifs de visualisation. Ils permettent de capter les gestes de l'utilisateur et de les transposer dans l'Environnement 3D. À l'inverse ils servent également à transformer les données de la simulation afin que l'utilisateur puisse les interpréter via ses sens (principalement la vue, l'ouïe ou le toucher mas pas exclusivement). La seconde famille de travaux (2) s'intéresse à l'environnement virtuel et à la définition des comportements des entités qui le composent. Les travaux présentés dans ce mémoire se placent dans cette famille.

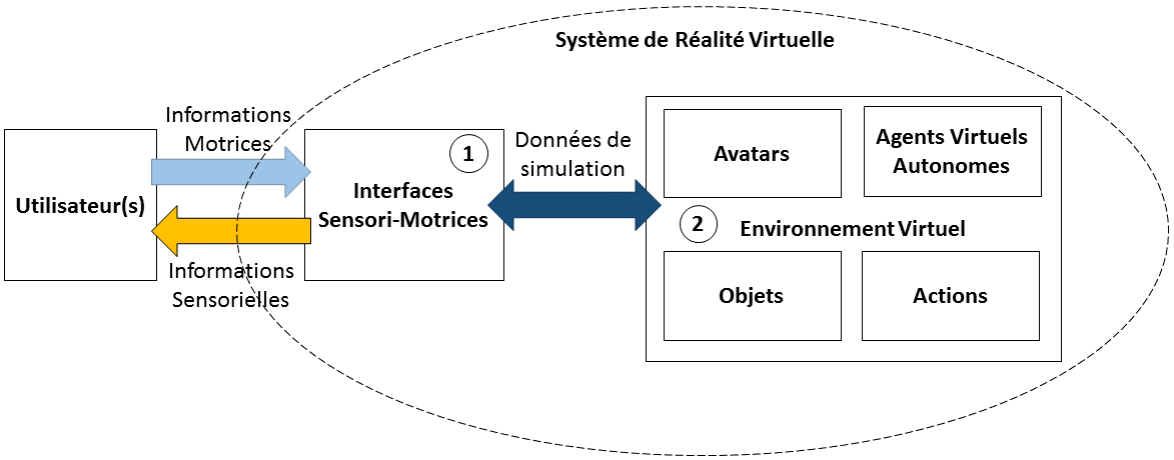


Figure 2.1 – Schéma de la réalité virtuelle

Les **Interfaces sensorimotrices** permettent à l'utilisateur de percevoir ou d'agir sur l'environnement virtuel. Il s'agit de composants matériels ou logiciels dont la fonction est de transmettre les actions physiques de l'utilisateur à l'environnement virtuel et/ou de transmettre à l'acteur des informations sur l'état de l'environnement virtuel. Les interfaces sensorimotrices participent grandement à l'immersion de l'utilisateur dans l'environnement virtuel. Pimentel et Teixeira [Pimentel and Teixeira, 1993] définissent l'immersion comme "*l'état (perceptif, mental et émotionnel) d'un sujet lorsque un ou plusieurs de ses sens sont isolés du monde extérieur et sont alimentés uniquement par des informations issues de l'ordinateur*". Les interfaces sensorielles offrent une immersion plus ou moins forte en fonction de leur capacité à isoler l'utilisateur du monde réel. Par exemple (Figure 2.2) la vue par un casque immersif (a) ou par l'affichage d'une salle immersive (b). De même les interfaces motrices vont offrir un contrôle plus ou moins naturel. Par exemple les gestes peuvent être contrôlés par une manette de jeu (a) ou par un dispositif de capture de mouvements (b).

Selon Moreau [Moreau, 2006], **l environnement virtuel** est une représentation d'une partie de la réalité ou d'un environnement imaginaire. La modélisation d'un environnement virtuel repose sur plusieurs familles de modèles. Nous pouvons notamment citer les modèles de représentation 3D et les modèles de comportements. Les modèles de représentation 3D permettent de décrire la forme des objets composant l'environnement virtuel. Les modèles de comportements permettent de décrire ce qu'il peut se passer dans l'Environnement. Ces comportements sont attachés aux **objets** qui composent l'Environnement. L'état d'un environnement virtuel est l'ensemble des états de ses objets. Cet état est défini par un ensemble de variables et peut être modifié par le biais d'**actions**.

D'après Smith et al. [Smith et al., 1999] un système de réalité virtuelle est un système hybride : il est composé d'une partie continue et d'une partie discrète. La partie continue est définie par les composants physiques des interfaces sensorimotrices. La simulation est de nature discrète : un

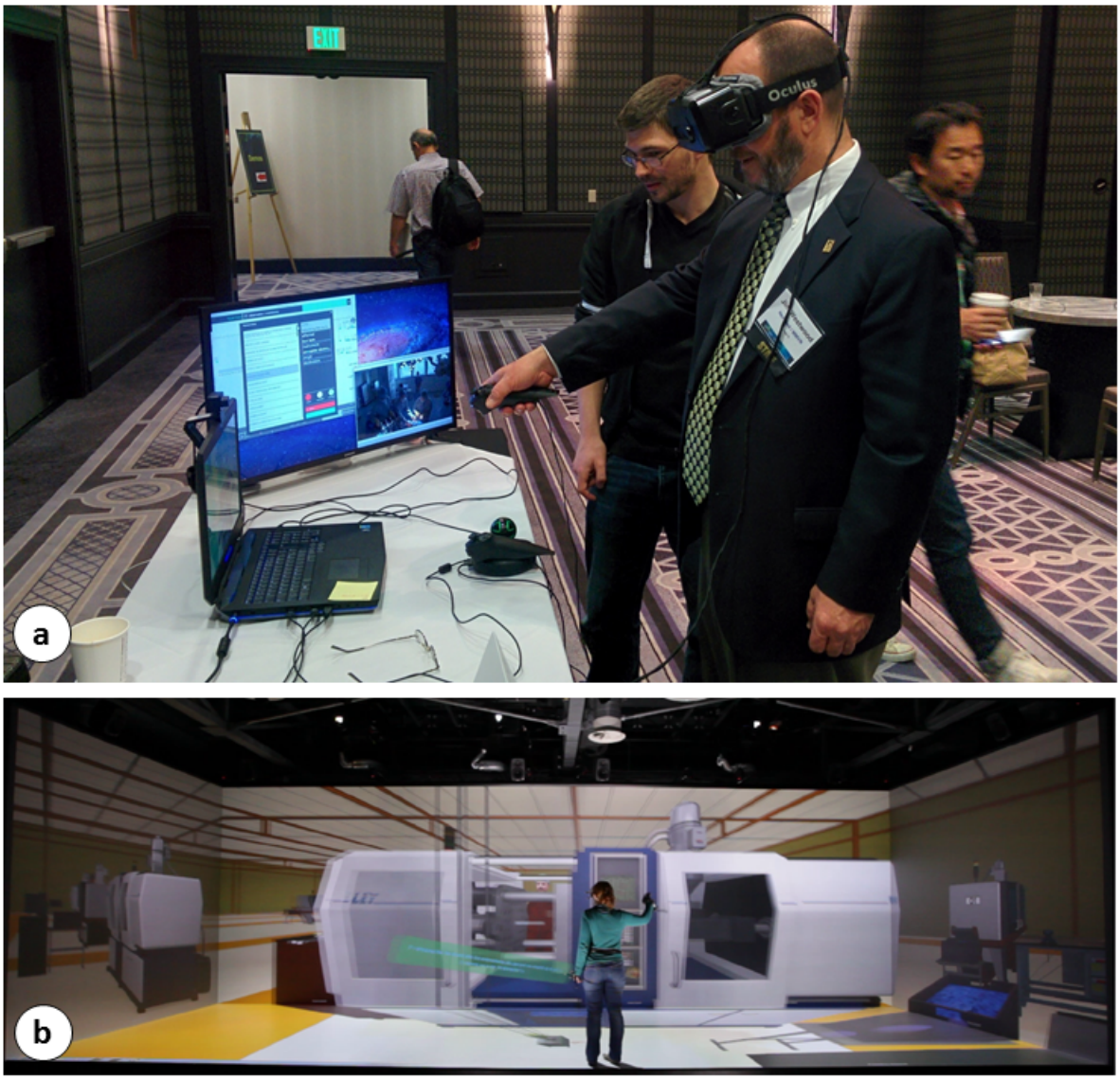


Figure 2.2 – Exemples d'interfaces sensorielles et motrices.

nouvel état de l'environnement virtuel est calculé à chaque pas. Une **action** est une succession de changements de l'état de l'environnement virtuel et est déclenchée par un composant du système. Nous appelons **évènement** un changement de l'état de l'environnement virtuel dont l'occurrence va impliquer des réactions de la part de certains composants de l'environnement virtuel (dits alors *réactifs*). Il peut s'agir par exemple du début d'une action, ou du fait qu'un objet atteigne une position précise dans l'environnement virtuel. Un objet peut-être réactif. Par exemple, si un feu se déclenche, le détecteur de fumée va faire sonner l'alarme. Une action

peut durer tout au long de la simulation ou se terminer. Puisque son but est de modifier l'état de l'environnement virtuel, une action s'applique à au moins un objet. Par exemple, l'action "*Régler la Luminosité*" implique un personnage, un potentiomètre et une lampe. Le personnage déclenche l'action en prenant le potentiomètre. Tout au long de l'action, il peut tourner le potentiomètre autour de son axe, ce qui modifie son état. À chaque fois que l'état du potentiomètre change, la luminosité de la lampe est adaptée. L'acteur met fin à l'action en relâchant le potentiomètre.

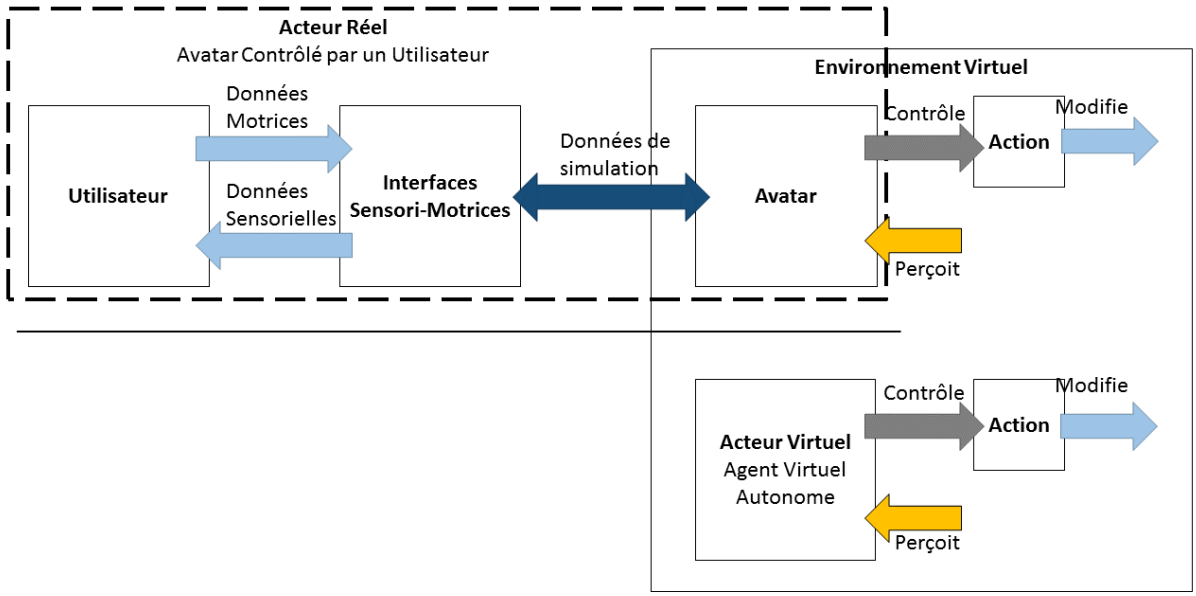


Figure 2.3 – Les Acteurs d'un environnement virtuel peuvent être de deux natures : Purement logiciels ou contrôlés par un utilisateur.

Les acteurs agissent dans l'environnement virtuel. Ils vont par exemple conduire un véhicule, communiquer entre eux, réparer une machine ou soigner un patient. Un acteur peut être un *agent virtuel autonome* ou un *utilisateur et son avatar* (voir Figure 2.3). Un **avatar** : une représentation qui permet à l'utilisateur de percevoir, d'être perçu et d'agir dans l'environnement virtuel [Thalmann, 2000]. L'avatar peut être plus ou moins détaillé. Il peut par exemple s'agir d'un personnage 3D photoréaliste. Il peut également être représenté par des métaphores ou encore simplifié au maximum au point qu'il n'ait plus de représentation graphique. Un agent virtuel autonome possède également une représentation dans l'environnement virtuel. Elle lui offre les mêmes possibilités de perception et d'action que l'avatar de l'utilisateur. Cependant, cette représentation est directement liée à un comportement. Il existe différentes manières de définir ces comportements (p. ex. Automates, intelligence artificielle) [Thalmann, 2000]. Un acteur est une entité de l'environnement virtuel qui possède un état défini par une ou plusieurs variables et qui a un comportement propre. Ce comportement implique l'exécution d'actions

dans le but de modifier l'état de l'environnement virtuel. Un *environnement virtuel collaboratif* comprend plusieurs acteurs. De nombreux travaux ont pour objectif la modélisation des acteurs et leur activité dans l'environnement. Par exemple, certains s'intéressent à modéliser des acteurs qui soient des formateurs ou des collaborateurs [Johnson and Rickel, 1997] [Luna et al., 2013]. **La collaboration** nécessite que les acteurs aient des capacités d'interactions leur permettant de coopérer les uns avec les autres. Margery et al. [Margery et al., 1999] proposent une classification des niveaux de collaboration possibles. Nous utiliserons par la suite cette classification pour caractériser les possibilités offertes par les solutions existantes. En voici une version résumée :

- **Niveau 1 - *Perception :*** Les acteurs sont capables de percevoir les autres acteurs.
- **Niveau 2 - *Interaction :*** Les acteurs sont capables d'agir sur la scène
  - **Niveau 2.1 - *Contrainte :*** Les modifications sont contraintes pas la scène. Par exemple déclencher des animations.
  - **Niveau 2.2 - *Libre :*** Les actions ne sont plus contraintes. Par exemple un acteur peut déplacer un objet n'importe où dans la scène.
- **Niveau 3 - *Concurrence :*** les acteurs peuvent agir à plusieurs sur un même objet.
  - **Niveau 3.1 - *Exclusion :*** Les acteurs peuvent agir sur un même objet s'ils modifient des éléments non liés. Par exemple l'un sur la position et l'autre sur la couleur.
  - **Niveau 3.2 - *comanipulation :*** Les acteurs peuvent agir ensemble sur un même élément d'un même objet. Par exemple le déplacement d'une table à deux en prenant chacun un côté.

Par exemple, COVET (Collaborative Virtual Environment for Training) [Hosseini and Georganas, 2001; Oliveira et al., 2000a,b,c] offre la possibilité à plusieurs utilisateurs (un formateur et des apprenants) d'être dans le même environnement. Cependant, seulement un seul d'entre eux peut interagir avec l'environnement. Les autres ne font que regarder et discuter entre eux. De plus, les actions exécutées sont choisies par l'acteur, mais pilotées par le système. Il s'agit donc d'une collaboration de niveau 2.1 "*Interaction Contrainte*". Autre exemple, dans SecuReEVI [Querrec et al., 2003, 2004; Querrec and Chevaillier, 2001; Querrec et al., 2001] les acteurs peuvent réaliser ensemble, des actions collaboratives complexes. L'apprenant doit prendre des décisions dans des situations opérationnelles en tant que chef d'une équipe de pompiers et le formateur peut agir sur l'environnement pour déclencher des incidents. Il offre une collaboration de niveau 3.2 "*comanipulation*".

La suite de cette section s'intéresse particulièrement aux modèles de comportements liés à la spécification de l'environnement virtuel et à son utilisation.

1.2 Modélisation des environnements virtuels

Ce mémoire porte sur l'Agencement des Actions dans les environnements virtuels. Il s'agit de définir quelles actions sont réalisables en fonction de l'état de l'Environnement et de l'avancement de la simulation. Les modèles utilisés pour représenter un environnement virtuel de Formation définissent comment fonctionnent les actions et comment sont impliqués les acteurs et les objets dans leur exécution. Selon Gerbaud [Gerbaud, 2008] "*, les Environnements virtuels de Formation permettent de faire acquérir ou de renforcer les compétences ou le savoir-faire d'apprenants, en vue de leur application dans des situations réelles.*" Les environnements virtuels de formation reposent sur le concept d'*apprendre en faisant* basé sur la théorie constructiviste [Piaget, 1976]. Les possibilités d'actions offertes aux acteurs doivent être adaptées aux compétences à acquérir lors de la session d'entrainement. Par exemple dans MRE [Swartout et al., 2006a], l'utilisateur est un militaire sur le terrain qui doit prendre des décisions en fonction de situations stressantes. Les actions sont disponibles à travers des choix multiples lors de discussion avec des personnages. Les compétences acquises sont clairement de l'ordre de la décision et des relations sociales. Dans CORVETTE [Berthelot et al., 2014] les acteurs collaborent pour réaliser la maintenance d'une machine industrielle. Ils peuvent communiquer entre eux par la parole ou utiliser directement les objets à leur disposition. Il s'agit par exemple de positionner de manière correcte les composants de la machine. Dans ce cas, les compétences acquises sont de l'ordre du geste technique et de l'apprentissage de procédures.

Les "*systèmes de narration interactive*" s'intéressent également à ces mécanismes. Szilas [Szilas, 2003] définit la narration interactive comme une extension de la narration linéaire (p. ex. livres, cinéma). Selon Riedl et Bulitko [Riedl and Bulitko, 2012], "*la narration interactive est une forme d'expérience digitale et interactive dans laquelle des utilisateurs créent ou influence une histoire par leurs actions. L'objectif des systèmes de narration interactive est d'immerger les utilisateurs dans un monde tel qu'ils croient faire partie intégrante du déroulement d'une histoire et que leurs actions peuvent significativement influer la direction et/ou la conclusion de l'histoire*". Dans Madame Bovary [Cavazza et al., 2007], l'utilisateur est intégré dans une histoire en tant qu'un des personnages. Il agit en discutant avec les autres acteurs (virtuels) et ces actions vont influer sur leur état mental et donc sur les décisions qu'ils vont prendre dans la suite de la simulation. Dans TEATRIX [Paiva et al., 2001] les acteurs ont des objectifs qu'ils doivent essayer de remplir. Pour ce faire, ils ont accès à un ensemble d'actions simples comme prendre, poser ou utiliser un objet. La narration interactive n'implique pas forcément l'utilisation de techniques de Réalité virtuelle. Cependant, elle nécessite quand même la modélisation du comportement d'un environnement virtuel par des modèles définissant, entre autres, les actions des acteurs et les comportements des objets. Par exemple IDTension [Szilas, 2003] propose des descriptions textuelles d'un récit dans lequel les actions des acteurs ont une part très importante.

En formation comme en narration interactive, l'**action** semble toujours être un élément de l'environnement virtuel. Comme nous l'avons vu précédemment, l'action modifie l'état des **objets**. Ces deux types d'éléments nous semblent devoir être clairement définis dans un Modèle

d'environnement virtuel. Ils doivent être rendus accessibles de manière explicite aux autres composants de l'environnement virtuel, car ils sont les fondations de toutes simulations.

Certains objets sont réactifs : ils ont un comportement propre qui va à son tour déclencher d'autres actions et modifier de nouveau leur état, voir celui d'autres objets. Willans et al. [Willans and Harrison, 2001b] donnent l'exemple d'une gazinière composée de plusieurs éléments : un bouton d'ouverture du gaz et un interrupteur d'étincelle. Si le gaz circule (c.-à-d. l'état du bouton d'ouverture, est "*ouvert*") et que l'on appuie sur l'interrupteur, alors le feu s'allume. L'état de l'objet "*feu*" dépend de l'état des différents autres objets et actions déclenchée. Cet état est donc défini par certaines séquences d'évènements possibles, conditionnés par le comportement des différents objets intervenants. Il est important qu'un modèle d'environnement virtuel soit capable de représenter les comportements des objets.

Certains Environnements virtuels, les Environnements virtuels collaboratifs, utilisent plusieurs acteurs en même temps. En formation par exemple, ils jouent le rôle des équipiers [Berthelot et al., 2014] ou des formateurs [Johnson and Rickel, 1997]. En narration interactive ils sont les autres personnages importants de l'histoire [Cavazza et al., 2007; Paiva et al., 2001]. La collaboration entre les acteurs peut aller de la simple perception à la manipulation collaborative d'objets. Un modèle d'environnement virtuel doit permettre la représentation des actions collaboratives allant jusqu'à la comanipulation d'objets afin d'offrir un panel d'actions le plus large possible et répondre aux maximums de cas d'utilisation.

Nous allons étudier les solutions existantes d'après les critères suivants :

- Les deux types de composants de l'environnement virtuel qui nous semblent les plus importants sont les objets et les actions (et leurs comportements associés). Ils doivent prendre une part importante dans le modèle. Nous étudions donc ici ***comment sont représentés les comportements des objets et des actions?***

- Comme l'environnement virtuel est l'élément sur lequel repose la simulation, le modèle utilisé pour le définir doit permettre d'exprimer un grand nombre de cas d'utilisation. Il doit notamment permettre de représenter tous les niveaux nécessaires de collaboration entre les acteurs (réels et/ou virtuels). Nous étudions donc ***quel est le niveau de collaboration offert par le modèle?***

- Afin de connaitre comment les autres composants peuvent interagir avec l'environnement virtuel, nous cherchons à connaitre ***quelle est la nature des Entités manipulées?***

La suite de cette section est découpée en trois parties. La section 1.2.1 s'intéresse aux *modèles de comportements* qui ont étés créés pour définir les comportements en Environnements virtuels de manière générale. La section 1.2.2 porte sur les *objets synoptiques*, une famille de modèles permettant de décrire les comportements liés aux objets dans un environnement virtuel. Dans la section 1.2.3 nous étudions la famille des *modèles objets-relations*, qui considèrent les actions comme des relations entre objets.

1.2.1 Les modèles de comportements génériques

Certains modèles ont été spécifiquement développés pour décrire le comportement des entités composant un environnement virtuel. Ces modèles reposent sur leurs capacités à observer l'environnement et à réagir aux évènements. Leur formalisme repose sur des machines à états hiérarchiques : un état est une autre machine ou un état atomique. Les états atomiques demandent la spécification de comportements (via des fonctions ou des scripts) pour interagir avec l'environnement virtuel. La Figure 2.4 donne un exemple de machine à états hiérarchique. Elle est composée de deux types d'entités : les états et les transitions. Une transition permet de passer d'un état à un autre. Ici M1 possède un état de départ (cercle noir) et deux autres états M11 et E12. M11 est lui même composé de cinq états : un état de départ trois états intermédiaires E21, E22 et E23 et un état final. L'état de départ indique le début du comportement. E21 exécute sa fonction puis passe la main à E22 puis à E23. À la fin de E23, M1 change d'état et passe à E12. Une fois E12 terminé, E21 reprend la main.

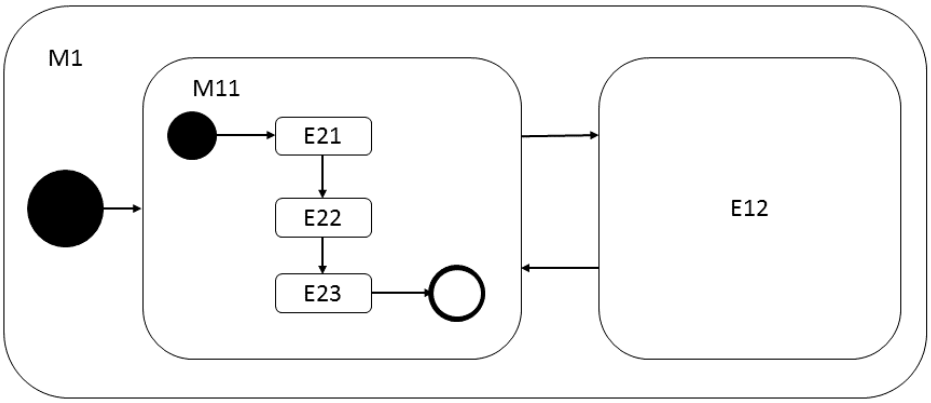


Figure 2.4 – Exemple de machine à état hiérarchique

Les "*modèles de comportements*" ne proposent pas d'entités spécifiques pour la représentation des actions ou des objets ce qui complexifie leur utilisation avec les autres composants de l'environnement virtuel. De plus, ils ne prennent pas directement en compte les acteurs. Ils ne prévoient donc pas les mécanismes nécessaires à la collaboration. Il est possible de les utiliser dans des contextes multiacteurs, mais cela risque de demander beaucoup de travail ou l'utilisation de modèles complémentaires. De manière générale, ils peuvent s'adapter à beaucoup de contextes d'utilisation, mais demandent des développements supplémentaires pour faire le lien avec l'environnement.

A Hierarchical Concurrent State Machines (HCSM)

HCSM [Cremer et al., 1995] est un framework pour définir les comportements des objets et les scénarios d'environnements virtuels basés sur les Statecharts [Harel, 1987]. Un état atomique

Solution	Types des entités	Nv. Collab.	Comportements
HCSM	Fonctions	non prévue	Fonctions + Structure
HPTS++	Fonctions et Ressources	non prévue	Fonctions + Structure

Table 2.1 – Résumé des modèles de Comportements.

d'une HCSM est une fonction définie lors de la spécification. Les HCSM parentes (c.-à-d. non atomiques) ont deux types: séquentielles (c.-à-d. une seule sous-HCSM est active à la fois) ou concurrentes (toutes les sous-HCSM sont actives en même temps). Les HCSM concurrentes offrent la possibilité de décrire l'ensemble des comportements des objets de l'environnement. La Figure 2.5 donne un exemple d'HCSM. La sous-machine 1 est concurrente, les machines 2 et 3 sont séquentielles. Les états des sous-machines sont des états atomiques.

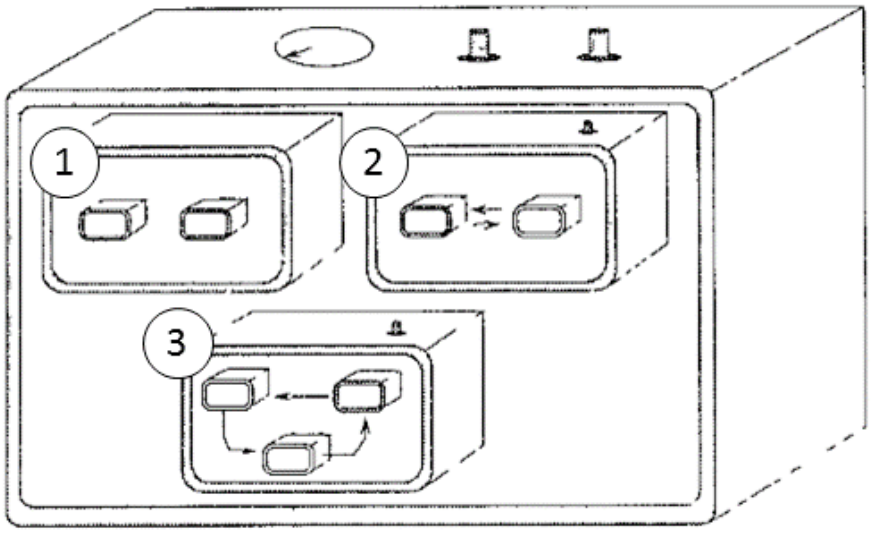


Figure 2.5 – Exemple de HCSM (Origine [Cremer et al., 1995])

B Hierarchical Parallel Transition Systems ++ (HPTS++)

HPTS++ [Lamarche and Donikian, 2002] est une extension de HPTS [Donikian, 2001], un langage similaire à HCSM. Il y ajoute cependant une fonctionnalité : la gestion des **ressources** qui permet de synchroniser des comportements. Par exemple, si deux comportements nécessitent un tournevis, mais qu'il est déjà utilisé par un des deux, l'autre devra attendre que le tournevis soit libéré. Le tableau 2.1 résume les modèles de cette famille.

1.2.2 Les objets synoptiques

La seconde famille de modèles, les "*objets synoptiques*" utilisent l'objet virtuel en tant qu'interface pour les actions. Les actions qui utilisent un objet lui sont directement attachées et sont disponibles ou indisponibles en fonction de l'état de l'objet. L'acteur *utilise* alors l'objet pour effectuer une action. Par exemple, l'action "*allumer la lumière*" est attachée directement à l'objet "*interrupteur*". L'acteur utilise l'interrupteur pour allumer la lumière. L'interrupteur change alors d'état et sa prochaine utilisation déclenchera l'action "*éteindre la lumière*". Cette approche est très efficace lorsqu'il s'agit d'impliquer un unique objet dans une action. Il est beaucoup plus difficile d'exprimer des actions qui impliquent des combinaisons d'objets au choix de l'utilisateur. Il faut définir explicitement chaque action avec chaque combinaison d'instances d'objets possibles. Ceci peut vite devenir très fastidieux à mettre en place. Les actions collaboratives de co manipulation sont également difficiles à mettre en pratique puisqu'il faut prendre en compte les différentes combinaisons possibles des actions des acteurs dans le comportement de l'objet. Les actions de manipulation exclusives sont réalisables si un objet peut avoir plusieurs comportements en parallèle.

A Smart-Objects

Smart-Objects [Kallmann and Thalmann, 1999] propose d'intégrer dans l'objet les informations nécessaires à toutes les actions possibles le concernant.

- *Ses propriétés intrinsèques* définissent l'objet en lui-même ce qui l'implique dans l'Environnement (p. ex. ses parties mobiles ou ses propriétés physiques comme son poids.)
- *Son comportement* est défini par son état. Par exemple une porte automatique peut se fermer si elle est ouverte et qu'il n'y a rien dans sa trajectoire.
- *Des informations d'interaction* décrivent les zones d'interaction (p. ex. une poignée de porte) et/ou son effet sur l'acteur (par exemple des contraintes cinématiques).
- *Le comportement des acteurs* décrit comment l'acteur est censé se comporter (p. ex. animations, changements d'état) face au comportement de l'objet.

La définition des objets demande la définition de tous ses comportements possibles. Ceci est défini dans un langage spécifique associé à son propre environnement de développement.

B Starfish

Starfish [Badawi and Donikian, 2004] repose sur des objets contenant une description de leur interactivité. Cette information se base sur deux composants: les Actions et les surfaces interactives. Les actions sont définies soit par des actions atomiques prédéfinies (par exemple déplacer, donner, prendre ou parler), soit par une composition de ces actions atomiques. Les surfaces interactives définissent les parties de l'objet qui correspondent à l'exécution d'une action.

Par exemple la poignée d'une valise peut être liée à l'action "*prendre*". Les objets contiennent une machine à états, définie en HPTS++ [Lamarche and Donikian, 2002], qui indique quelles sont les actions disponibles en fonction de l'état de l'objet. L'exécution d'une action change l'état courant de l'objet. Il est impossible de déclencher plusieurs actions en parallèle sur un même objet. Cela rend difficile la modélisation de co manipulations. Le modèle permet de réutiliser un automate pour des objets proposant le même comportement. Par contre, pour chaque objet il faut définir la relation entre la surface interactive et le comportement associé.

C Marigold

Le framework Marigold [Willans and Harrison, 2001a] [Willans and Harrison, 2001b] propose d'exprimer le comportement des objets en utilisant une représentation hybride (comme proposé par Smith et al. [Smith et al., 1999]) constituée :

- d'entrées/sorties continues,
- d'entrées/sorties discrètes,
- d'une représentation de l'état et du comportement de l'objet par un réseau de Petri saufs (un jeton par place maximum, voir par exemple [Glynn, 1987] [Murata, 1989] ou l'Annexe A).
- d'un modèle continu décrivant les règles d'interactions avec l'objet.

La Figure 2.6 donne un exemple de comportement d'objet utilisant ce modèle . Ce réseau représente une partie du fonctionnement d'une cuisinière à gaz : Si le bouton de mise a feu est activé (IGNITION ON) alors l'étincelle se produit (SPARK). Si le gaz est ouvert (KNOB ON puis GAS), alors la flamme s'allume (FLAME ON). Elle reste allumée tant que le gaz est ouvert. Les propriétés mathématiques des réseaux de Petri permettent de vérifier que la conception est correcte par exemple en détectant les interblocages.

La modélisation des interactions repose sur le formalisme des Flownet [Smith et al., 1999]. L'idée est de prendre en compte directement les données des utilisateurs et de l'état de l'environnement et de calculer leurs effets sur les entrées de l'objet. À l'inverse, l'état interne de l'objet permet de calculer des informations à retourner à l'utilisateur. La Figure 2.7 donne un exemple d'interaction d'un utilisateur avec la cuisinière à gaz. Ici, les données prises en compte sont la position de la main et les positions des différents interrupteurs. Il est possible de représenter des comanipulations en prenant en compte les données liées à plusieurs acteurs, mais ceci semble complexe à mettre en place. Le tableau 2.2 résume les modèles de la famille des objets synoptiques.

1.2.3 Les modèles Objets-relations

Les "*modèles objets-relations*" utilisent un type d'entités supplémentaires pour permettre l'implication de plusieurs objets : les relations. Une relation définit deux éléments:

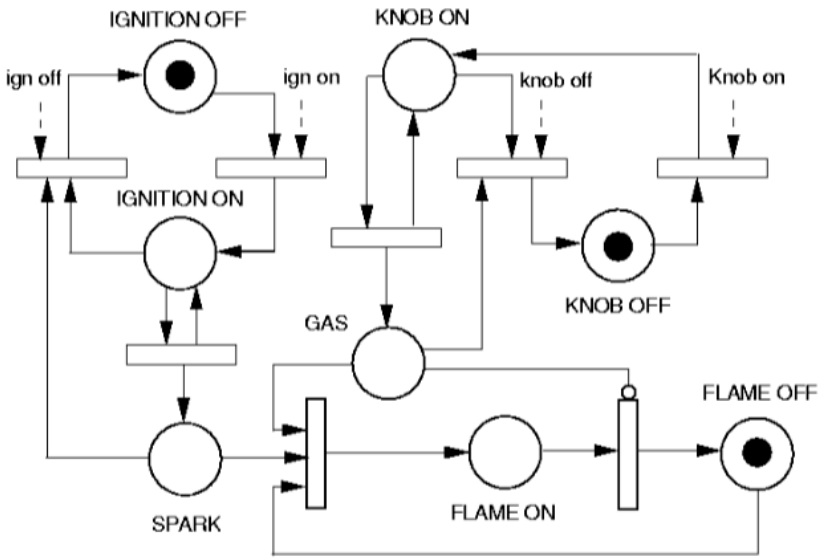


Figure 2.6 – Exemple de comportement d'objet modélisé par Marigold. (Origine [Willans and Harrison, 2001b])

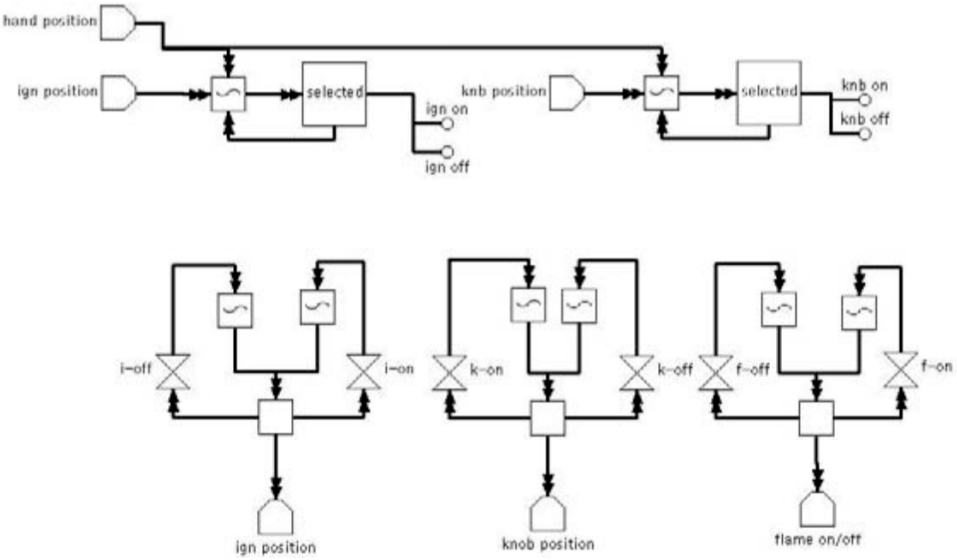


Figure 2.7 – Exemple de modélisation d'interaction avec Marigold. (Origine [Willans and Harrison, 2001a])

- Un comportement qui implique différents objets et modifie donc leurs états,
- Des préconditions sur les types et l'état des objets qui peuvent être impliqués dans l'exécution de la relation.

Solution	Types des entités	Nv. Collab.	Comportements
Smart-Objects	Objets	2.2	Lié à l'Objet
Starfish	Surfaces interactives	2.2	Lié à l'objet
Marigold	Objets + Interactions	3.2	Lié à l'Objet

Table 2.2 – Synthèse des modèles d'objets synoptiques. Le niveau de collaboration est basé sur [Margery et al., 1999]

Une relation bien définie peut donc être utilisée pour garantir la cohérence de l'environnement entre avant et après sa réalisation.

Certains modèles objets-relations peuvent de prendre en compte plusieurs acteurs et objets dans la réalisation d'une action. Ils sont donc capables de modéliser les interactions de comanipulation.

A Cause and effects

Dans Cause and Effects, Lugrin et Cavazza [Lugrin and Cavazza, 2006] proposent de représenter les actions en utilisant trois éléments:

- L'évènement déclencheur, par exemple la collision entre deux objets
- Des préconditions sur l'état du monde pour que l'action soit possible,
- Les effets de l'action sur l'état du monde.

Les préconditions sont définies dans une ontologie. Chaque objet possède des types qui lui permettent ou non de participer à la réalisation d'une action d'après les préconditions. Cependant, les effets doivent être définis pour chaque instance d'objets. Par exemple si deux vis sont définies dans l'environnement, il faut définir l'action visser utilisant la vis 1 et celle utilisant la vis 2.

B (Simulation and Training Object-Relation Model (STORM)

STORM [Mollet et al., 2007] repose sur deux modèles spécifiques: les objets comportementaux et les relations. Un objet STORM (voir Figure 2.8) contient des capacités: une activité interne associée à une interface d'interactions définissant un protocole de communication avec d'autres objets STORM. Le modèle n'impose pas de formalisme pour la définition des activités.

Les relations STORM sont des objets STORM spécifiques qui représentent le lien qui existe entre des objets lors d'une interaction. Une relation utilise les capacités des objets (voir Figure 2.8) pour les faire interagir entre eux. STORM permet d'utiliser plusieurs relations, en même temps, avec un même objet. Il est donc possible de réaliser des actions collaboratives de comanipulation. De plus l'objet adapte son comportement dans les relations en parallèle comme dans un modèle de système hybride. Les relations résonnent sur les interfaces proposées par les objets. La Figure 2.9 donne un exemple de relation. Une relation visser permet de mettre

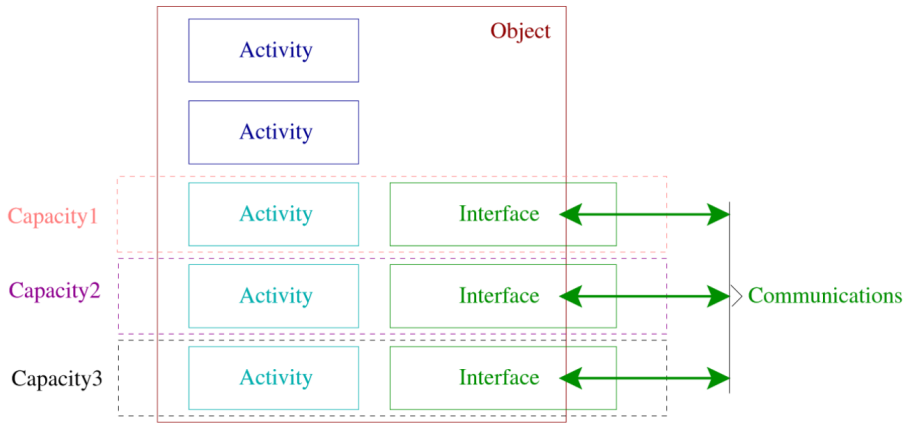


Figure 2.8 – Structure des objets STORM (Origine [Mollet et al., 2007]).

en relation trois objets : une prise un support femelle et un tournevis. La relation définit une interaction entre ces trois objets : elle transfère le mouvement du tournevis vers la prise. Elle maintient également le lien mécanique entre la prise et le support femelle. Les objets communiquent entre eux via la relation. Ce modèle est très générique, car la modélisation des actions et des objets est indépendante. Ajouter un objet dans l'environnement permet de l'utiliser directement avec les actions associées et inversement.

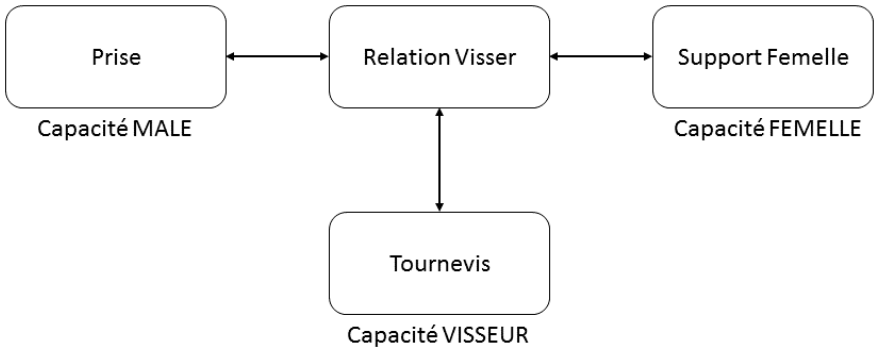


Figure 2.9 – Exemple de Relation STORM (Origine [Mollet et al., 2007]).

C VEHA

VEHA [Chevaillier et al., 2012] est un composant du framework MASCARET (utilisé notamment dans ses différentes versions dans les environnements virtuels pour la formation SécuRéVi[Querrec et al., 2003] Gaspar[Marion N. and Querrec, 2007] ou EAST [Taoum et al., 2015]). Il est basé sur UML [Rumbaugh et al., 2004] et permet de représenter des concepts

comme la relation conceptuelle, physique ou la composition d'objets (voir Figure 2.10).

Les différents comportements d'un objet (voir Figure 2.11) peuvent être représentés par :

- Une machine à état - Un évènement spécifique fait transiter d'un état à un autre et peut déclencher d'autres comportements internes.
- Une boite noire - Un code spécialisé sans représentation sémantique spécifique.

Les évènements peuvent être déclenchés par l'interaction entre les objets, les actions des acteurs ou le comportement interne des objets.

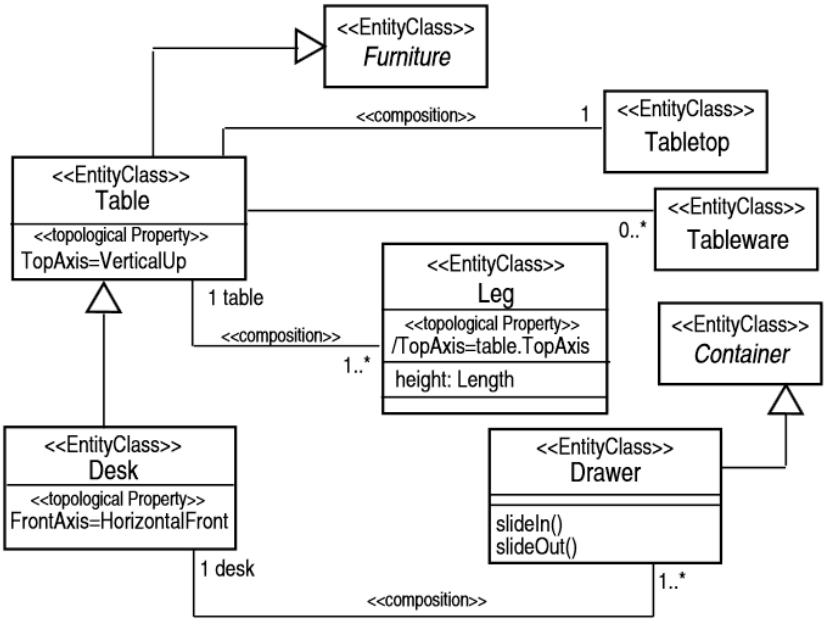


Figure 2.10 – Définition d'un bureau en VEHA (origine [Chevaillier et al., 2012]).

VEHA permet à plusieurs acteurs d'utiliser différents comportements d'un même objet ou la comanipulation. Une difficulté réside cependant dans la synchronisation des différents comportements d'un objet complexe en utilisant des machines à états.

D Domain-DL

Domain-DL, composant de la plateforme HUMANS [Carpentier, 2015; Lanquepin et al., 2013] utilisée notamment dans des environnements virtuels de formation tels que [Amokrane et al., 2014] ou V3S [Barot et al., 2013], repose sur les définitions de **contraintes** entre l'état de l'environnement virtuel, les **actions**, les **comportements** et les **évènements**.

Un **objet** est un ensemble de propriétés. Il peut être de trois types. Un objet de type *abstrait* n'a pas de représentation visuelle (p. ex. Un ordre donné par un chef d'équipe). Un

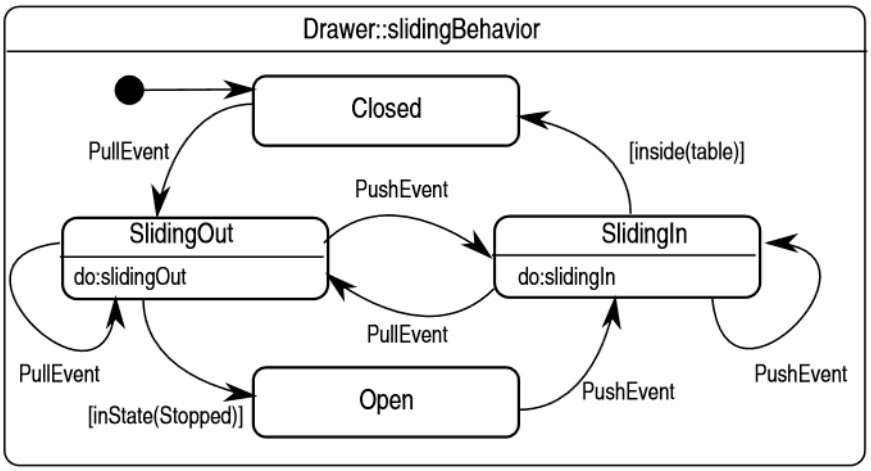


Figure 2.11 – Définition du comportement d'un tiroir en VEHA (origine [Chevaillier et al., 2012]).

objet de type *concret* possède une représentation visuelle (p. ex. Une chaise). Enfin, un objet de type *Agent* est capable d'interagir de manière autonome avec l'environnement virtuel (p. ex. Un humain virtuel).

Les **actions** définissent des relations entre les objets. Elles sont réalisées par un agent. Elles possèdent des paramètres pouvant être des "*objets*" et/ou des données. Les données peuvent être décrites dans les types de bases flottant, booléen ou chaine de caractères. Une action est réalisable si des préconditions sur les types des objets et l'état du monde sont validées. Une action réalisable crée des comportements.

Les **comportements** précisent des conditions d'exécution et des effets sur les objets. Les conditions d'exécution peuvent porter par exemple sur l'état du monde, la réalisation d'une action ou le déclenchement d'un évènement. Une fois déclenché, un comportement modifie l'état d'un ou plusieurs objets.

Les **évènements** peuvent être ponctuels ou durer dans le temps [Carpentier, 2015]. Un évènement avec une durée est décrit avec deux évènements ponctuels : son début et sa fin. Un évènement ponctuel est créé sous certaines conditions. Par exemple une cuve qui atteint son maximum de remplissage déclenchera l'évènement "*La Cuve est pleine*".

E Framework for Interaction in Virtual Environements (#FIVE)

#FIVE [Bouville et al., 2015] reprend les travaux de Saraos Luna [Saraos Luna et al., 2012] dans lesquels était proposé une version étendue de STORM permettant la modélisation les interactions collaboratives. #FIVE s'intéresse à deux concepts : les relations et les interactions.

Dans #FIVE, les objets sont représentés par un ensemble de types. Par exemple un scalpel est un *Instrument*,et un *Tranchant*. Une **relation** définit des contraintes sur les **types** des

Solution	Types des entités	Nv. Collab.	Comportements
Cause and Effects	Objets + Relations	3.2	dans les Relations
STORM	Objets + Relations	3.2	Lié à l'Objet
MASCARET	Objets + Relations	3.2	Lié à l'Objet
World-DL	Comportements + Relations + Objets + Évènements	3.2	indépendant
#FIVE	Objets + Relations + interactions	3.2	Objets et/ou Relations

Table 2.3 – Synthèse des modèles objets-relations. Le niveau de collaboration est basé sur [Margery et al., 1999]

objets. Par exemple la relation *inciser* nécessite deux objets: un objet *instrument* et *tranchant* et un objet *patient*

Une **réalisation** est une instance de relation utilisant les objets de l'environnement. Elle propose deux fonctionnalités:

- Vérifier que l'état des objets est compatible avec l'exécution d'un comportement,
- Exécuter effectivement ce comportement en modifiant l'état d un ou plusieurs objets impliqués.

Les **interactions** sont définies en utilisant deux concepts : les **attributs contrôlables** des objets et les **interacteurs** qui lient un ou plusieurs attributs de différents objets par une fonction. Par exemple, un interacteur peut être utilisé pour modéliser le lien entre un tournevis et une vis. Ainsi la rotation du tournevis impacte la vis.

Les composants de #FIVE peuvent être utilisés conjointement ou indépendamment. L'Annexe B présente en détail ces composants. Le tableau 2.3 résume les modèles de la famille des modèles objets-relations.

1.2.4 Conclusion

Les modèles Objets-Relation offrent les possibilités des autres modèles (c.-à-d. modélisation de comportements, manipulation d'attributs non liés d'un même objet) et en proposent des supplémentaires (c.-à-d. comanipulation, actions génériques). Le tableau 2.4 résume les modèles étudiés dans cette section. Dans la suite de ce manuscrit nous nous considèrerons que le modèle utilisé pour représenter l'Environnement virtuel offre les mêmes propriétés que les modèles Objets-Relations. Notamment la possibilité de maintenir cohérence de l'environnement par les préconditions et la possibilité de modéliser les co manipulations.

1.3 Synthèse

Un environnement virtuel est peuplé d'objets et d'acteurs. Les acteurs peuvent être contrôlés par des utilisateurs en utilisant des interfaces sensorimotrices ou entièrement virtuelles.

L'environnement virtuel peut être modifié au cours de la simulation, par les objets et les acteurs, via des actions. Les actions définissent des changements possibles de l'état de l'Environnement. Si l'environnement contient plusieurs acteurs, il est collaboratif. La collaboration peut atteindre différents niveaux en fonction des possibilités offertes par le modèle d'environnement virtuel utilisé. Parmi ces modèles la famille des modèles objets-relations est la plus complète. Ces modèles offrent la possibilité aux autres composants de manipuler directement les objets de l'environnement et les actions. De plus, ils offrent plus simplement le niveau de collaboration le plus haut : la comanipulation. Enfin, grâce aux modèles existants, il est possible de représenter des Environnements virtuels offrant de grandes possibilités d'action.

Solution	Types des entités	Nv. Collab.	Comportements
Modèles de comportements génériques			
HCSM	Fonctions	non prévue	Fonctions + Structure
HPTS++	Fonctions et Ressources	non prévue	Fonctions + Structure
Objets synoptiques			
Smart-Objects	Objets	Interaction Libre	Lié à l'Objet
Starfish	Surfaces interactives	Interaction Libre	Lié à l'objet
Marigold	Objets + Relations	Exclusion	Lié à l'Objet
Modèles Objets-Relations			
Cause and Effects	Objets + Relations	Comanipulation	dans les Relations
STORM	Objets + Relations	Comanipulation	Lié à l'Objet
MASCARET	Objets + Relations	Comanipulation	Lié à l'Objet
World-DL	Comportements + Relations + Objets + Évènements	Comanipulation	indépendant
#FIVE	Objets + Relations + interactions	Comanipulation	Objets et/ou Relations

Table 2.4 – Synthèse des modèles de comportements d'Environnements virtuels. Le niveau de collaboration est basé sur [Margery et al., 1999]

2 Séquencement des actions

L'environnement virtuel à lui seul est rarement suffisant pour réaliser puis utiliser une simulation dans un but précis. D'autres composants logiciels vont pouvoir interagir avec lui pour fournir plus de fonctionnalités au système. Par exemple, pour les Environnements Virtuels Collaboratifs de Formation, Gerbaud et al. [Gerbaud et al., 2009] proposent l'utilisation d'une description des tâches à réaliser, d'un système de répartition des actions entre les acteurs et de paramétrage pédagogique de la session des utilisateurs. Dans la littérature, la description des tâches à réaliser et la répartition des actions entre les acteurs sont associés aux **Scénarios**.

Au cinéma, le scénario est un "*document écrit décrivant le film qui sera tourné*" [Larousse, 2016]. Il définit donc les actions qui vont avoir lieu ainsi que leur agencement logique et temporel. L'utilisateur final (le spectateur) ne peut pas influer sur ce scénario. Le même constat peut être fait pour un roman ou une nouvelle. Pourtant, Szilas [Szilas, 2003] reprend Umberto Eco [Eco, 1985] : "*un texte a besoin de quelqu'un pour l'aider à fonctionner.*". Szilas fait la distinction entre la narration linéaire (livre, cinéma) et la narration interactive. Dans cette dernière, l'utilisateur peut avoir un impact sur le déroulement des évènements. Dans certains cas il est directement impliqué [Ponder et al., 2003], dans d'autres il va simplement influencer les autres acteurs en discutant avec eux [Aylett et al., 2005] ou en modifiant l'environnement [Magerko et al., 2004]. La difficulté consiste alors à définir quelles sont les possibilités d'actions qui sont offertes aux acteurs et quelles sont celles qui doivent leur être retirées afin de maintenir une cohérence de l'environnement vis-a-vis des objectifs de l'utilisation de la simulation. Certains états de l'environnement virtuel ne doivent pas être atteints, car ils sont incompatibles avec ces objectifs. Par exemple, si un acteur rend inaccessible ou détruit un outil indispensable à une procédure de maintenance il ne sera pas en mesure de la mener jusqu'au bout. Il peut donc être judicieux de retirer à l'acteur la possibilité de réaliser ces actions. À l'inverse, il peut être nécessaire que l'environnement virtuel atteigne certains états au cours de la simulation. Par exemple le héros de l'histoire doit entrer en possession d'un trésor pour pouvoir continuer son périple. Puisque l'état de l'environnement virtuel doit être maitrisé, il faut mettre en place des mécanismes de contrôle des actions. Dans la littérature chaque Système propose sa propre vision du scénario. Selon Gerbaud [Gerbaud, 2008] "*un scénario doit permettre de* ***décrire*** *l'****agencement*** *à la fois* ***temporel*** *et* ***logique*** *des* ***actions***". Selon Barot [Barot, 2014] il s'agit d'un "*ensemble d'****évènements*** *particuliers, partiellement* ***ordonnés*** *et instanciés dans un environnement virtuel*". En narration interactive les termes "*scénario*", "*histoire*" et "*intrigue*" sont utilisés de manière équivalente pour décrire les actions qui peuvent avoir lieu au cours de la simulation [Brom and Abonyi, 2006] ou qui ont lieu lors d'une session d'utilisation de l'environnement virtuel[Cavazza et al., 2008].

2.1 Scénarios pour les environnements virtuels

En nous basant sur la littérature, il nous parait difficile de savoir si un scénario décrit une séquence spécifique d'actions ou l'ensemble des séquences possibles. Pour éclaircir ce point, dans la suite de ce manuscrit nous appelons **scénario** "*l'agencement temporel et causal des actions dans l'environnement virtuel lors d'une session d'utilisation du système de réalité virtuelle.*". Nous définissons un **modèle de scénarios** comme le "*formalisme et l'organisation de données permettant de décrire des scénarios ainsi que les mécanismes qui permettent leurs utilisations par les composants du système de réalité virtuelle*" (voir Figure 2.12). Nous appelons **spécification des scénarios** la description de l'ensemble des scénarios possibles. Une spécification des scénarios est réalisée à partir d'un modèle. Enfin, certains systèmes utilisent un moteur de scénarios (voir Figure 2.13). Ce dernier se base sur une spécification de scénarios pour contraindre ou orienter les actions au cours de la simulation.

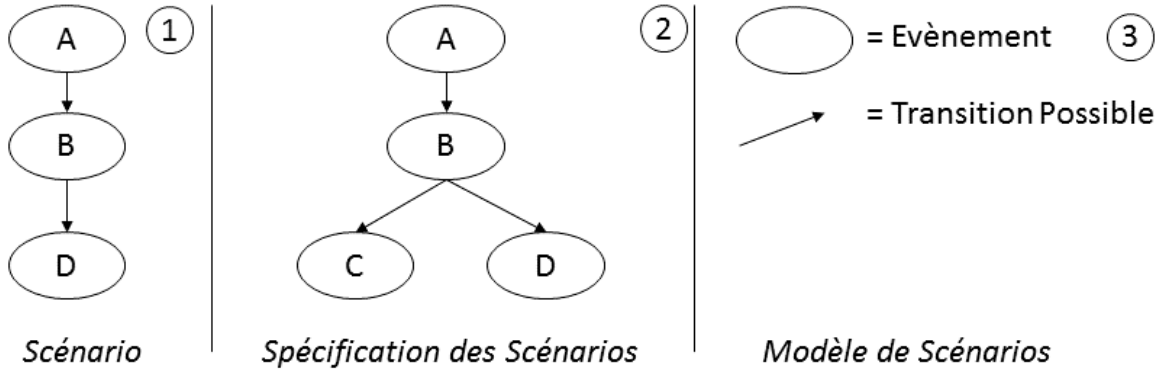


Figure 2.12 – Le scénario (1) est une des exécutions possibles prévues par la spécification des scénarios (2). La spécification est basée sur un modèle de scénario (3) qui définit quelles sont les données et comment les utiliser.

Comme les actions définissent des séquences d'évènements (voir section 1.2), la spécification de l'ensemble des scénarios consiste à définir quelles actions sont peuvent être réalisées (par les acteurs ou non) en fonction de l'état de l'environnement virtuel et de l'avancement de la simulation. En fonction des objectifs de l'utilisation de l'environnement virtuel, les acteurs devront être plus ou moins guidés dans leurs actions. Prenons par exemple un environnement virtuel de formation au montage d'une culasse utilisée dans deux cas distincts : la formation et l'évaluation. Dans le cas de la formation, il peut s'agir d'apprendre à l'utilisateur la procédure exacte à réaliser. Les actions accessibles à chaque étape sont alors limitées aux actions correctes. L'ensemble des scénarios possibles est plus restreint sur les actions disponibles à chaque étape du montage (voir Figure 2.14). Dans le cas de l'évaluation, les actions possibles sont toujours toutes disponibles, même si elles sont incohérentes avec la procédure. Une fois que la culasse est montée, une évaluation de la séquence d'actions réalisée par l'utilisateur est réalisée par rapport à

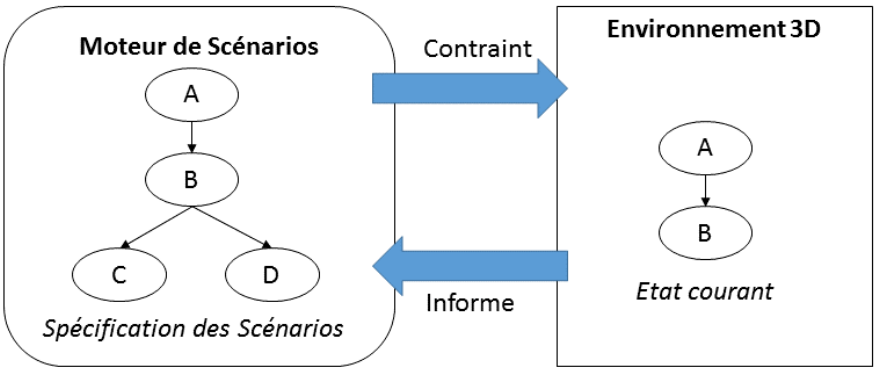


Figure 2.13 – un moteur de scénarios impose des contraintes aux entités de l'environnement virtuel afin de contrôler de manière plus ou moins forte les actions pouvant y avoir lieu. Pour ce faire, le moteur utilise une spécification des scénarios possibles et l'état de l'environnement.

l'ensemble des séquences prévues par la procédure. En fonction des besoins de lier à l'utilisation de l'environnement virtuel, le Modèle de scénarios doit permettre d'exprimer différents **niveaux de contrôle des acteurs**.

La notion d'ordre dans l'ensemble des évènements d'un scénario est très importante. On la retrouve dans les systèmes de formation [Barot, 2014; Gerbaud, 2008] comme dans la narration interactive [Theune et al., 2003]. Lors de la spécification des scénarios possibles, il faut prendre en compte que certains évènements doivent avoir lieu avant ou après d'autres en fonction de l'état de l'environnement et de l'avancement global de la simulation. Certains sous-scénarios seront obligatoires alors que d'autres auront des alternatives ou seront optionnels. Reprenons en exemple le montage d'une culasse. Supposons que nous ayons plusieurs acteurs pouvant réaliser la séquence de montage, mais un seul tournevis disponible. Alors, seul un des acteurs peut réaliser le montage. Pendant ce temps les autres acteurs peuvent effectuer d'autres tâches, avec leurs propres séquences d'actions. La simulation se terminera lorsque toutes les tâches auront été réalisées. Le modèle doit permettre d'exprimer les **contraintes de causalité et de temporalité entre les actions** (parallélisme, concurrence, séquences).

Dans notre exemple, le tourne-vis est une ressource nécessaire à l'exécution des actions. Les ressources nécessaires ont un impact important sur la spécification des scénarios. Elles participent à la synchronisation des actions en fonction de leur disponibilité. Par exemple, deux actions ne sont pas réalisables en parallèle s'il n'y a qu'un seul outil de disponible pour les réaliser. Les ressources participent également à la spécification des actions possibles. Le montage de notre exemple doit se faire en croix en prenant en compte le point de départ (voir Figure 2.14). Ceci donne 8 scénarios possibles sans prendre en compte les ressources. Si maintenant nous prenons en compte qu'il y a quatre vis v_1 à v_4 et que l'action "*visser*" utilise une vis et un emplacement alors il existe exactement 192 scénarios possibles. Nous pouvons ajouter à ceci le nombre de tournevis disponibles. La combinatoire action/ensemble d'objets fait exploser le nombre d'actions à prendre en compte dans l'agencement. De plus, si l'environnement est modifié

(ajout d'un tourne-vis par exemple), il faut le répercuter sur la spécification des scénarios. Une solution consiste à **généraliser les définitions des actions** à réaliser. Dans notre exemple les définitions d'actions indiqueront spécifiquement l'emplacement, mais les vis et tourne-vis ne seront pas définis explicitement. Ainsi seuls les 8 scénarios initiaux seront à décrire.

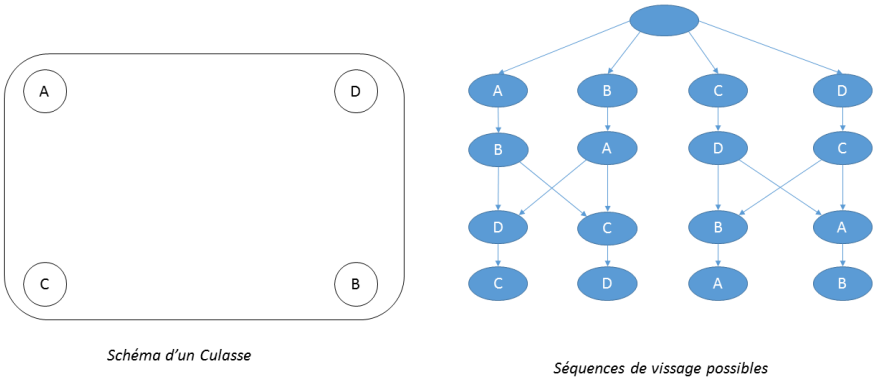


Figure 2.14 – Les vis doivent être placées, en croix, dans les pas de vis associés (A, B, C, D). L'automate à droite donne les différents scénarios possibles.

Tous les évènements n'ont pas pour origine les actions des acteurs. Il peut être intéressant de modifier l'environnement indépendamment de leur volonté afin de les obliger à gérer des situations non prévues. Dans le cadre de narration interactive, l'objectif peut être d'ajouter de la tension narrative, par exemple en coupant l'électricité dans un bâtiment, plongeant ainsi ses occupants dans le noir. En formation il peut s'agir d'évaluer la capacité des apprenants à gérer une situation de crise. Par exemple comment réagir si l'alarme à incendie se déclenche en plein milieu d'une procédure risquée comme une opération chirurgicale. Un modèle de scénarios doit donc permettre de **déclencher des évènements** dans l'environnement hors du contrôle des acteurs.

Le nombre et la nature des acteurs (agent virtuel ou utilisateur) vont eux aussi dépendre de l'utilisation prévue du Système de Réalité virtuelle et de l'environnement virtuel. En narration interactive, par exemple dans l'adaptation de Madame Bovary de Cavazza et al. [Cavazza et al., 2007], un unique utilisateur est intégré à l'histoire en tant qu'un des protagonistes. Tous les autres acteurs sont des agents virtuels. Dans CORVETTE [Berthelot et al., 2014], les acteurs peuvent être indépendamment des utilisateurs ou des agents virtuels et participent à une procédure de maintenance industrielle. Un modèle de scénarios ne doit **pas faire d'hypothèses sur la nature ou le nombre d'acteurs** sous peine de limiter les cas d'utilisation.

Enfin, que ce soit en formation [Mollet and Arnaldi, 2006] ou en narration interactive [Cavazza et al., 2008] la spécification des scénarios reste un problème majeur. Une des raisons est la difficulté pour l'auteur d'appréhender le problème dans son ensemble. L'utilisation d'une représentation graphique est reconnue comme un moyen d'aider à la compréhension de systèmes [Sugiyama et al., 1981]. Ceci permet également à des non-informaticiens, par exemple les experts d'un domaine lors de la réalisation d'un système de formation, de participer plus activement

à la conception. Sur ce point les modèles dont le formalisme permet une **représentation graphique** possèdent un avantage non négligeable. Par exemple, le projet EAST [Taoum et al., 2015] vise à intégrer fortement dans la conception d'un environnement virtuel de formation sur les experts métiers, les pédagogues et les formateurs.

2.2 Modèles de scénarios pour environnements virtuels

La spécification des scénarios permet de définir quelles sont les actions qui vont être réalisées par les acteurs en fonction de l'avancement de la simulation. La famille des *modèles de scénarios prédéfinis* répond directement en proposant une description de l'ensemble des scénarios possibles définie par un spécialiste/auteur. Une autre famille de solutions s'intéresse également à ce problème. Les *modèles à scénario émergent* reposent sur les comportements des acteurs, contraints par des règles, pour obtenir le scénario directement au cours de la simulation. Sur la base de la section précédente, nous nous intéressons aux propriétés suivantes :

- Le formalisme de la solution permet-il d'exprimer des ***Agencements*** temporels et causaux complexes ?
- Comment le formalisme de la solution permet-il de prendre en compte les ***Ressources*** disponibles ?
- Les mécanismes de la solution permettent-ils de définir des scénarios ***actifs*** ? C'est-à-dire de déclencher des évènements indépendamment des actions des acteurs ?
- Le modèle s'intéresse-t-il à la ***Collaboration*** ? Fait-il des hypothèses sur le nombre ou la nature des acteurs ?
- Le formalisme de la solution permet-il d'utiliser différents niveaux de ***Contrôle*** des acteurs ?
- La solution propose-t-elle une représentation ***graphique*** du formalisme utilisé ?

La section 2.2.1 s'intéresse à la famille des **Modèles de Scénarios prédéfinis** et la section 2.2.2 à la famille des **modèles à scénarios émergents**.

2.2.1 Modèles à scénarios prédéfinis

Les modèles à scénarios prédéfinis reposent sur une description complète de l'ensemble des scénarios possibles lors de la simulation. Leur formalisme se base de manière générale sur des machines à états ou des réseaux. Ils peuvent donc, au moins partiellement, être représentés de manière graphique. Pour la plupart, ils sont également capables de décrire les contraintes de temporalité et de causalité. Sauf cas particulier (voir Paragraphe B), ils peuvent prendre en compte la collaboration d'au moins un acteur réel avec des acteurs virtuels.

A Story Nets

Les Story Nets, utilisés dans Mission Rehearsal Exercise (MRE) [Hill Jr et al., 2003; Rickel et al., 2002; Swartout et al., 2005, 2006a,b; Traum et al., 2003], sont des machines à états. Leur utilisation met en situation un seul utilisateur avec un ensemble d'humains virtuels pouvant raisonner, discuter et montrer des émotions. Le modèle ne propose qu'un seul niveau de contrôle : chaque noeud est une scène du scénario global ou l'utilisateur doit agir et/ou faire des choix. En fonction du résultat prévu de la scène, la machine à état déclenchera la transition vers un nouveau noeud. Il n'est pas possible de décrire des scénarios parallèles. Un seul utilisateur est pris en compte. Une transition d'un état à un autre est un script fixe ou le scénario avance sans intervention possible de la part de l'utilisateur. Ce script décrit les animations et les changements d'état de l'environnement. La machine à état permet une représentation graphique partielle. Enfin, la gestion des ressources n'est pas prise en compte par le modèle et doit être modélisée lors de la spécification d'un noeud sauf si elle intervient dans les conditions de transition d'un noeud à l'autre.

B Hierarchical Concurent State Machines (HCSM) et Hierarchical Parallel Transition System ++ (HPTS++)

Certains modèles ont été créés pour répondre directement au problème des scénarios pour Environnements virtuels. Ils considèrent la gestion des scénarios comme le comportement général de l'Environnement. C'est notamment le cas pour HCSM et HPTS++. Ces modèles proposent des structures d'automates parallèles hiérarchiques afin de représenter les agencements possibles des évènements. Ils peuvent donc être au moins partiellement représenté graphiquement. Ces modèles ne proposent pas par défaut certains concepts liés aux scénarios comme les acteurs ou les actions. Ils ne proposent pas non plus de fonctionnalités pour le guidage des acteurs, la prise en compte de collaboration ou la modification de l'environnement. Ces fonctionnalités doivent être ajoutées spécifiquement dans les scripts liés aux états atomiques. Ces scripts font la liaison directe entre l'environnement virtuel et la spécification des scénarios. Il n'est pas proposé de niveaux d'abstractions intermédiaires. Ces modèles génériques demandent donc des compétences en programmation d'Environnements virtuels assez poussées pour pouvoir être utilisées, car ils demandent de travailler directement avec les composants de l'Environnement.

En plus de la modélisation du comportement d'objets (voir section 1.2), HCSM [Cremer et al., 1995] peut être utilisé pour décrire le scénario d'environnements virtuels. Il est le modèle type de cette famille. HPTS++ [Lamarche and Donikian, 2002] possède les mêmes propriétés que HCSM en tant que modèle de scénarios. Il propose cependant en plus une fonction de gestion interne des ressources. Ces dernières sont vues comme des sémaphores sur l'exécution possible de sous-scénarios. Si les ressources dans l'environnement sont modifiées, il faut répercuter cette modification dans la spécification des scénarios.

C Language for Object Relation Application (LORA)

LORA [Mollet and Arnaldi, 2006] et son extension LORA++ [Gerbaud et al., 2007] reposent sur un formalisme proche de Grafcet [DAVID, 1989] ainsi que des langages d'automates parallèles hiérarchiques comme HCSM et HPTS (voir Paragraphe B). LORA++ peut représenter des agencements complexes utilisant la concurrence, le parallélisme ou la séquentialité en utilisant des connecteurs (voir Figure 2.15) entre les différentes étapes et propose un système de gestion des ressources. Il existe différents types d'étapes: calcul, conditionnelle, consultation, sous-scénario et action. Les étapes de calcul interprètent des fonctions qui affectent des variables liées à l'automate. La consultation permet de lire l'état de l'environnement. La représentation des actions repose sur le modèle Objets-Relations de STORM [Mollet et al., 2007] ce qui lui offre notamment la possibilité de décrire des scénarios collaboratifs avec des acteurs réels et/ou virtuels. LORA++ n'a pas vocation à piloter l'environnement, mais peut déclencher les actions prévues à la place d'un acteur. Cette propriété permet d'adapter artificiellement le niveau de guidage des acteurs en proposant de déclencher des actions ou des séquences d'actions à la place d'un acteur afin de simplifier sa prise de décision. Il est donc possible de forcer le déclenchement des actions liées à l'environnement en définissant un acteur spécialisé. Enfin, les automates LORA++ sont paramétrables et offrent la possibilité d'utiliser des variables afin de les rendre réutilisables dans d'autres simulations. L'accès à LORA++ nécessite un peu d'apprentissages pour connaitre les différentes entités et des connaissances en informatique pour pouvoir utiliser toutes ses propriétés. Cependant son formalisme à base d'automate parallèle hiérarchique et sa représentation graphique en simplifient l'utilisation.

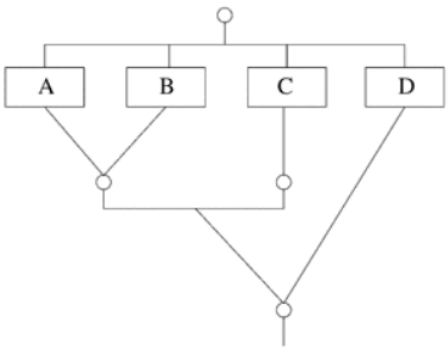


Figure 2.15 – Exemple d'agencement d'étapes en LORA : (((À OU B) ET C) OU D). Origine [Mollet and Arnaldi, 2006]

D HAVE

HAVE [Chevaillier et al., 2012] est utilisé en conjonction du modèle objets-relations VEAH (représentation de l'environnement, voir Section 1.2) dans le framework MASCARET [Buche et al., 2003] [Le Corre et al., 2014]. Il prend donc en compte ses possibilités de collaboration entre

acteurs réels et/ou virtuels. C'est une extension du diagramme d'activité utilisé dans UML. Il repose sur les mêmes composants de base. Il offre donc les mêmes possibilités d'agencement d'évènements. Les actions des acteurs sont considérées comme des activités atomiques. Les objets intervenants dans les actions sont identifiés par leur type. Ceci permet de généraliser lors de la spécification et de proposer une gestion des ressources. HAVE ne semble pas prévu pour pouvoir déclencher des évènements dans l'environnement. Une solution serait de créer un acteur "*Environnement*" qui déclencherait ces actions. Un système pré et postconditions lie les actions ou activités et l'environnement. Ce sont les acteurs qui vérifient la faisabilité d'une action et son bon déroulement.

E A behavior Language (ABL)

ABL [Mateas and Stern, 2002] est organisé comme une collection de comportements définis via un langage proche de Java ou C#. Ces comportements sont intégrés dans des éléments de scénario appelés beats. Les beats sont organisés par une machine à état. Chaque beat définit le comportement des acteurs en fonction de celui de l'utilisateur et les conditions de sorties vers d'autres Beats. La définition des beats demande beaucoup de développement. Tous les cas doivent être définis et le formalisme n'est utilisable que par des développeurs. De plus les beats sont très vite spécialisés et non réutilisables. Ce fonctionnement contraint à avoir toujours un unique utilisateur et un ou plusieurs acteurs virtuels. La gestion des ressources est définie au niveau de la description du comportement des acteurs. Le niveau de contrôle sur les acteurs virtuels est très fort et basé sur le comportement de l'utilisateur.

F IVE

IVE [Brom and Abonyi, 2006; Brom et al., 2007] utilise un modèle de scénarisation basé sur les réseaux de Petri [Petri, 1966]. Ceci permet une représentation graphique des scénarios et permet d'exprimer les séquences, le parallélisme et la concurrence. Le modèle utilise deux types de places (acteur ou précondition). Dans ces places, les jetons peuvent contenir des données liées respectivement aux acteurs (p. ex. le magicien est dans le pub), ou à des préconditions système (p. ex. le personnage veut acheter une poupée). Les données liées aux acteurs peuvent représenter les ressources auxquelles ils ont accès. Ceci permettrait d'ajouter des parties de scénario pour gérer l'accès aux ressources, même si ce n'est pas directement prévu dans le modèle. Les transitions du réseau représentent les actions. Elles sont déclenchées si toutes les places en amont de la transition possèdent un jeton et si les conditions sur les données d'un jeton, définies sur les arcs entrants, sont validées. Un jeton peut être produit sur une place si certaines conditions sont remplies. Par exemple, il est 16H, le théâtre arrive en ville : un jeton est produit sur la place "*théâtre*". Toutes les actions sont déclenchées par le scénario, pas par les acteurs. Un acteur ne fait que mettre un jeton sur une place précondition. Par exemple, un acteur veut "*Voler une poupée*", un jeton est produit sur la place associée et l'action voler est alors déclenchée. IVE ne présuppose par nombre ou de la nature des acteurs participants à la

simulation. L'architecture liant les acteurs, le moteur de scénarios et l'environnement impose un niveau de contrôle strict des actions des acteurs.

G Langages de Scénarios pour la Formation assistée par ordinateur

Les langages de scénarios pour la formation assistée par ordinateur comme IMS-LD [Koper and Olivier, 2003] ou PALO [Rodriguez-Artacho, 2002; Rodríguez-Artacho and Maíllo, 2004] sont des langages de scénarios prédéfinis qui permettent de décrire des cursus de formation complexes dans le cadre de formations proposées par des plateformes d'e-learning comme MOODLE[Dougiamas and Taylor, 2003]. Ils proposent certaines propriétés communes avec les modèles de Scénarios prédéfinis pour Environnements virtuels comme la capacité de modéliser des séquencements plus ou moins complexes ou la collaboration entre plusieurs apprenants et/ou formateurs. Cependant, ils considèrent les activités comme atomiques : elles ne sont pas décrites par la représentation des scénarios possibles, au-delà de leurs entrées ou sorties [Marion et al., 2009]. Le niveau de guidage offert dépend donc de la granularité utilisée pour définir les activités.

2.2.2 Modèles à scénario émergent

Les modèles à Scénarios émergents utilisent les comportements plus ou moins contraints des acteurs. Le scénario "*émerge*" alors au cours de la simulation. Ces modèles ne proposent pas de solution pour forcer l'agencement des actions. Les solutions utilisées sont de deux types. Soit le comportement de l'acteur est prédéfini pour réagir à un état spécifique de l'environnement virtuel (p. ex. la porte est fermée, il faut utiliser la clé pour la déverrouiller). Soit certains objectifs lui sont donnés (p. ex. aller dans la pièce suivante) et il prévoit lui même ses actions. La deuxième solution repose sur des techniques de planification d'action (par exemple HSP :[Bonet and Geffner, 2001]), développées du domaine des intelligences artificielles.

A Thatrix

Theatrix [Paiva et al., 2001] utilise trois modes de fonctionnement, un mode préparation (backstage), le mode jeu (on-stage) et un mode lecture (audience). Les trois modes de fonctionnement sont directement liés à l'applicatif. Lors de la préparation, les personnages sont répartis dans des scènes ainsi que des objets. Cette répartition est réalisée par un des utilisateurs et définit une situation initiale. En mode jeu, les utilisateurs contrôlent chacun un personnage. Les personnages restants sont contrôlés par le système. Les actions des acteurs sont définies par un ensemble statique d'actions de base accessibles à tous (p. ex. prendre, poser, utiliser). Le scénario émerge du comportement des acteurs au cours de la session d'édition. Theatrix n'offre pas de guidage des acteurs autres que la situation initiale. L'agencement des actions n'est pas contrôlé par le système. Seules certaines relations de causalité sont représentées grâce à une gestion des ressources nécessaires pour certaines actions liées à l'utilisation des objets. Il n'est pas possible de modifier l'environnement en dehors de la volonté des acteurs.

B IDTension

IDTension [Szilas, 2003] fonctionne sur un système à base de règles qui définissent les actions possibles par rapport à l'état des acteurs. Les actions possèdent des paramètres qui sont des éléments narratifs comme des objectifs, des objets ou des acteurs. L'état des acteurs est défini par des éléments comme "*souhaite réaliser un objectif*", "*connais une information*" ou "*connais l'existence d'un obstacle*". Les règles définissent des conditions pour l'exécution d'une action (voir la figure 2.17). Le comportement des acteurs est guidé par leur personnalité (définie par des variables, par exemple non-violence=0.8), par les actions possibles en fonction du contexte et par leurs objectifs. Il est difficile de fournir un guidage fort à un acteur. Le travail de l'auteur est de mettre en place les composants narratifs. Le système fonctionne en trois modes : autonome (que des agents virtuels) et première personne (un utilisateur joue le protagoniste de l'histoire) et narration interactive (l'utilisateur et le système choisissent une action dans tous les possibles par tous les acteurs chacun leur tour). La formalisation des règles n'est pas adaptée à une représentation graphique. Les acteurs ne sont contraints, mais pas guidés. De ce fait, il n'y a pas de contrôle sur l'agencement des évènements. Les agencements complexes peuvent être exprimés si les règles sont très précises, mais cela risque d'être fastidieux. Enfin, un acteur "*Environnement*" peut être créé pour permettre les déclenchements d'actions hors du contrôle des acteurs "*personnages*". Mais il faut encore une fois bien définir les règles pour que ces actions aient lieu au bon moment et ce n'est pas une utilisation prévue du modèle. Enfin, le modèle ne propose pas explicitement de gestion des ressources.

C EmoEmma

EmoEmma [Cavazza et al., 2007] repose sur un ensemble d'action associé à chaque acteur. Ces actions sont définies avec des préconditions sur l'état de l'environnement et des acteurs. Lorsqu'un acteur exécute une action, cette dernière influe sur ces états et modifie donc les actions accessibles à chaque acteur. Les acteurs choisissent la prochaine action à exécuter en utilisant la technique d'action planning HSP [Bonet and Geffner, 2001]. Les actions sont des scripts qui modifient l'état de l'environnement ou déclenchent des animations. Comme pour IDTension, le niveau de contrôle des acteurs est difficilement adaptable. La spécification explicite d'un agencement complexe d'action précise demande une description de contraintes complexe et fastidieuse. La gestion des ressources n'est pas non plus prévue explicitement par le modèle. Encore une fois, la spécification des règles n'est pas adaptée à une représentation graphique. Ceci risque de limiter l'utilisation du modèle sur des cas d'utilisation très complexes. Le modèle est prévu pour intégrer un unique utilisateur et plusieurs acteurs virtuels. Le modèle ne propose pas de déclenchement d'action hors du comportement des acteurs. Comme pour IDTension, un acteur "*Environnement*" peut répondre partiellement au problème en tenant compte des problèmes de contrôle.

D VRaptor

Dans VRaptor [Shawver, 1997] le scénario émerge du comportement des acteurs virtuels à partir de l'état de l'environnement et des actions de l'utilisateur. Ces comportements sont définis indépendamment par un langage de script spécialisé, proche de langages comme python [Lutz, 2010]. Leur spécification demande l'utilisation de concepts de très bas niveau comme les durées des actions ou les animations à déclencher. Un agencement complexe d'actions prenant en compte beaucoup d'acteurs ayant un comportement différent demandera énormément de travail. De plus il est difficile d'utiliser une représentation graphique de ces comportements. Chaque acteur virtuel prend en compte l'état de l'environnement virtuel indépendamment. La gestion des ressources est possible si elle est incorporée dans le comportement des acteurs virtuels. Le modèle semble pouvoir être utilisé dans un système comportant plusieurs utilisateurs. Ces derniers sont libres d'agir dans la limite des actions réalisables dans l'environnement. Le modèle ne propose pas d'autres niveaux de contrôle. Comme VRaptor s'intéresse au comportement indépendant des acteurs virtuels, le déclenchement d'actions hors de la volonté des acteurs doit encore une fois passer par un acteur "*Environnement*".

E SELDON

DIRECTOR est le moteur de scénario du modèle SELDON [Lanquepin et al., 2013] [Carpentier et al., 2013] [Barot, 2014][Carpentier, 2015]. SELDON ne propose qu'un niveau de guidage très libre. Il utilise des techniques de planification pour pousser l'environnement à atteindre un état cible à partir de l'état du monde, des activités à réaliser (par exemple éteindre un feu) et d'un ensemble de contraintes qui permet de relier les activités entre elles. Par exemple, il peut essayer de faire faire une erreur à un acteur virtuel en déclenchant un appel téléphonique stressant de son supérieur. Stressé, il aura alors plus tendance à faire des erreurs. Cependant, les évènements nécessaires ne sont pas toujours surs d'arriver. Il n'est donc pas possible de maitriser totalement le déroulement d'une simulation. Si DIRECTOR n'arrive pas à générer une situation, il demande à TAILOR de relâcher les contraintes. Les activités du domaine sont décrites dans un langage de qui les découpe en sous-activités jusqu'au niveau minimum, l'action. Cette description indique simplement quelles sont les actions à réaliser, sans contraindre complètement l'ordre. Il est possible d'intégrer dans l'Environnement au moins un utilisateur avec des acteurs virtuels. L'utilisation des différents langages de HUMANS-DL demande la maitrise de nombreux concepts et des connaissances en programmation logique. Les modèles utilisés sont très génériques et permettent de représenter des évènements ou des activités qui pourront être facilement réutilisés une fois définis. Une partie des modèles peut utiliser des représentations graphiques. C'est par exemple le cas de la description des activités. La gestion des ressources est déléguée au modèle d'environnement virtuel sous-jacent (c.f. Section 1.2.3).

2.3 Synthèse

Les domaines de la formation comme de la narration interactive possèdent tous deux des modèles à scénarios prédéfinis ou à scénarios émergents. Nous ne distinguons pas de raison évidente pour qu'une famille de modèles soit plus intéressante qu'une autre pour l'un ou l'autre des domaines. Du point de vue de nos critères, les modèles à Scénarios prédéfinis offrent bien plus de contrôle sur l'agencement des actions ainsi qu'une représentation graphique simplifiant leur utilisation. Les modèles génériques (HCSM et HPTS++) peuvent être adaptés à de nombreux cas d'utilisation, mais ils ne permettent pas de travailler directement à un niveau d'abstraction élevé. Ils ont donc été intégrés comme moteur sous-jacent d'autres modèles (par exemple, LORA s'exécute en HPTS++). LORA++ et HAVE demandent quelques ajustements pour pouvoir facilement interagir avec l'environnement en dehors des actions des acteurs et ne sont prévus que pour interagir avec un modèle d'environnement précis. La majorité des modèles de scénarios émergents ne propose pas de moyens pour impacter l'environnement en dehors des actions des acteurs. Puisque l'ensemble des scénarios possibles n'a pas de définition explicite, le formalisme utilisé dépend de la spécification des actions, des objets et des comportements des acteurs. La gestion des ressources ne peut être prise en compte que si le modèle d'environnement virtuel ou le modèle de définition des acteurs le propose. Enfin, la majeure partie des solutions repose sur un ou plusieurs langages textuels de descriptions de préconditions ou de comportements ce qui les rend plus difficiles à appréhender, principalement car il est difficile de fournir une représentation graphique claire.

Le tableau 2.5 synthétise les propriétés des différents modèles étudiés dans cette section.

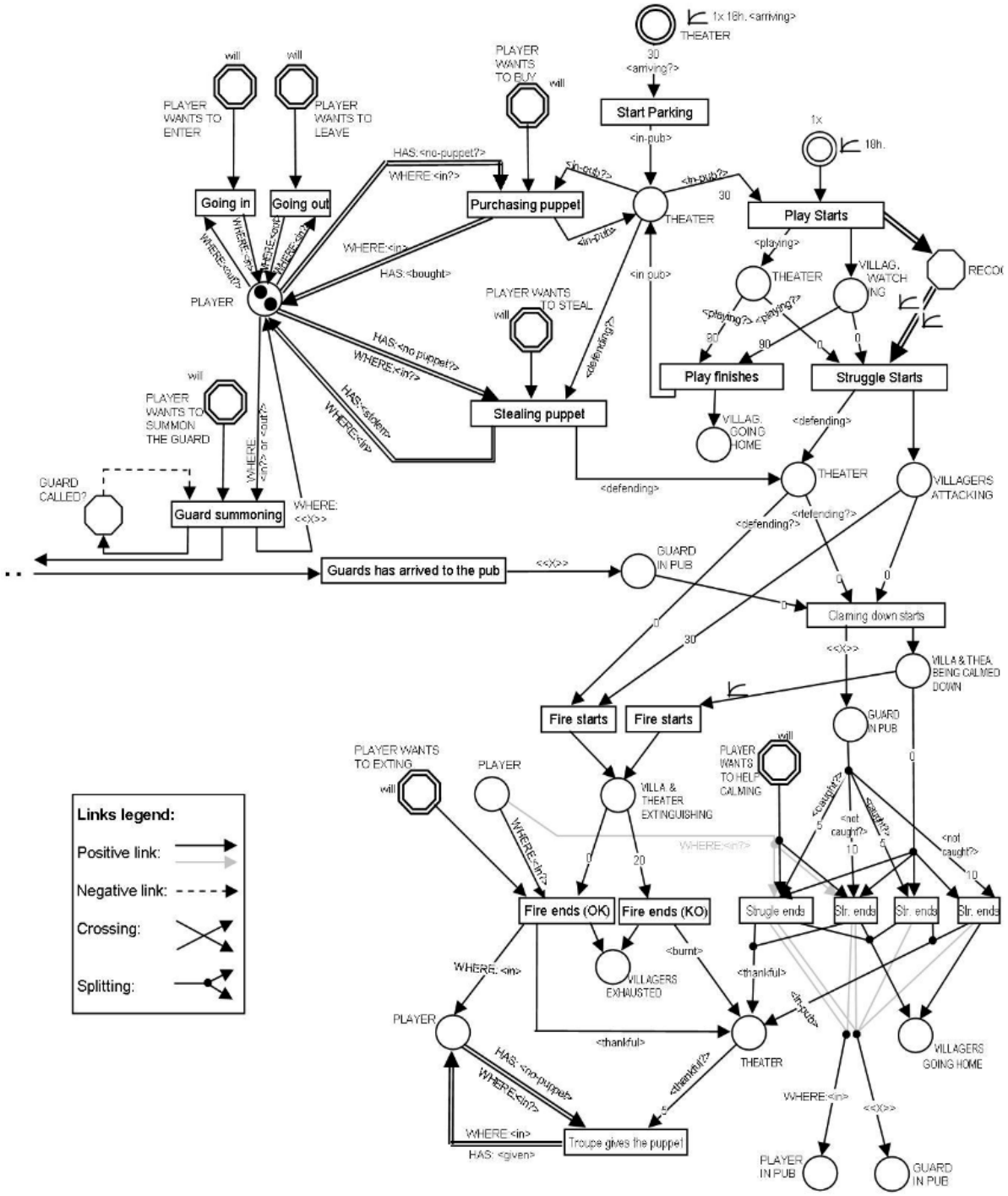


Figure 2.16 – Partie d'une spécification de scénarios avec le modèle IVE. Origine [Brom and Abonyi, 2006]

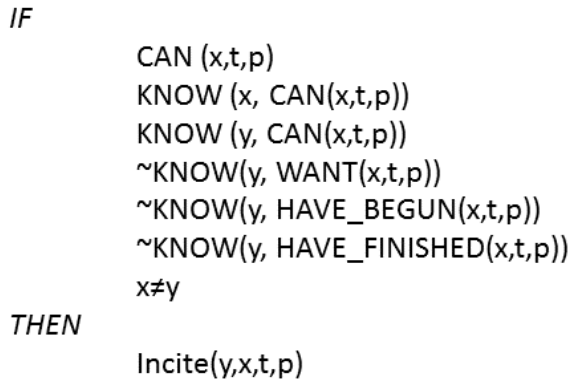


Figure 2.17 – Exemple de règle dans IDTension. Ici, l'action "*incite*" peut-être utilisée si les préconditions définies en amont sont satisfaite. x et y sont des acteurs, t es une action et p un paramètre optionnel. Origine [Szilas, 2003]

Solution	Utilisation	Niveau(x) de Contrôle	Agencement	Ressources	Actif	Collaboration	Graphique
modèle à scénarios prédéfinis							
HCSM	Générique	si prévu dans les scripts	séquentiel, parallèle, Concurrent	si prévu dans les scripts	si prévu dans les scripts	si prévu dans les scripts	oui
HPTS++	Générique	si prévu dans les scripts	séquentiel, parallèle, Concurrent	ressources symboliques intégrées (Sémaphores)	si prévu dans les scripts	si prévu dans les scripts	oui
Story Nets	Formation	Choix limités	Séquence, Concurrent	non	Pendant les Transitions	Un Utilisateur et plusieurs Agents	partiellement Possible
LORA++	Formation	Actions et Séquences d'actions prédéfinies	Séquentiel, parallèle, Concurrent	Oui	via un rôle "Environnement"	Utilisateurs et agents virtuels	oui
HAVE	Formation	Séquences d'actions prédéfinies	séquentiel, parallèle Concurrent	oui	via un acteur "Environnement"	Utilisateurs et Agents virtuels	oui
ABL	Narration	Comportements prévus	Séquentiel, Concurrent	non	non	un utilisateur et plusieurs Agents	partiellement Possible
IVE	Narration	Volonté d'action des acteurs	séquentiel, parallèle, concurrent	indirectement par les données des jetons	Obligatoire	utilisateurs et agents virtuels	oui
Modèles à scénario émergent							
Theatrix	Narration	actions libres, Guidées par objectifs	non Contrôlé	Inventaire d'objet acquis par l'acteur	non	Utilisateurs. et Agents virtuels	non, mode éditeur
IDTension	Narration	Actions libres avec préconditions, Guidées par objectifs	non Contrôlé	État des acteurs, connaissances	non	Un Utilisateur et plusieurs Agents	non
EmoEmma	Narration	Actions libres avec pré et post conditions	non Contrôlé	État des acteurs	non	Un Utilisateur et plusieurs Agents	non
VRaptor	Formation	Actions libres, État de l'Environnement	non Contrôlé	Environnement	non	au moins un Utilisateur et plusieurs Agents	non
SELDON	Formation	Actions libres, tâches à Réaliser	Tâches partiellement ordonnées, liées par contraintes sur l'env.	oui	oui, non contrôlé	au moins un Utilisateur et plusieurs Agents	partiellement Possible

Table 2.5 – Synthèse des modèles de scénarios existants.

3 Attribution des actions aux acteurs

Selon Gerbaud et al [Gerbaud et al., 2009], une fois l'ensemble des scénarios spécifiés, un système de répartition des actions entre les acteurs permet de représenter l'asymétrie dans leurs activités. Par exemple, lors d'une procédure chirurgicale, pendant que le chirurgien opère, l'infirmière instrumentiste nettoie et réarrange les instruments déjà utilisés. Il s'agit de **fournir à chaque acteur** (réel ou virtuel) **un ensemble d'actions** qu'il peut où doit exécuter en fonction de l'implication qu'il peut ou doit avoir dans la simulation. Dans le domaine des environnements virtuels, la solution commune est de représenter la notion de **rôle**. Le rôle d'un acteur est un ensemble de données utilisées par un moteur pour définir ce que cet acteur peut ou doit faire. Les mécanismes de ce moteur et la structure des données du rôle sont définis par un **modèle de rôle**.

3.1 Rôles des acteurs en réalité virtuelle

La *théorie des rôles* est un domaine des sciences humaines et sociales qui s'intéresse à la manière dont sont organisés des groupes de personnes en fonction de leurs membres et de leurs activités. Il existe de nombreuses définitions du rôle dans ce domaine [Biddle, 1986]. La définition de Biddle et Thomas [Biddle and Thomas, 1966] est proche de l'utilisation faite en réalité virtuelle : "*Ensemble de concepts pris en compte par quelqu'un au sujet du comportement d'une personne ou d'une position sociale*". Ces concepts peuvent être de **différentes natures** comme les compétences, les droits, ou les préférences de la personne. Ils sont utilisés pour estimer ce que la personne concernée peut, sait ou doit faire et quel peut-être le comportement attendu de cette personne. En réalité virtuelle, ces informations sont liées aux données définissant l'acteur.

Un élément important de la théorie des rôles [Biddle and Thomas, 1966] est qu'un rôle peut **évoluer au cours du temps**. Cette évolution peut être **due à des évènements** dans l'environnement social ou physique de l'acteur. Nous appelons "*environnement social d'un acteur*" le groupe d'acteurs dans lequel il est intégré au cours de la simulation. Dans un système de réalité virtuelle, il s'agit donc de l'ensemble des acteurs réels ou virtuels évoluant dans l'environnement virtuel. Ces acteurs sont liés entre eux par des faits (p. ex. Georges est le frère de Lucien) et des règles (p. ex. le plus expérimenté du groupe donne les ordres) ou des tâches à réaliser (p. ex. réparer une machine). Les évènements sociaux sont des changements dans ces faits ou ces règles qui peuvent impacter la simulation. La modélisation des rôles des acteurs et de leur évolution nécessite d'être capable d'exprimer **la structure et les mécaniques internes liées à un groupe d'acteurs**. Par exemple, un acteur peut acquérir un nouveau droit parceque son supérieur le lui a donné par un ordre (une action). Enfin l'évolution du rôle d'un acteur peut être liée à un besoin spécifique de la simulation et doit donc pouvoir être **déclenchée par un composant tiers** comme le moteur de scénarios. Par exemple, dans le cadre d'une formation, une tâche peut être attribuée à un utilisateur en particulier, car elle est prévue dans son cursus de formation alors qu'en réalité elle est réalisable par n'importe quel membre de l'équipe.

En tant que composant de l'environnement virtuel, le moteur de rôle va nécessiter sa propre spécification de comportements. Tout comme la spécification des scénarios, cette tâche peut demander un travail non négligeable et l'intervention d'expert connaissant le domaine de travail. La réutilisation de tout ou partie de la spécification d'un système de rôle va permettre de gagner en productivité. Il est raisonnable d'envisager qu'un même groupe d'acteurs puisse être impliqué dans différentes simulations. Par exemple, une procédure de chirurgie nécessitera toujours certains acteurs comme le chirurgien, l'interne, l'infirmière instrumentiste et la circulante. Ces acteurs ont des rôles spécifiques, quelle que soit l'opération réalisée. L'organisation et le fonctionnement de l'équipe peuvent être en partie similaires d'une simulation à l'autre. Les spécifications des rôles et des règles d'évolutions doivent pouvoir être **réutilisées et adaptées facilement à d'autres cas d'utilisation**. Enfin, afin de pouvoir réutiliser ces spécifications, les modèles liés aux rôles des acteurs doivent dépendre le moins possible des autres composants de l'environnement virtuel comme le moteur de scénarios, les actions ou les objets puisque ces derniers peuvent changer d'une simulation à l'autre.

3.2 Modèles de rôles

Dans cette section, nous nous intéressons à plusieurs domaines de solutions. Comme précédemment pour les modèles de scénarios, nous étudions les Environnements virtuels collaboratifs de Formation et les systèmes de storytelling. Nous étudions en plus le domaine des "*Systèmes Multi-Agents centrés organisation*". Ils nous intéressent ici, car ils utilisent des concepts proches des rôles et des équipes [Ferber et al., 2004] comme nous avons pu aborder dans la section précédente. De plus certains travaux en Environnements virtuels collaboratifs reposent sur des concepts issus des systèmes multi agents centrés organisation.

Sur la base de la section précédente, nous nous intéressons aux propriétés suivantes :

- Quels sont les ***Concepts liés à l'Acteur*** pris en compte dans la modélisation de son rôle ? Le modèle est-il capable d'en tirer ce que l'acteur peut, sait ou doit faire ?
- Le rôle d'un acteur peut-il ***évoluer au cours de la simulation*** pour s'adapter à des changements de situations sociales ou physiques ? Notamment les règles d'un groupe.
- Le modèle permet-il à un Rôle ou à un fonctionnement d'équipe d'être ***réutilisé*** ? Le fonctionnement du modèle est-il intégré ou indépendant des autres composants du système ? Est-il spécialisé pour un domaine précis ?

Nous étudions les solutions existantes suivant les angles suivant *la nature de ces données* (Section 3.2.1) et *les capacités d'évolution* des modèles (Section 3.2.2).

3.2.1 Nature des données

Nous avons constaté que deux approches distinctes étaient utilisées dans la représentation du rôle: les approches *orientées scénario* (Paragraphe A) et les approches *orientées acteur*

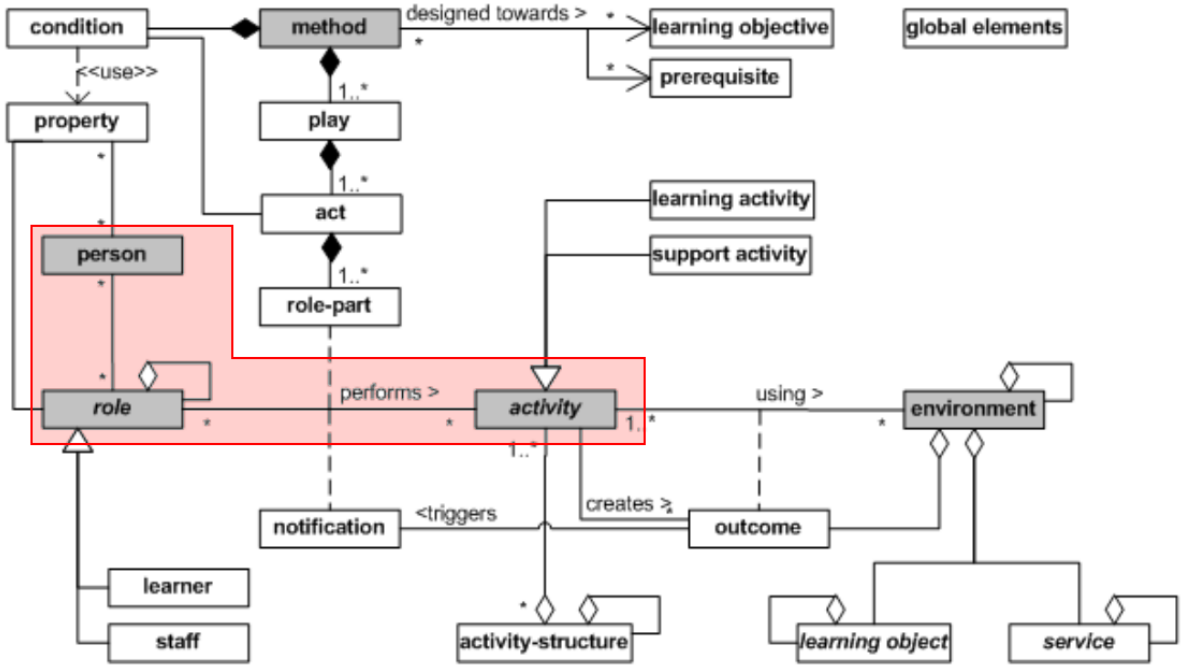


Figure 2.18 – Structure du modèle de rôle d'IMS-LD (zone rouge). Origine [Koper and Olivier, 2003]. Ce modèle est proche de ceux utilisés dans d'autres modèles comme Mascaret [Chevaillier et al., 2012] ou AALAADIN [Ferber and Gutknecht, 2000].

(Paragraphe B). Ces approches définissent la nature des données utilisées dans le rôle d'un acteur.

A Orientée scénarios

L'approche la plus courante est celle que nous appelons "*orientée scénarios*". Les données du rôle sont issues de celles définies dans la spécification des scénarios. Par exemple, LORA++ [Gerbaud et al., 2007] et Mascaret [Chevaillier et al., 2012] utilisent les actions directement définies dans le scénario. Un acteur peut réaliser une action à un instant donné si elle est réalisable dans le scénario et que le rôle de l'acteur la contient. Il s'agit généralement de l'approche la plus directe et la plus simple, également utilisée dans IMS-LD [Koper and Olivier, 2003]. Dans ABL [Mateas and Stern, 2002] les comportements des acteurs et leurs conditions de déclenchement sont définis à chaque étape du scénario. Dans TEATRIX [Paiva et al., 2001] le rôle de chaque acteur est défini par des objectifs. Ces objectifs sont prédéfinis en cas d'un acteur virtuel et défini de manière tacite pour les acteurs réels. L'acteur doit ensuite prendre la décision de l'action qu'il va entreprendre. Le défaut de cette approche est que la définition du rôle est dépendante du scénario. Il est donc difficile, voire impossible, de réutiliser les rôles et les mécanismes

associés avec un autre scénario. Son principal avantage est que son utilisation est intuitive : un acteur n'accède qu'aux éléments indiqués dans son rôle. Cassiopeia [Collinot and Drogoul, 1998] propose une version étendue ou le rôle d'un acteur définit l'ensemble des actions disponibles et une méthodologie de choix dans cet ensemble.

B Orientée acteur

L'approche orientée acteur consiste à utiliser des données liées à l'acteur pour calculer un ensemble d'actions disponibles. Dans EmoEmma [Cavazza et al., 2007] ces données décrivent les relations sociales entre les acteurs ainsi que leur état d'esprit ou leur point de vue sur un sujet donné. Les actions sont définies avec un ensemble de préconditions qui les rendent ou non utilisables en fonction de l'état des acteurs. Par exemple, un acteur ne peut s'en prendre à un autre que s'il est en colère contre ce dernier. Dans IDTension [Szilas, 2003], un acteur possède un ensemble d'objectifs, des connaissances au sujet du monde et des propriétés liées à son caractère (par exemple sa propension à respecter la loi). Ces données sont utilisées pour définir les actions qui lui sont accessibles à un instant donné puis pour choisir laquelle doit être réalisée. L'avantage principal de cette solution réside dans le fait que le rôle d'un acteur n'est pas lié directement au scénario. Ce rôle est donc réutilisable dans d'autres simulations. Son défaut est de ne pas être directe facile d'accès. Le modèle IODA [Kubera et al., 2011] se base sur des données associées à l'acteur et définies par son type. Par exemple un acteur peut être de type chirurgien ou un interne. Les données accessibles sont donc dépendantes du type. Dans MOISE [Hannoun et al., 1999, 2000], les actions sont encapsulées dans un système de pré/postconditions. Ces conditions sont spécifiques à la simulation, mais les actions restent génériques.

C Sans orientation

Certains modèles ne tiennent pas compte de la nature des données liées au rôle. C'est notamment le cas du modèle AALAADIN [Ferber and Gutknecht, 2000]. Il s'agit d'un modèle organisationnel qui ne présuppose pas de la nature des données liées à un rôle et de l'utilisation qui en est faite. En fonction de ces données, cette utilisation peut impliquer, ou non, des dépendances avec d'autres composants. Elle a l'avantage d'être utilisable dans un plus grand nombre de cas.

3.2.2 Mécanismes d'évolution

L'évolution du rôle répond aux évènements pouvant changer l'implication des acteurs dans la simulation. Par exemple un acteur peut obtenir le droit de réaliser une action, car son supérieur hiérarchique le lui a donné. Les solutions existantes ont différents moyens de gérer cette évolution. Certaines ne proposent pas cette fonctionnalité (LORA++ [Gerbaud et al., 2007] et TEATRIX [Paiva et al., 2001]). D'autres l'intègrent à des composants déjà existants du système (voir Paragraphes A et B). Enfin, certains proposent des modèles indépendants

permettant de décrire les règles d'évolutions liées à l'organisation des acteurs et à l'état de la simulation (voir Paragraphe C).

A Pilotée par le scénario

ABL [Mateas and Stern, 2002] découpe la spécification des scénarios en scènes qui peuvent avoir plusieurs issues différentes. Chaque scène définit le comportement associé de chaque acteur virtuel en fonction des comportements potentiels de l'utilisateur. En fonction de l'issue d'une scène, une nouvelle scène est choisie et de nouveaux comportements sont assignés aux acteurs virtuels. Ceci demande beaucoup de travail lors de la spécification du scénario pour prendre en compte tous les cas possibles. De plus, il est impossible de fournir des simulations alternatives simplement en adaptant le rôle des acteurs, car ils sont très fortement liés au scénario.

B Pilotée par les actions

Lorsqu'une action modifie l'état d'un acteur, son rôle peut être modifié. C'est le principe utilisé dans EmoEmma [Cavazza et al., 2007] et dans IDTension [Szilas, 2003]. Les préconditions filtrent les actions et ces dernières modifient l'état des acteurs. Par exemple, seul le chef d'un groupe a accès aux actions permettant de donner des ordres, car il répond aux préconditions associées. Il peut, par exemple, utiliser une action pour assigner une tâche à un des autres acteurs. Cette action modifie le rôle de l'acteur ciblé en lui ajoutant la responsabilité de cette tâche. Il accède alors à un nouveau panel d'actions. Cette méthode lie très fortement le modèle de rôle aux actions. Il faut donc redéfinir de nouvelles actions pour chaque simulation. L'encapsulation des actions dans un système de pré/post conditions proposé par IODA [Kubera et al., 2011] résout le problème des actions non-génériques. Dans IMS-LD [Koper and Olivier, 2003] intègre l'évolution des rôles en tant que conséquence de la résolution d'une action. La Figure 2.18 présente ce modèle : les rôles sont reliés aux actions (activités). Un acteur (personne) peut avoir plusieurs rôles.

C Modèles d'équipes

La modélisation des équipes permet de définir des règles pour l'accès aux actions. Cette approche est utilisée dans les systèmes multi-agents AALAADIN [Ferber and Gutknecht, 2000], Cassiopeia [Collinot and Drogoul, 1998] ou MOISE [Hannoun et al., 1999, 2000] et dans le framework pour environnement collaboratif Mascaret [Chevaillier et al., 2012] (qui s'inspire du modèle de système multi-agents Centré Organisation Voyelle [Demazeau, 1995]). Une équipe est un groupe d'acteur. Dans une équipe un acteur est associé à une ou plusieurs positions. À tout moment, un acteur peut demander à entrer ou sortir d'une position. Des préconditions sur l'état de l'acteur sont définies sur l'accès aux positions. Si la demande est accordée, alors le rôle est modifié en fonction. Par exemple, un acteur peut demander à devenir le chef d'une équipe. Un test est alors fait pour vérifier qu'il possède les compétences requises. Si c'est le cas, son rôle est modifié pour prendre en compte cette nouvelle position. Cette solution permet d'abstraire l'attribution des

actions du scénario afin d'offrir plus de possibilités de réutilisation. Les systèmes multi-agent, comme Cassiopeia [Collinot and Drogoul, 1998], proposent en plus de modéliser les règles de l'équipe au même titre que les comportements des agents. Une équipe peut donc d'elle-même modifier le rôle d'un acteur si nécessaire.

3.3 Synthèse

Les modèles de rôle orientés scénario utilise des mécaniques simples qui permettent d'attribuer directement les actions aux acteurs. Ils sont cependant non-génériques. Un exemple de réutilisation simple serait, la formation à deux procédures différentes avec une même équipe d'acteurs. Les modèles orientés acteur vont répondre plus facilement à ce type de problème. Les mécaniques d'utilisation des données liées à l'acteur servent de filtre d'action entre le scénario et l'acteur. Il est donc possible de changer l'un sans impacter le second. De plus certains permettent de prendre en compte les informations comme les compétences, les droits ou les préférences des acteurs.

Les modèles qui proposent des mécaniques d'évolution des rôles ne se sont intéressés qu'un une possibilité parmi l'évolution pilotée par les actions, le scénario ou un modèle d'équipe. Pourtant, ces solutions ne semblent pas incompatibles entre elles et semblent répondre à des besoins différents. Il est raisonnable d'envisager d'utiliser un modèle d'équipe pour représenter les mécaniques sociales, l'effet des actions sur l'état des acteurs et des possibles modifications par le scénario pour des besoins spécifiques à l'utilisation de l'environnement virtuel.

À notre connaissance il n'existe pas de modèle qui propose une orientation acteur et l'évolution pilotée par un modèle d'équipe. Le tableau 2.6 propose une synthèse des modèles existants et étudiés dans cette section.

4 Conclusion

Dans ce chapitre nous traitons de l'environnement virtuel et des entités qui le composent, des scénarios et de leurs relations avec les acteurs. Les frameworks existants reposent sur différents modèles pour définir ces différents éléments. Par exemple, en formation, c'est le cas des plateformes GVT [Gerbaud et al., 2008] , MASCARET [Chevaillier et al., 2012] ou HUMANS [Lanquepin et al., 2013]. De plus, ces frameworks répondent également à d'autres problématiques, que nous n'avons pas traité ici, comme par exemple la pédagogie.

Réaliser des environnements virtuels en utilisant ces frameworks demande la maitrise de chacun des modèles qui les composent et des langages associés. Il serait intéressant d'unifier certains modèles utilisés, pouvant répondre à des besoins proches. Ceci permettrait de simplifier le travail des développeurs, principalement grâce à une économie de concepts. Ainsi, les développeurs d'une équipe sont alors moins spécialisés, plus rapidement opérationnels (temps de formation aux technologies réduit) et plus expérimentes dans un même modèle utilisé plus fréquemment.

Dans la littérature, de nombreux travaux ne font pas la distinction entre le comportement des entités dans l'environnement et les scénarios possibles. C'est très clairement le cas dans les modèles à scénarios émergents comme Theatrix [Paiva et al., 2001] ou VRaptor[Shawver, 1997]. Ce point de vue est soutenu par le fait que certains modèles, tels que HCSM [Cremer et al., 1995] ou HPTS++ [Lamarche and Donikian, 2002], peuvent être utilisés aussi bien pour décrire l'ensemble des scénarios possibles pour un environnement virtuel que le comportement des objets ou d'autres entités de l'environnement. Cependant, il leur manque un certain nombre de propriétés pour leur permettre d'être vus comme des langages de scénarios complets (c.f. Tableau 2.5).

Solution	Orientation du Modèle	Mécanique d'Évolution	Nature des données	Réutilisation	Dépendance
LORA++	Orienté Scénario (Action)	Aucune	Actions	Identifiants des rôles	Scénario
MASCARET	Orienté Scénario (Action)	Modèles d'équipe	Actions	Équipes	Scénario
IMS-LD	Orienté Scénario (Action)	Pilotée par les Actions	Actions	Identifiants des rôles	Scénario
ABL	Orienté Scénario (Action)	Pilotée par le Scénario	Actions	rien	Scénario
TEATRIX	Orienté Scénario (Objectifs)	Aucune	Objectifs	Objectifs	aucune
EmoEmma	Orienté Acteur	Pilotée par les Actions	État émotionnel de l'acteur	Définition d'état mental	Actions
IDTension	Orienté Acteur	Pilotée par les Actions	Connaissances et Caractère	Caractère des acteurs	aucune
CASIOPEIA	Orienté Scénario + Comportement	Modèle d'équipe	Actions	Modèle d'équipe	aucune
AALAADIN	-	Modèle d'équipe	Actions	Modèle d'équipe	-
MOISE	-	Groupes/Sociétés	Actions/Objectifs	Structure de groupe, Rôles	Actions
IODA	-	Pré/Post conditions	Données de l'acteur, Actions	Actions	Actions

Table 2.6 – Synthèse des différents modèles de rôles étudiés.

#SEVEN : Moteur de scénarios pour environnements virtuels

3

Dans ce manuscrit, nous proposons d'utiliser un modèle commun pour décrire l'ensemble des scénarios possibles, mais également les comportements d'entités dans l'environnement. Ainsi, les compétences requises par un développeur pour spécifier les comportements possibles des entités de l'environnement sont les mêmes que pour la spécification de l'ensemble des scénarios au niveau d'abstraction près. Les développeurs d'environnements virtuels sont alors moins spécialisés, ce qui peut offrir un gain de productivité non négligeable lors de la réalisation d'environnements complexes. Nous faisons l'hypothèse que décrire l'ensemble des comportements possibles d'une entité est équivalent à réaliser la spécification des scénarios locale à cette entité. Dans ce Chapitre, nous proposons donc un modèle qui ne fait pas d'hypothèse sur la nature de l'environnement dans lequel il est utilisé.

Un modèle de scénarios définit un formalisme et une organisation de données permettant de décrire des scénarios, ainsi que les mécanismes qui permettent leurs utilisations dans un système. Dans le Chapitre 2, nous avons étudié les travaux existants au regard d'un ensemble de propriétés que nous jugeons nécessaires à un modèle de scénarios. Il doit permettre

1. De décrire des agencements complexes (temporels et causaux) d'activités.
2. De prendre en compte les ressources disponibles pour adapter l'état du moteur ou le synchroniser avec les autres composants du système (p. ex. Acteurs, pédagogie, objets).
3. De modifier l'environnement virtuel en dehors des actions des acteurs.
4. De décrire des scénarios, quel que soit le nombre ou la nature des acteurs impliqués (réels ou virtuels).
5. D'exprimer différents niveaux de contrôle des acteurs.
6. Une spécification et une utilisation simple notamment par une représentation graphique.

Le fonctionnement de ce modèle passe par un moteur, exécutant une spécification des scénarios réalisée dans un langage spécialisé. Ce langage permet une spécification, par un spécialiste, d'un ensemble de scénarios possibles, mais peut également être utilisé par des outils de génération automatique comme proposée dans le projet S3PM (c.f. Chapitre 1). Chaque

scénario décrit un agencement (temporel et causal) précis des actions qui doivent être réalisées. Chaque combinaison d'alternatives ou d'inversion d'ordre d'action possible est définie par la spécification de l'ensemble des scénarios.

Pour nous, la spécification, par le spécialiste, des scénarios se rapproche du développement logiciel classique. Un des aspects importants des langages de programmation actuels est l'intérêt particulier qu'ils accordent à la productivité. Ils permettent la réutilisation, totale ou partielle, de composants et offrent ainsi la possibilité de tirer profit du travail déjà réalisé. Ceci est obtenu grâce à des concepts tels que les interfaces, les surcharges et l'héritage ainsi que par l'utilisation de bibliothèques externes. L'un des autres aspects importants de ces langages est le confort d'utilisation qu'ils offrent : une grande partie des concepts avancés ne sont à utiliser que lorsqu'ils sont nécessaires. Ainsi, un programme simple et séquentiel n'utilisera que des aspects basiques du langage comme la séquentialité ou les fonctions. Un logiciel offrant beaucoup de fonctionnalités complexes utilisera des concepts plus avancés comme l'héritage de type, la surcharge des opérateurs ou la gestion des accès concurrents aux ressources. La réutilisation et l'adaptation des composants permettent à ces langages de couvrir un ensemble de domaine et de cas d'application très large.

Dans ce Chapitre, nous présentons une contribution : #SEVEN ("*Scenarios Engine for Virtual Environments*"), qui offre les fonctionnalités nécessaires à la spécification et l'exécution de scénarios pour environnements virtuels. Le modèle #SEVEN décrit un langage permettant la spécification d'ensembles de scénarios ainsi que les mécaniques qui permettent leur exécution au travers d'un moteur. Dans la suite de ce manuscrit, nous faisons référence à ces éléments comme "modèle", "langage" et "moteur" #SEVEN. Nous mettons l'accent sur la productivité en permettant une utilisation adaptée (et adaptable) aux besoins des développeurs d'environnements virtuels. La Section 1 présente #SEVEN, son fonctionnement et son intégration dans un environnement. La Section 2 décrit en détail le modèle responsable de l'agencement temporel et causal des actions. La Section 3 ajoute des outils utilisés par #SEVEN pour étendre la spécification des scénarios.

1 Le Moteur #SEVEN dans son environnement

Un moteur de scénarios est une entité intégrée dans un environnement (virtuel ou non). Les échanges entre le moteur et l'environnement sont réalisés au travers des entrées et sorties du moteur. Les entrées lui fournissent des informations sur l'état de l'environnement. Les sorties lui permettent d'agir sur l'environnement.

Un modèle d'environnements virtuels utilise des familles d'entités pour représenter par exemple les objets, les acteurs ou les actions. Ces familles sont différentes d'un modèle à un autre. Le modèle de scénarios #SEVEN ne présuppose d'aucune propriété ou fonctionnalité du modèle d'environnement avec lequel il est utilisé. Il peut donc être utilisé avec n'importe quel environnement virtuel, mais peut également être intégré dans des environnements d'autres natures. Par exemple, il peut piloter un système de domotique en exploitant les sondes et les

contrôleurs du bâtiment.

Son fonctionnement repose sur un ensemble de classes d'entités, au sens de la programmation orientée objet, spécialisées pour une utilisation dans un environnement de nature précise. Comme en programmation, #SEVEN utilise des bibliothèques définissant ces classes pour permettre son intégration dans un environnement donné. Ainsi une bibliothèque offre la possibilité d'interagir avec un modèle d'environnement virtuel précis et une autre avec un système de domotique.

La "*spécification des scénarios*" est externe au moteur. Elle décrit les des instances nécessaires de ces classes ainsi que leur organisation pour définir le comportement du moteur dans l'environnement. Cette spécification est chargée à l'initialisation du moteur pour construire une "*représentation interne*" de l'ensemble des scénarios possibles. Cette représentation est ensuite maintenue par le moteur en fonction du temps et des informations issues de l'environnement par les entrées. Le changement d'état de la représentation induit des actions de la part du moteur par l'intermédiaire de ses sorties. La représentation interne est consultable par d'autres entités présentes dans l'environnement (par exemple les Acteurs dans un environnement virtuel).

Pour fonctionner, un moteur #SEVEN utilise

- Un "*contexte externe*", responsable des entrées/sorties, fait l'interface avec l'environnement,
- une "*spécification des scénarios*" réalisée dans le langage #SEVEN qui décrit l'état initial de la représentation interne (i.e l'ensemble des scénarios possibles).

La Figure 3.1 présente l'intégration générale d'un moteur #SEVEN dans un environnement. Le contexte externe permet les échanges entre l'environnement et la représentation interne du moteur.

Une classe de ***contexte externe*** est un élément central d'une bibliothèque #SEVEN. Elle fournit les entrées et sorties faisant l'interface avec un type précis d'environnement. Les entrées permettent d'accéder à toutes les entités de l'environnement pour connaitre leur état courant. Les sorties permettent d'envoyer des commandes à ces entités pour contrôler leur comportement. Si l'environnement est un environnement 3D, le contexte externe va fournir les méthodes permettant de connaitre la position d'un objet dans l'espace, mais également la possibilité de déplacer cet objet s'il est mobile. Si l'environnement est le monde réel, le contexte externe va fournir les informations obtenues par des sondes ou des capteurs ainsi que la possibilité de déclencher des actions par le biais de commandes. Par exemple, dans un système de domotique, le contexte externe offre un accès aux sondes qui informent sur l'état du bâtiment (p. ex. Température, humidité) et aux commandes des éléments électriques et mécaniques (p. ex. Radiateurs, volets, lumières). Pour utiliser des moteurs dans différents types d'environnements il faut donc fournir différents contextes externes correspondants. Nous prenons l'hypothèse d'un seul contexte externe par moteur.

Certaines entités faisant partie de l'environnement peuvent avoir besoin d'accéder à des informations liées à l'état courant de la représentation interne du moteur de scénarios. Dans

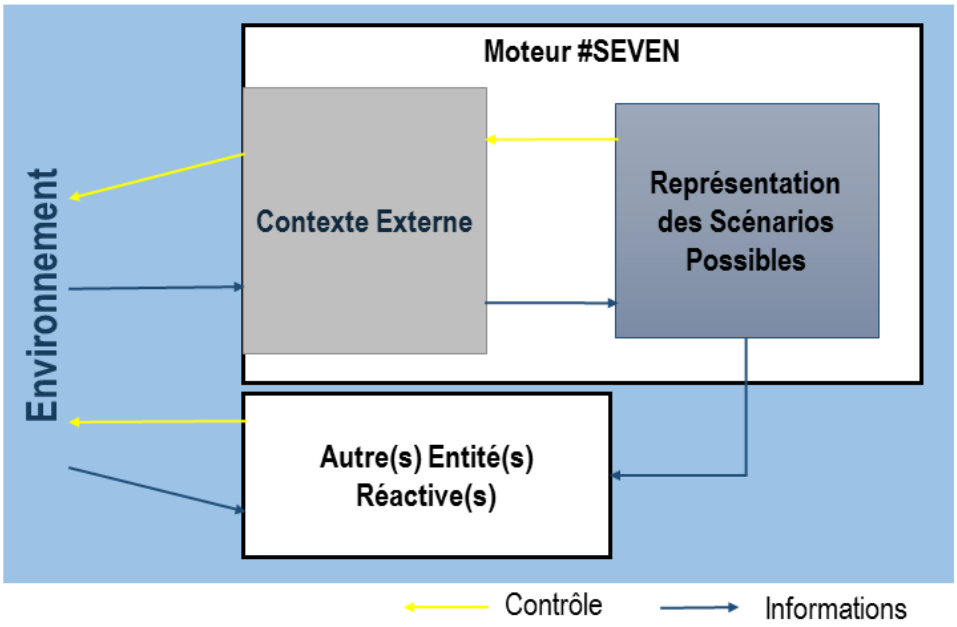


Figure 3.1 – Le moteur #SEVEN et son environnement.

le cas d'un environnement virtuel, un acteur peut vouloir connaitre quelles sont les prochaines actions qui feront avancer la simulation. Un moteur #SEVEN fournit l'interface permettant d'accéder à sa "*représentation interne*".

Un moteur #SEVEN fonctionne suivant le principe de la boucle de perception [Mallot, 1997] (ou boucle PDA - perception, décision, action). Il s'agit d'une représentation des interactions entre une entité ayant un comportement propre et son environnement. Elle est utilisée notamment en systèmes multi-agents et en robotique pour lier le comportement d'une entité réactive à son environnement (p. ex. [Reynolds, 1987], [van de Panne and Fiume, 1993], [Donikian, 2001; Lamarche and Donikian, 2002]). L'entité dispose d'un certain nombre de *senseurs* qui lui permettent d'acquérir des données sur son environnement. Ces données sont ensuite utilisées dans un processus interne de décision. Ce processus déclenche ensuite des actions, via des *effecteurs*, qui vont venir modifier l'environnement. Les données sont ensuite acquises de nouveau.

La Figure 3.2 montre le fonctionnement de la boucle :

1. L'entité réactive acquiert des données liées à son environnement.
2. Le système de décision du composant est mis à jour grâce aux données acquises.
3. Le composant réagit et modifie son environnement.

Un moteur #SEVEN utilise trois classes d'entités pour réaliser le fonctionnement de la boucle. Ces entités définissent également sa représentation interne.

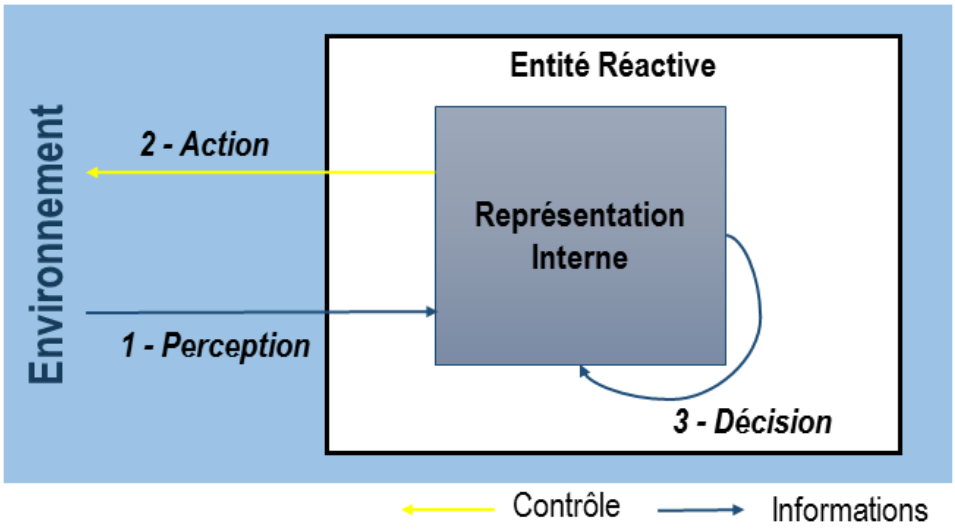


Figure 3.2 – Fonctionnement de la boucle de perception.

- "*les Senseurs*" vérifient une condition dans l'état de l'environnement - ***Perception*** ,
- "*Le Séquencement*" s'adapte aux informations perçues et prend les décisions d'actions - ***Décisions***,
- "*les effecteurs*" déclenchent des comportements dans l'environnement - ***Action***.

Ces classes d'entités sont définies dans les bibliothèques #SEVEN. Celles de senseurs et d'effecteurs sont spécialisées pour l'environnement cible au travers des propriétés et fonctionnalités fournies par le contexte externe. La Figure 3.3 représente le fonctionnement interne du moteur.

Les Senseurs attendent la réalisation de conditions dans l'environnement au travers du contexte externe. La Figure 3.4 présente leur fonctionnement. Un senseur permet au séquencement de savoir si une condition attendue est réalisée ou non. Le fonctionnement et la configuration d'un senseur dépendent de sa classe et sont liés à des entités de l'environnement. Les paramètres du constructeur définissent la condition attendue. Prenons par exemple une classe de senseurs qui permet d'utiliser une sonde thermique. Ses paramètres possibles sont :

- "*Probe*" - Une référence vers la sonde dans le contexte externe ,
- "*Value*" - un entier
- "*Operation*" - un test à réaliser parmi (Exemples donnés à titre indicatif)
  - EST_ACTIVE : Vérifier que la sonde est active et opérationnelle,

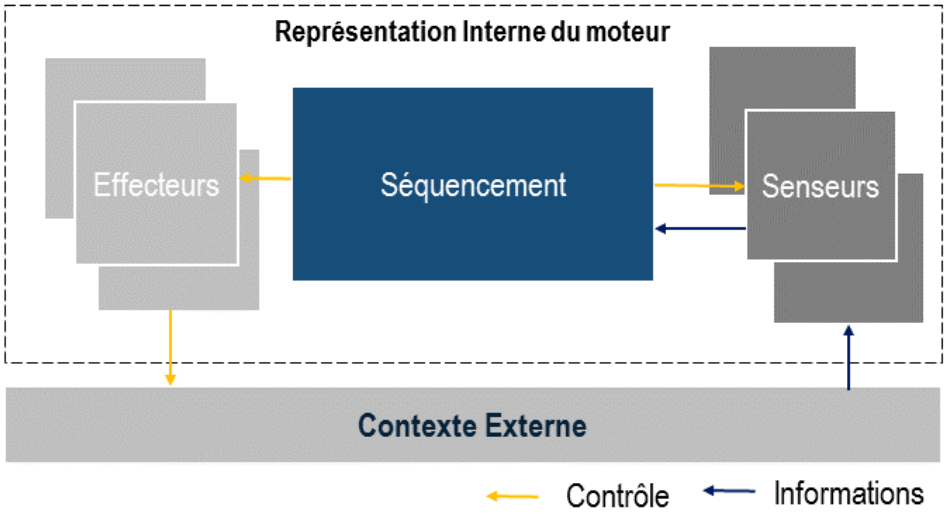


Figure 3.3 – Représentation interne du moteur de scénarios.

- $TEMPERATURE\_INFERIEURE\_A$: Vérifier que la température est bien inférieure à une borne donnée.
- $TEMPERATURE\_SUPERIEURE\_A$: Vérifier que la température est bien supérieure à une borne donnée.

L'Algorithme 1 donne un exemple d'utilisation de cette classe de senseur. Dans cette configuration, il cherche à savoir si la température dans une pièce est inférieure à 18 degrés. La condition décrite peut avoir plusieurs manières d'être validée. Dans notre exemple, cette condition est tout de même validée que la température de la pièce soit à 17 ou à 12 degrés. Il peut alors être intéressant de connaitre la raison de la validation d'une condition. Pour cela, une entité "*contexte évènementiel*", fournissant les éléments ayant participé à la validation de la condition, est retournée par le senseur. Par exemple, le senseur de sonde thermique retourne l'identité de la sonde, son état et la valeur mesurée au moment du test. Le même principe s'applique à n'importe quel type de capteurs ou pour connaitre l'état d'une entité dans un environnement virtuel.

Les Effecteurs offrent une interface qui permet à un moteur de déclencher un comportement dans l'environnement (c.f. Figure 3.5). Comme pour les senseurs, une classe d'effecteurs définit son fonctionnement, et les paramètres de son constructeur spécialisent le comportement attendu. Ce fonctionnement est associé aux entités faisant partie de l'environnement. Pour ne pas bloquer le moteur de scénarios dans un état intermédiaire, le moteur déclenche le comportement de l'effecteur sans en attendre le retour. Leur exécution est soit instantanée, soit réalisée de manière asynchrone. Il est possible d'obtenir le résultat du comportement en utilisant une classe de senseur dont le rôle est de superviser son résultat. Ainsi, le moteur est prévenu de la fin du comportement et a accès à son résultat. Dans la Section 2, nous verrons quelques constructions,

Algorithm 1 Exemple d'utilisation d'une classe de senseurs dans le formalisme XML de #SEVEN.

```
<sensorCheck classname="TemperatureProbeSensor, SEVEN.Automation">
    <param name="Probe"
    value="$(ExternalContext.Etage.ChambreA.TemperatureProbe)"/>
    <param name="Operation" value="TEMPERATURE_LESS_THAN"/>
    <param name="Value" value="18"/>
</sensorCheck>
```

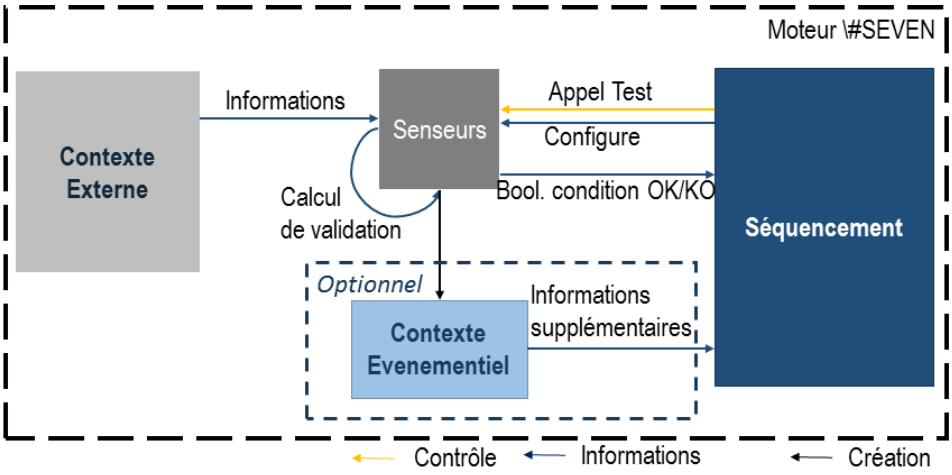


Figure 3.4 – Fonctionnement d'un senseur.

dans le langage #SEVEN, permettant de travailler sur ce comportement temporel. L'algorithme 2 donne un exemple d'effecteur qui déclenche la mise en route d'un radiateur et son réglage à une température donnée.

2 Séquencements

Le Séquencement définit les relations temporelles et causales entre les conditions de senseurs et des effecteurs. La mise à jour de son état est pilotée par le moteur. "*Un séquencement*" active certains senseurs à un instant donné. Lors de l'étape de décision, si la condition de l'un ou plusieurs de ces senseurs est validée, la séquence adapte son état, déclenche des potentiels effecteurs et active ou désactive des senseurs. Lorsque la condition d'un senseur est validée, la séquence a accès à son "*contexte évènementiel*". Cela permet de faire transiter des informations depuis le senseur vers le séquencement ou vers les effecteurs. Enfin, le moteur doit savoir comment initialiser le séquencement et quand son exécution est terminée. Pour cela le séquencement doit avoir un état initial et potentiellement un ou plusieurs états finaux prévus

Algorithm 2 Exemple d'utilisation d'une classe de senseurs dans le formalisme XML de #SEVEN.

```
<effectorUpdate classname="RadioatorControler,␣SEVEN.Automation">
    <param name="Radiator"
    value="$(ExternalContext.Etage.RoomA.Radiator)"/>
    <param name="TargetState" value="ON"/>
    <param name="Temperature" value="18" />
    <param name="Scale" value="CELSIUS" />
</effectorUpdate>
```

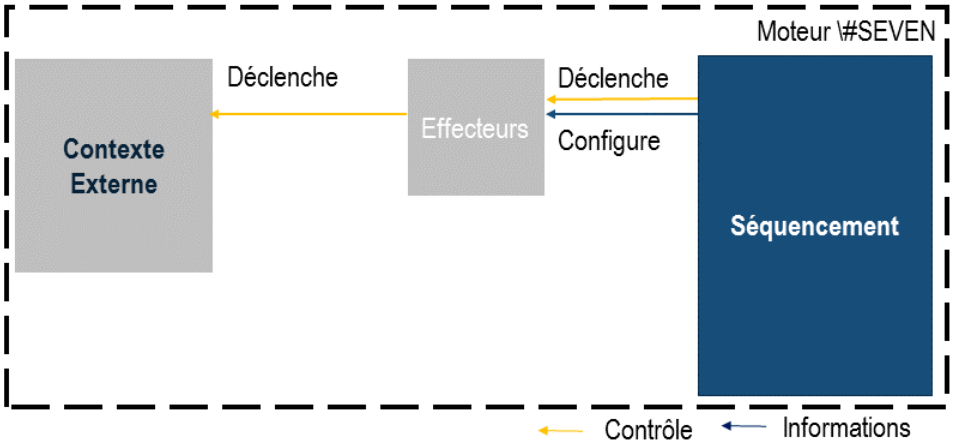


Figure 3.5 – Fonctionnement d'un Effecteur.

dans la spécification des scénarios.

Dans notre contexte, le projet S3PM (c.f. Chapitre 1), les modèles de procédures utilisent les réseaux "*Test and Flip*" [Caillaud, 2013]. Ces réseaux sont capables d'exprimer la concurrence et le parallélisme. Par contre, par défaut ils ne sont pas hiérarchiques et deviennent difficiles à comprendre dès que la taille de l'ensemble de scénarios possibles augmente. De plus, nous cherchons un modèle utilisable pour de la génération automatique qui puisse être utilisé facilement pour de la validation ou de l'écriture manuelle.

Dans le domaine de la synthèse de comportements parallèles, les réseaux de Petri [Murata, 1989; Petri, 1962, 1966] (c.f. Annexe A) sont très utilisés. La synthèse de réseaux de Petri est un problème largement traité dans la littérature [Badouel et al., 2016; Peterson, 1981]. De plus, le formalisme graphique des réseaux de Petri les rend faciles à interpréter et à spécifier. Ils sont par exemple déjà utilisés en tant que modèle de base pour la scénarisation d'environnements virtuels [Brom and Abonyi, 2006; Brom et al., 2007]. La différence fondamentale avec ces travaux est que le moteur #SEVEN n'impose pas une utilisation particulière. Il ne doit pas être le seul composant de l'environnement à pouvoir déclencher effectivement des actions. Dans les travaux

de Brom et al., les autres entités (acteurs de l'environnement virtuel) doivent demander au moteur de déclencher des actions à leur place. Ce fonctionnement peut être mis en place via un moteur #SEVEN, mais n'est pas une obligation.

Les réseaux de Petri possèdent des propriétés intéressantes pour la spécification des séquences, et donc des ensembles de scénarios. Ceci permet au modèle #SEVEN de répondre à certains problèmes soulevés dans le chapitre 2 :

- Ils permettent d'exprimer la séquentialité, la concurrence et le parallélisme, notions très importantes pour permettre de décrire des ensembles de scénarios complexes.
- Ils possèdent une représentation graphique claire, facilitant leur spécification et leur compréhension.
- Ils permettent une structure hiérarchique, offrant ainsi la possibilité de découper les réseaux en différents sous-réseaux. Le fameux diviser pour régner cher aux informaticiens.
- Il est possible de calculer les séquencements possibles entre deux états du réseau pour participer à la prévention des interblocages.

Dans cette section, nous utilisons une classe de réseaux de Petri spécifique : les Réseaux de Petri Saufs Hiérarchiques. Ces réseaux sont équivalents aux machines à états [Murata, 1989], mais proposent un formalisme mieux adapté à la représentation de parallélismes (c.f. Annexe A). La Section 2.1 présente le modèle de réseaux que nous proposons. La Section 2.2 présente comment utiliser les sous-réseaux et les propriétés qu'ils offrent. La Section 2.3 présente une puissante fonctionnalité du modèle qui augmente l'expressivité du langage, toujours pour permettre de définir des scénarios complexes tout en économisant les spécifications. La Section 2.4 détaille comment obtenir simplement des scénarios complexes grâce aux propriétés d'équivalence des réseaux de Petri. Enfin, la Section 2.5 propose de généraliser le concept de séquencement du modèle #SEVEN.

2.1 Fonctionnement

Un réseau de Petri est un graphe orienté composé de Places et de Transitions. Chaque place est reliée à des transitions et chaque transition est reliée à des places. Une place peut contenir des jetons. Ils sont consommés et/ou produits lors de l'activation des transitions. Une transition est dite sensibilisée si chaque place qui la précède immédiatement dans le réseau possède un jeton. Elle peut alors être déclenchée. Une transition peut être attachée au plus à un senseur et un effecteur. Elle peut être déclenchée lorsqu'elle est sensibilisée et que la condition de son senseur est validée. Puisque nous utilisons des réseaux de Petri saufs, chaque place du réseau n'accueille au plus qu'un jeton. La raison de l'utilisation des réseaux saufs est détaillée dans la Section 2.3. Ceci n'a pas un impact fort sur notre utilisation, car nous ne modélisons qu'un seul processus : l'évolution des scénarios possibles. La Figure 3.6 donne un exemple de réseau de Petri utilisé

comme séquence dans la spécification des scénarios écrite dans le langage #SEVEN (formalisme graphique). L'Algorithme 3 donne son équivalent en formalisme XML.

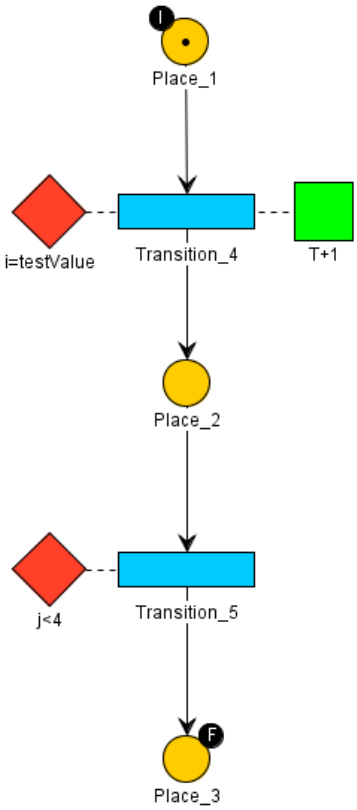


Figure 3.6 – Équivalent graphique du XML de Algorithm 3

Le marquage d'un réseau (nombre de jetons par places) définit son état courant. Les places peuvent être définies comme "*Initiales*" ou "*Finales*". Les places "*initiales*" (places 1 et 15, marquées par un I dans la Figure 3.8) possèdent un jeton à l'initialisation du réseau. Les places "*finales*" (places 21 et 23, marquées par un F dans la Figure 3.8) permettent de décrire l'ensemble des marquages finaux du réseau. Un marquage du réseau est dit final si au moins toutes les places indiquées comme finales possèdent un jeton. Un réseau dont le marquage est final est considéré dans un état final.

La structure du réseau de Petri permet d'exprimer la séquentialité, le parallélisme et la concurrence [Murata, 1989] (c.f. Annexe A). La structure du réseau est chargée à l'initialisation du moteur. La Figure 3.7 présente des exemples de structures de base qui peuvent être utilisées pour exprimer des ensembles de scénarios complexes. Les réseaux 1, 2, 3 et 4 sont à la base d'un réseau #SEVEN. La transition du réseau 1 ne possède ni senseur ni effecteur. Elle est

Algorithm 3 Exemple de spécification de scénarios #SEVEN en XML. La séquence est de la classe des Réseaux de Petri Saufs.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<scenario xmlns="http://www.insa-rennes.fr/#SEVEN">
    <contextInit classname="ContextMapInit, SEVEN.Extra">
        <param name="testValue" value="12"/>
    </contextInit>
    <event id="Event_2">
        <sensorCheck classname="NumberSensor, SEVEN.Extra">
            <param name="Operator1" value="$(ExternalContext.element.j)"/>
            <param name="Operator" value="LESS_THAN"/>
            <param name="Operand2" value="4"/>
        </sensorCheck>
    </event>
    <event id="Event_1">
        <sensorCheck classname="NumberSensor, SEVEN.Extra">
            <param name="Operand1" value="$(ExternalContext.element.i)"/>
            <param name="Operator" value="EQUALS"/>
            <param name="Operand2" value="$(ScenarioContext.testValue)"/>
        </sensorCheck>
        <sensorLabel>i=testValue</sensorLabel>
        <effectorUpdate classname="NumberEffector, SEVEN.Extra">
            <param name="Operand1" value="$(ExternalContext.element.T)"/>
            <param name="Operator" value="INCREASE"/>
            <param name="result" value="$(ScenarioContext.T)"/>
        </effectorUpdate>
    </event>
    <sequence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="SafePetriNet" id="-1">
        <place xsi:type="Place" id="Place_3" label="Place_3"/>
        <place xsi:type="Place" id="Place_1" label="Place_1"/>
        <place xsi:type="Place" id="Place_2" label="Place_2"/>
        <transition id="Transition_4" label="Transition_4">
            <event idref="Event_1"/>
            <upstreamPlace idref="Place_1"/>
            <downstreamPlace idref="Place_2"/>
        </transition>
        <transition id="Transition_5" label="Transition_5">
            <event idref="Event_2"/>
            <upstreamPlace idref="Place_2"/>
            <downstreamPlace idref="Place_3"/>
        </transition>
        <initialPlace idref="Place_1"/>
        <finalPlace idref="Place_3"/>
    </sequence>
</scenario>
```

peut être déclenchée dès que la place à laquelle elle est liée possède un jeton et ceci n'a pas de conséquence autre que de changer l'état du réseau. Une telle transition peut être utilisée comme élément structurel interne du réseau (par exemple comme proposé dans le réseau 8). Le réseau 2 présente une transition qui possède une condition A définie par un senseur, mais qui ne déclenche pas d'actions. À l'inverse, le réseau 3 ne possède pas de condition (tout comme le réseau 1), mais son déclenchement va également déclencher l'action X. Le réseau 4 présente un lien de causalité entre une condition et une action. Ici, il exprime "Si A, alors faire X" ou A est une condition et X une action. Le réseau Le réseau 5 présente une boucle. Le senseur associé à la transition est réinitialisé dès que la transition est activée, car sa place amont reçoit un jeton directement. Le réseau 6 représente un "*ou exclusif*". Les conditions B et C sont en concurrence. Le principe peut être étendu à un grand nombre de conditions. Un cas particulier existe si C vaut $\overline{B}$, alors le réseau 2 signifie "Si B, alors ... Sinon ...". Le réseau 7 décrit un parallélisme. D et E peuvent être réalisés, ou pas, et dans n'importe quel ordre. Encore une fois, ce principe peut être étendu à plus de deux conditions. Le réseau 8 décrit une conjonction. Pour pouvoir déclencher la transition H, les conditions F et G doivent toutes deux être réalisées, quel que soit l'ordre. Ce principe peut également être étendu à plus de deux conditions.

La Figure 3.8 donne un exemple de réseau #SEVEN utilisant différents agencements possibles. La Transition 2 sort sur deux branches parallèles qui se rejoignent à la transition 22. Les Transitions 12 et 13 sont en concurrences sur la place 11 (ou exclusif). Le réseau de droite (Transitions 16, 17 et 18) est en parallèle du réseau de gauche, car ils sont initialisés au même moment (places initiales 1 et 15). Ils sont donc équivalents à un unique réseau ou les places 1 et 15 seraient en aval d'une même transition.

La mise à jour des réseaux de Petri peut être réalisée par plusieurs algorithmes. Ils diffèrent dans leurs choix de la prochaine transition à activer. Celui de base propose de déclencher aléatoirement une transition. Afin de garantir le contrôle sur le comportement du moteur, nous conseillons d'utiliser un algorithme déterministe dans le choix des transitions. Par exemple, dans notre implémentation, nous parcourons les transitions dans leur ordre de création jusqu'à stabilisation de l'état du réseau. Cet algorithme est complété par un système détection des boucles (par exemple par marquage) pour ne pas rester indéfiniment dans un pas de mise à jour du réseau. Ainsi, nous assurons la reproductibilité des exécutions lorsque les actions dans l'environnement ont lieu dans le même ordre. Ceci n'est qu'une possibilité. Il est facile de remplacer cet algorithme par un autre si nécessaire. Enfin, lors de la spécification des scénarios, le développeur ne doit pas tenir compte de cet algorithme pour éviter qu'une spécification ne soit spécialisée à une implémentation du modèle et donc difficilement réutilisable.

Puisque le moteur de scénarios est en attente de la réalisation de conditions, des situations d'interblocage peuvent avoir lieu. Par exemple, une entité E de l'environnement attend que la représentation des scénarios arrive à un état A. De son côté le moteur de scénarios attend que la condition C ait lieu dans l'environnement pour atteindre A. Si E est la seule entité à pouvoir réaliser C alors E et le moteur de scénarios sont en interblocage. Le calcul d'accessibilité (p. ex. [Mayr, 1984], voir Annexe A) permets de savoir s'il existe une séquence de marquages d'un réseau permettant d'atteindre un marquage cible à partir du marquage courant. En supposant

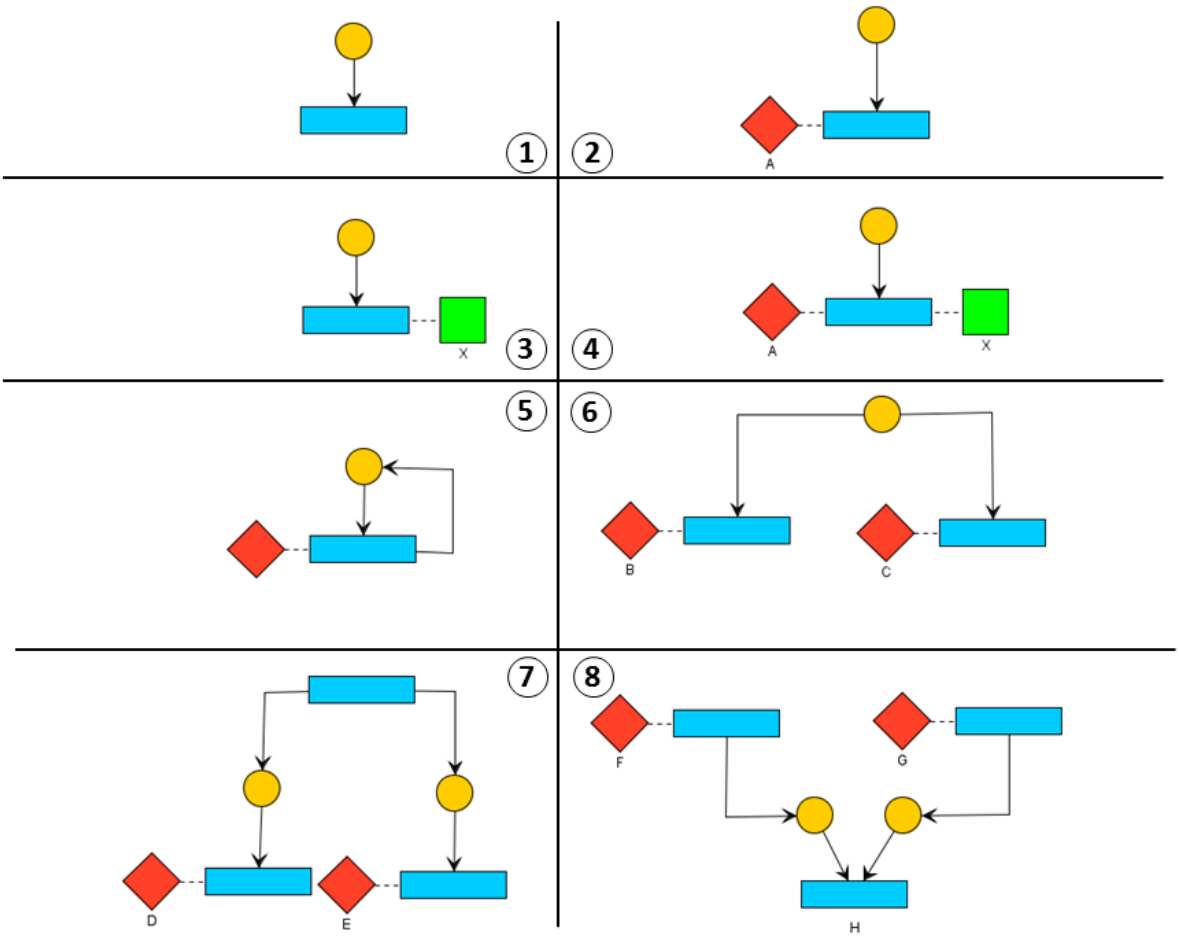


Figure 3.7 – Exemples de structures de base pour réaliser des ensembles de scénarios complexes

que les conditions des senseurs puissent être validées, il est donc possible de savoir si, pour le marquage courant, le réseau peut atteindre au moins un des marquages finaux du réseau et donc de prévoir les blocages au niveau du moteur #SEVEN. Le calcul d'accessibilité permet aux entités autonomes dans l'environnement de raisonner sur les marquages à emprunter pour prévenir les interblocages par exemple en utilisant des méthodes d'action planning.

2.2 Organisation Hierarchique

Les réseaux de Petri peuvent être hiérarchiques au niveau des places ou des transitions. Notre modèle nous n'utilisons que la possibilité de définir des places hiérarchiques pour deux raisons. La première est que ceci ne réduit pas l'expressivité du modèle initial, car il est possible de décrire des réseaux équivalents utilisant des sous-réseaux au niveau des places ou des transitions [Ladet, 1989]. La seconde est que cela ajouterait deux approches différentes dans la manière de

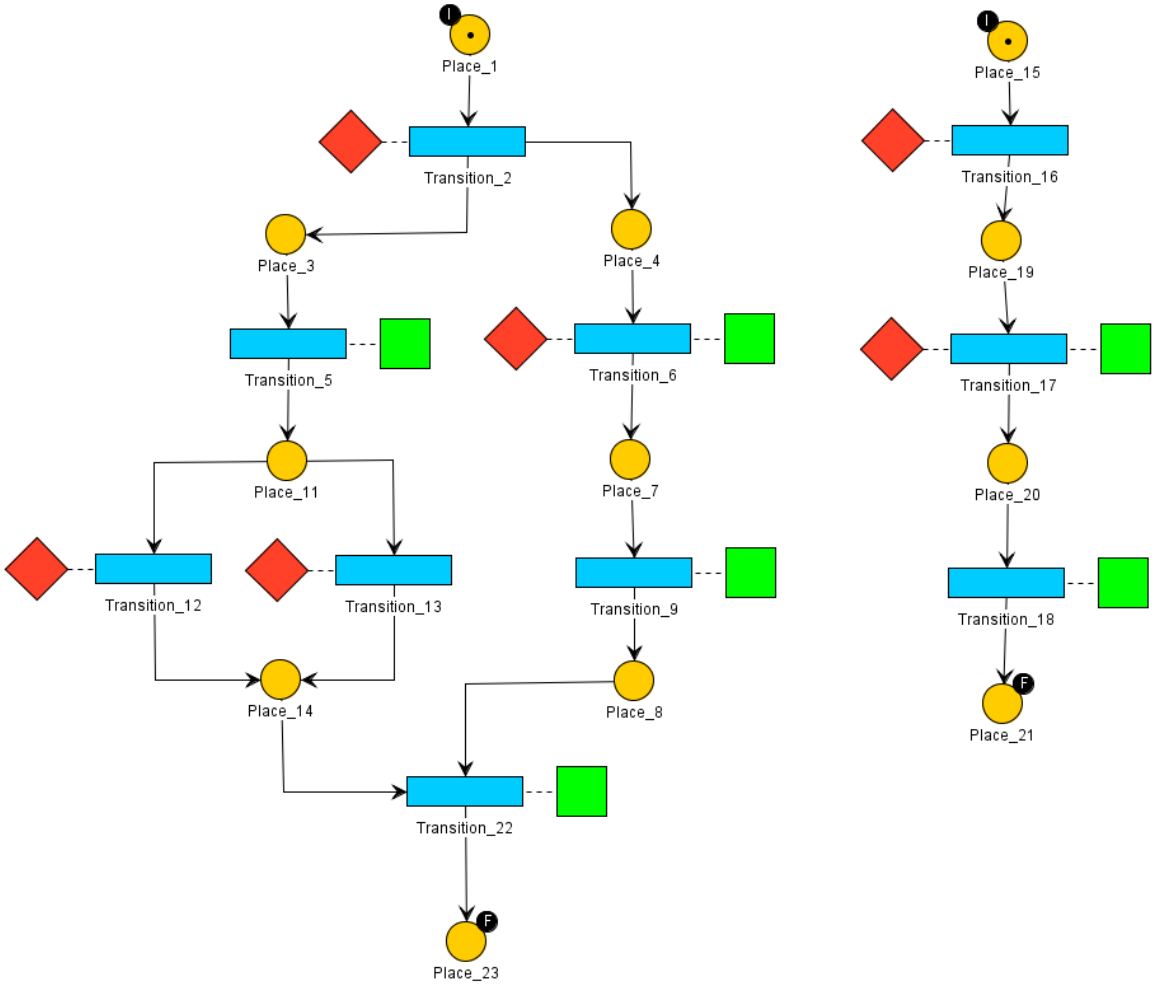


Figure 3.8 – Exemple de séquencement #SEVEN à partir du modèle de Réseaux de Petri. Ces deux réseaux sont considérés comme deux branches parallèles d'un unique réseau à cause du marquage initial.

décrire les sous-réseaux, ce qui complexifierait l'apprentissage et l'utilisation du langage.

Les places hiérarchiques permettent l'équivalent #SEVEN des fonctions dans un langage de programmation classique. Cela permet de découper la spécification des scénarios pour la rendre plus lisible et simplifier la spécification. Une place peut directement décrire le sous-réseau qu'elle contient ou lire une spécification de scénarios externe. Le sous-réseau est chargé dynamiquement lorsqu'un jeton entre dans la place. Ceci permet de définir le réseau chargé à l'exécution, pour offrir notamment la possibilité de décrire des spécifications récursives.

Un sous-réseau possède également des places marquées comme "*Initiales*" ou "*Finales*".

Lorsqu'un jeton entre dans une place hiérarchique, un jeton est produit sur chaque place initiale du sous-réseau. Lorsque toutes les places finales du sous-réseau portent un jeton, la place hiérarchique porte un jeton. Si le jeton de la place hiérarchique est consommé, les jetons des places finales du sous-réseau sont consommés également. La mise à jour d'un sous-réseau est appelée par le réseau parent lors de sa propre mise à jour.

La Figure 3.9 donne deux exemples d'utilisations de sous-réseaux. Le réseau A possède un sous-réseau sur la place 4. Lorsqu'un jeton y est placé, un jeton est également placé sur la place 7. La transition 2 ne pourra être déclenchée que s'il y a un jeton sur la place 8. La place 5 est également un sous-réseau, mais défini dans un fichier externe. Lorsque ce sous-réseau atteindra un marquage final, la place 5 sera également marquée et le réseau A portera aussi un marquage final.

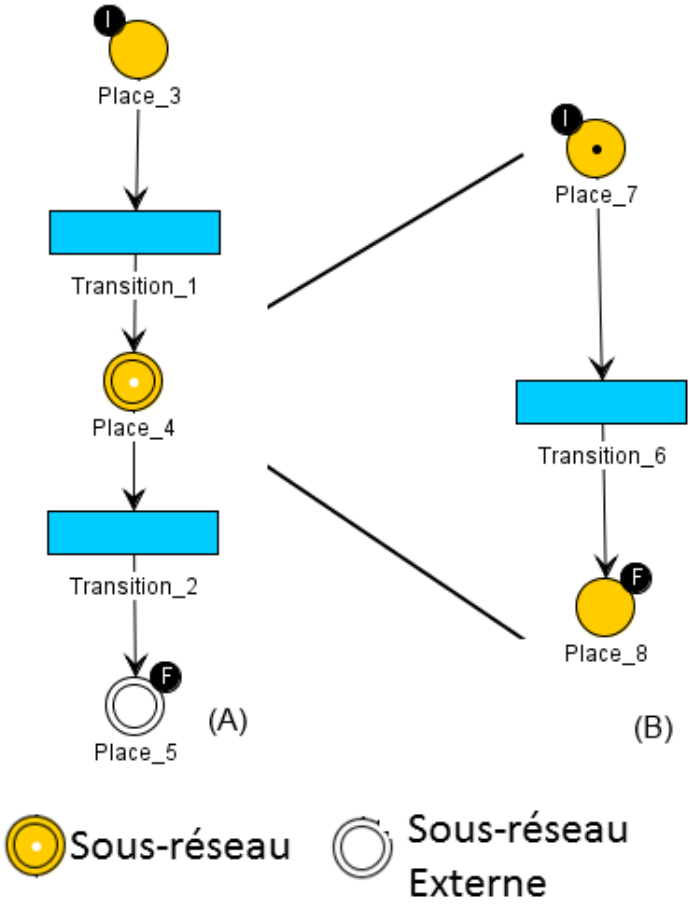


Figure 3.9 – Exemples d'utilisation des sous-réseaux.

2.3 Paramétrage local du Réseau

Afin d'offrir plus d'expressivité au langage, il peut être intéressant de pouvoir paramétrer dynamiquement certains éléments comme les senseurs/effecteurs ou de pouvoir prendre certaines décisions en fonction d'informations obtenues plus avant dans le réseau. Prenons par exemple le calcul de la suite de Fibonacci (pour rappel $n = n_{-1} + n_{-2}$). Sans configuration dynamique, il est impossible de réaliser ce calcul, car leurs résultats ne sont pas mémorisés. Nous proposons d'étendre le modèle de réseaux en ajoutant la possibilité de faire transiter des informations grâce aux jetons. Pour cela, nous utilisons les éléments suivants :

Un jeton est une collection de variables. Ces variables peuvent être de types de base ou des entités plus complexes. Les senseurs et effecteurs peuvent utiliser ces variables par l'intermédiaire du "*Contexte local*" lié à une transition en aval de la place qui contient le jeton. Les places initiales peuvent décrire le jeton qu'elles contiennent lorsqu'il est créé. Dans le cas des sous-scénarios, cette fonctionnalité est ignorée pour permettre d'utiliser le jeton en entrée du réseau à la manière des paramètres des fonctions.

Le contexte local est un ensemble de données issues de tous les jetons contenus dans les places en amont d'une transition dont la portée est limitée à cette transition. Pour manipuler les données des jetons, une transition est associée à deux fonctions. "*La fonction de fusion*" construit le contexte local à partir des jetons et "*la fonction de distribution*" distribue les données du contexte local dans les jetons des places en aval. Les senseurs et les effecteurs ont accès au contexte local en lecture. De plus, certains types d'effecteurs peuvent modifier les variables du contexte local. Ces effecteurs réalisent leur calcul avant de rendre la main à la transition pour la suite de la mise à jour. Ils doivent donc réaliser des calculs instantanés. Dans notre implémentation, nous proposons un ensemble de senseurs et d'effecteurs qui permettent de réaliser des opérations de base, ou plus avancées, sur des types de bases. Par exemple, réaliser des calculs sur des variables numériques ou manipuler des chaines de caractères.

La Fonction de Fusion construit le contexte local lors de la vérification de la condition d'un senseur. Par défaut elle réalise une union des variables issues de chaque jeton. Cependant, elle peut être redéfinie pour prendre en compte des cas qui poseraient des problèmes. Par exemple, si deux jetons contiennent une variable nommée A associée des valeurs différentes "*la fonction de fusion*" peut proposer différents traitements comme en renommer une, l'ignorer, réaliser une moyenne ou choisir la plus grande valeur des deux.

La Fonction de Distribution construit les jetons positionnés dans les places en aval après le déclenchement de l'effecteur. Elle réalise une copie du contexte local dans chacun des jetons. Elle permet également d'ajouter dans les jetons des données issues d'autres sources. Par exemple, elle peut permettre de faire passer à la suite du réseau des informations issues du "*contexte évènementiel*" (c.f. Section 1) obtenu lors du déclenchement de son senseur. L'algorithme de déclenchement d'une transition est donc le suivant :

1. Création du contexte local par la fonction de fusion

2. Vérification de la condition par le senseur

3. Récupération du contexte évènementiel (c.f. définition des senseurs, Section 1)

4. déclenchement de l'effecteur.

5. Consommation des jetons

6. Production des nouveaux jetons par la fonction de distribution

L'introduction de cette fonctionnalité est la raison pour laquelle nous utilisons des réseaux de Petri saufs. Elle induit un problème dans un réseau de Petri non-sauf (plus de 1 jeton par places) : lorsqu'une transition est activée, elle consomme un jeton au hasard dans chacune de ses places en amont. Par exemple, si une transition possède des places amont A et B et qu'elles contiennent chacune deux jetons a_1, a_2 et b_1, b_2, la transition consommera aléatoirement une des paires parmi $(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)$,. Ceci peut être un problème si les jetons contiennent des données qui doivent être cohérentes entre elles. En théorie, les jetons représentent des exécutions parallèles du processus représenté par le réseau. Puisque nous cherchons à représenter l'exécution d'un scénario parmi un ensemble de possibles, nous avons donc restreint la classe de réseaux utilisés aux réseaux sauf pour assurer la cohérence des données.

La Figure 3.10 présente un réseau permettant le calcul de la suite de Fibonacci. Lors du chargement du réseau, le jeton initial contient les deux premiers éléments de la suite (A et B), ainsi que le nombre d'itérations à réaliser $Iter$. L'effecteur de la Transition 2 réalise le calcul de l'élément suivant et l'ajoute au jeton dans la variable RES. La transition 3 décrémente le nombre d'itérations à réaliser grâce à son effecteur, puis réalise grâce à la fonction de sa répartition, l'échange des valeurs telles que la valeur de B du jeton dans la place $PResult$ soit donnée à la variable A du jeton dans sa place en sortie et que la valeur de la variable RES soit donnée à la variable B. Tant que le nombre d'itérations n'est pas atteint, le réseau réinjecte le nouveau jeton à la place initiale. Lorsque la valeur de la variable $Iter$ atteint 0, le jeton est placé dans la place finale "*Fin*".

La Figure 3.11 présente le même algorithme récursif utilisant les sous-réseaux. Si $Iter$ ne vaut pas 0, le réseau courant charge un sous-réseau qui réalise une étape du calcul. Le jeton initial dans le sous-réseau est alors remplacé par le jeton issu du réseau parent. Lorsque $Iter$ atteint 0 dans le réseau de plus bas niveau, le jeton est placé sur la place finale (Transition 4) sans créer de nouveau sous réseau. La transition 10 du réseau parent consomme alors le jeton de la place hiérarchique et le place sur la place finale, résolvant la récursivité jusqu'au réseau parent de plus haut niveau.

Les jetons permettent de passer des données d'un réseau parent à un sous-réseau et inversement. Les "*places hiérarchiques*" utilisent une "*Fonction de Distribution*" pour répartir les données dans les jetons des places initiales de leur sous-réseau. Elles utilisent également une "*Fonction de Fusion*" pour créer le jeton qu'elles contiennent, lors d'un marquage final du sous-réseau, à partir des jetons contenus dans les places finales de leur sous-réseau.

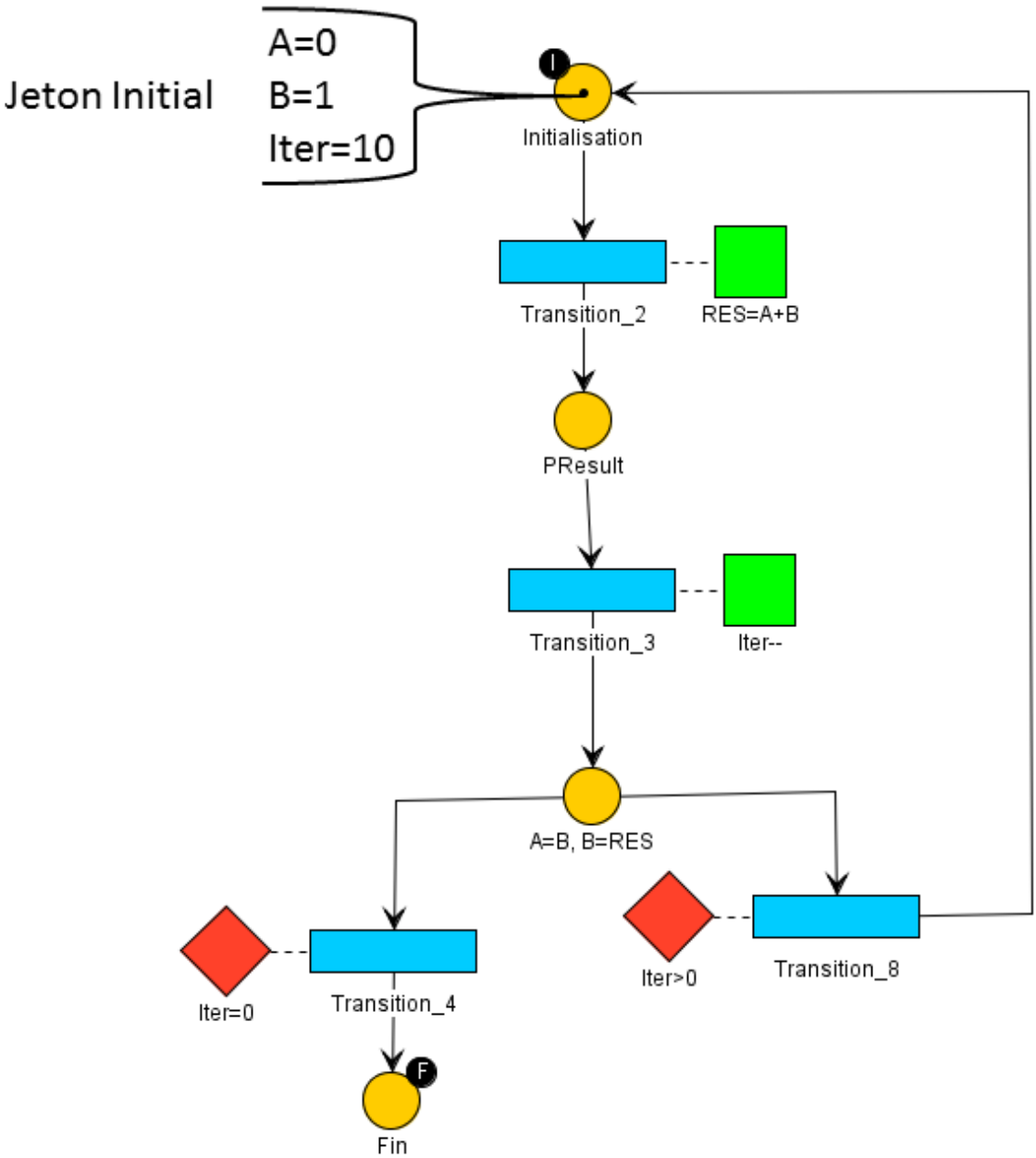


Figure 3.10 – Calcul de la suite de Fibonacci grâce à un réseau #SEVEN et à la configuration locale.

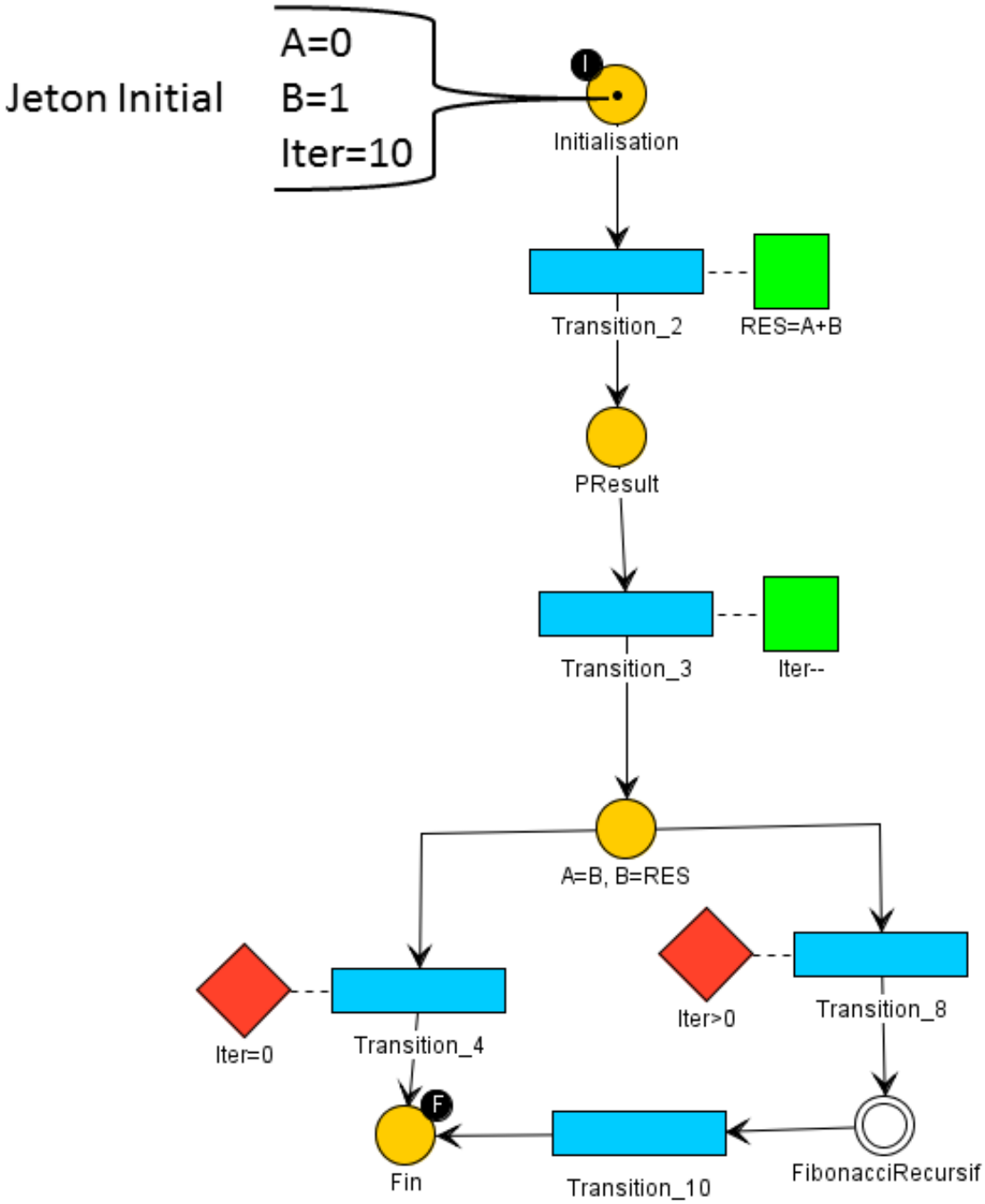


Figure 3.11 – Implémentation récursive du calcul de la suite de Fibonacci.

2.4 Transitions et Senseurs équivalents

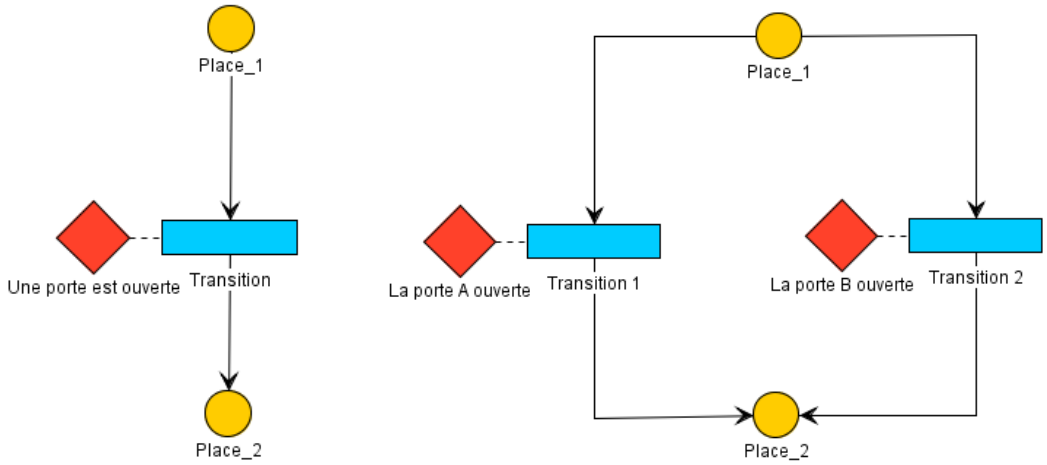
Certaines conditions définies par les senseurs peuvent être validées de différentes manières (c.f. Section 1). Preonons par exemple deux réseaux de Petri R_1 et R_2 :

- R_1 est composé de N transitions t_1 à t_n en concurrence (par exemple comme présenté dans le réseau 6 de la figure 3.7), déclenchées respectivement par les conditions c_1 à c_n,.
- R_2 est composé d'une seule transition T déclenchée par la condition C.
- R_1 et R_2 sont équivalents
  - Si la condition C est validée pour n'importe quelle validation de condition c_1 à c_n
  - et que la condition C n'est déclenchée par aucune autre condition.

De notre point de vue, si la condition d'un senseur est réalisable de plusieurs manières différentes, il est possible d'exprimer plus de scénarios tout en gardant un réseau compact. Par exemple, les scénarios des Figures 3.12a et 3.12b sont équivalents si dans l'environnement il n'y a que les portes A et B. Si l'environnement comprend une porte C, alors le réseau de la Figure 3.12a propose un scénario supplémentaire (porte C ouverte) que l'ensemble des scénarios proposés par le réseau 3.12b ne contient pas. L'ensemble des scénarios définis par une spécification peut donc dépendre de l'environnement dans lequel elle est utilisée.

2.5 Utilisation d'autres séquencements

Le modèle #SEVEN ne fait pas d'hypothèses sur la classe d'automates utilisée pour définir la séquence. Néanmoins, dans le chapitre 2 nous indiquons qu'un modèle de scénarios doit être capable d'exprimer la concurrence et le parallélisme pour permettre la description d'agencements complexes. Nous préconisons donc l'utilisation d'automates parallèles comme les machines à états parallèles [Cremer et al., 1995] ou les réseaux de Petri [Petri, 1962, 1966] (c.f. Annexe A). Les langages de programmation actuels permettent de simplifier la spécification des programmes permettant un découpage et une réutilisation de certaines parties (p. ex. Fonctions et classes). Pour simplifier la spécification, nous préconisons donc également l'utilisation de classes d'automates hiérarchiques pour permettre un découpage scénarios/sous-scénarios simplifiant la lecture et permettant la réutilisation de scénarios externes. Enfin, une représentation graphique permet d'appréhender plus simplement le comportement du système par une vue d'ensemble structurée [Sugiyama et al., 1981]. Néanmoins, une classe de machines à états classiques peut être suffisante dans des cas simples où le parallélisme et la structure hiérarchique ne sont pas nécessaires à l'expression des scénarios. Cette propriété de #SEVEN permet également d'intégrer des automates générés par un processus automatique, comme proposé dans notre contexte projet (c.f. Chapitre 1).



(a) **(b)**

Figure 3.12 – Ces spécifications de scénarios sont équivalentes si l'environnement contient uniquement les deux portes A et B.

2.6 Portées et utilisations des variables

La configuration des scénarios passe par un ensemble de variables. Par exemple, un système de domotique va se baser sur l'heure, une température cible différente dans chaque pièce ou l'ensoleillement sur une zone de la maison. L'utilisateur paramètre le comportement du système grâce à des valeurs souhaitées pour ces éléments. Par exemple il va indiquer que le chauffage se déclenche en dessous de 18 degrés. Il s'agit d'informations qui peuvent être utilisées à plusieurs endroits lors de la spécification des scénarios (p. ex. dans la configuration des senseurs, des effecteurs ou du séquencement). Les langages de programmation proposent également différentes portées des variables. Même si ce concept n'est pas indispensable, il est tout de même très pratique pour simplifier la réalisation de programmes complexes.

Dans #SEVEN, chaque contexte (externe, local et évènementiel) propose un accès à des variables. Chacun possède une portée différente. Dans cette section, nous présentons comment sont gérées les variables en fonction du contexte dans lequel elles sont définies.

Le contexte externe est accessible à tous les éléments du moteur de scénarios. Il s'agit de variables globales, partagées même avec les autres entités de l'environnement virtuel. Les actions déclenchées par les effecteurs dans l'environnement sont soit instantanées, soit réalisées de manière asynchrone, car elles pourraient bloquer le moteur dans un état intermédiaire. Si nécessaire, il est possible d'utiliser une classe de senseur pour attendre la réalisation effective de cette action, comme dans l'exemple de la Figure 3.13. L'effecteur déclenche l'action A, la *transition 2* bloque le séquencement jusqu'à la réalisation effective de A. Le résultat de A est

obtenu par le contexte évènementiel du senseur de la *transition 2*.

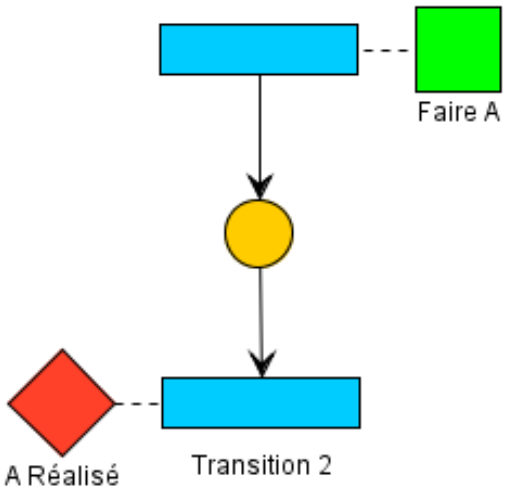


Figure 3.13 – Réseau #SEVEN permettant une attente active de la fin d'un traitement lancé en amont.

Si l'environnement prend en compte la notion de ressources, leurs états, et est capable de gérer leurs accès concurrents, alors le moteur #SEVEN peut se synchroniser avec le comportement d'autres entités de l'environnent. La Figure 3.14 montre un exemple de gestion des ressources. Le premier effecteur demande l'accès à la ressource. Lorsque c'est le cas (senseur de la seconde transition), la ressource peut être utilisée (deuxième effecteur). Une fois la ressource utilisée, elle est libérée (troisième effecteur).

Le contexte local permet de spécialiser une transition (et ses éléments associés) à partir des variables contenues dans les différents jetons de ses places directement en amont. Ces variables sont ensuite passées à la suite du réseau lors du déclenchement des transitions. Les effecteurs peuvent modifier les variables du contexte local uniquement de manière synchrone, car il n'existe que lors de la tentative d'activation de la transition. Le contexte local peut être utilisé dans la spécification du senseur et de l'effecteur pour permettre une configuration dynamique de l'ensemble des scénarios. Le contexte local permet également le transfert d'informations entre un réseau parent et un sous-réseau, à la manière des paramètres de fonctions dans un langage de programmation.

Le contexte évènementiel est fourni par un senseur pour décrire les éléments qui sont intervenus dans la validation de la condition. Ces variables sont accessibles uniquement lors de l'activation de la transition, si la condition du senseur est validée, par la transition (pour permettre par exemple un passage à la suite du réseau au travers du contexte local) ou par l'effecteur associé pour permettre sa configuration dynamique.

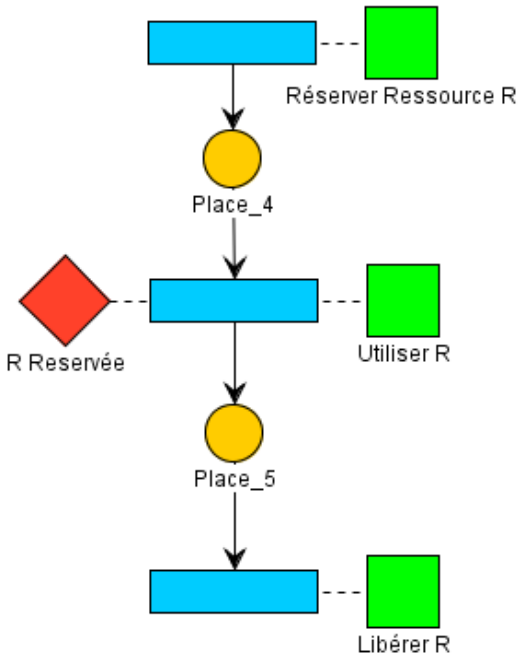


Figure 3.14 – Gestion des ressources dans un réseau #SEVEN

Le modèle #SEVEN propose donc des variables globales (contexte externe) ou des variables locales (contexte local ou évènementiel). De notre point de vue, il manque un niveau intermédiaire entre les variables accessibles à tous et celles accessibles de manière très locale. Cette portée permettrait de décrire des variables utilisées communément par plusieurs composants d'un réseau ou d'un sous-réseau. Pour cela, nous proposons une entité supplémentaire : le contexte de scénarios.

Le contexte de scénarios est un ensemble de variables qui permet la configuration de l'ensemble des scénarios définis par un réseau. Ces variables peuvent être utilisées comme paramètre par les senseurs, les effecteurs ou les transitions. Elles ne sont pas accessibles aux sous-réseaux. Chaque sous-réseau possède son propre contexte de scénarios. Les variables initiales du contexte de scénarios sont déclarées dans la spécification des scénarios. Le contexte de scénarios peut être utilisé et modifié en utilisant des classes de senseurs et d'effecteurs appropriées, de la même manière que le contexte local. Par exemple, il est possible de compter le nombre de déclenchements d'une transition spécifique pour pouvoir réagir dans le cas ou cette valeur dépasse un certain nombre. Pour cela, il est possible de déclarer un entier à 0 dans le contexte de scénarios et définir un effecteur qui incrémentera cette valeur à la validation du senseur dont on veut compter les déclenchements. Dans une autre partie du réseau, un senseur attendra que la variable atteigne le nombre d'exécutions prévu pour pouvoir déclencher un autre comportement.

3 Informations à destination des autres entités de l'environnement

La représentation de l'ensemble des scénarios possibles, interne au moteur de scénarios, permet à d'autres entités de l'environnement de connaitre quelles sont les actions qui doivent être réalisées pour faire passer ce moteur d'un état à un autre. Pour cela, elles peuvent utiliser les conditions attendues par les senseurs. Par exemple, dans un environnement virtuel, si le moteur de scénarios indique qu'il attend que la lumière soit allumée alors un acteur (réel ou virtuel) sait qu'il doit allumer la lumière. Néanmoins, il est possible que l'information ne soit pas suffisante pour que l'acteur décide de l'action à réaliser. Par exemple, s'il existe plusieurs interrupteurs, mais qu'un seul allume la lumière, l'acteur doit deviner celui qui correspond. Il peut donc être intéressant de compléter la spécification des scénarios grâce à des informations supplémentaires à destination des autres entités de l'environnement.

#SEVEN utilise des ***attributs génériques*** pour représenter ces informations. Contrairement aux variables des contextes, les attributs sont portés par les éléments d'une séquence. Un élément d'une séquence peut porter un nombre quelconque d'attributs. Un attribut possède un type nommé et une collection de paires clef/valeur. Par exemple, pour aider un acteur à choisir un interrupteur qui permet d'allumer une lumière précise, un attribut de type "*Aide_Action*" peut être utilisé pour donner l'instance d'interrupteur qui permet d'allumer effectivement la lumière, ainsi que l'action à réaliser. Il pourrait est défini de la manière suivante : $\{Type : Aide\_Action; Action : Appuyer; Objet : Interupteur3\}$.

Il existe plusieurs moyens de définir les attributs. Ils peuvent être intégrés dans la spécification des scénarios ou être ajoutés automatiquement par le moteur. Enfin, certaines entités de l'environnement peuvent ajouter des attributs à l'exécution. Par exemple, dans le cadre d'un environnement virtuel de Formation, un moteur de pédagogie complètera la spécification des scénarios en indiquant des embranchements comme autorisés ou interdits.

4 Synthèse

#SEVEN permet la spécification et l'exécution de scénarios dans différents environnements. Son fonctionnement se base sur un système Perception-Décision-Action. Il prend en compte les actions qui ont lieu dans l'environnement grâce aux senseurs, des entités permettant dans la vérification de certains types de conditions haut niveau. #SEVEN peut également agir sur l'environnement grâce aux effecteurs, des entités qui permettant le déclenchement d'actions dans l'environnement. L'intégration d'un moteur #SEVEN dans un environnement quelconque est possible grâce à un système de bibliothèques de classes (notamment senseurs et effecteurs) spécialisées. Le système de décisions peut se baser sur un vaste ensemble de modèles. Nous proposons d'utiliser une représentation à base de réseaux de Petri saufs hiérarchiques qui offrent une grande expressivité (p. ex. Parallélisme, concurrence), une représentation graphique et des fonctionnalités permettant de spécifier des ensembles de scénarios complexes grâce à des réseaux

compacts.

Nous utilisons deux moyens pour réaliser des spécifications de scénarios. La première, manuelle, consiste à utiliser l'outil auteur que nous avons développé en parallèle du modèle. Ce dernier est présenté dans la Figure 3.15. Une zone de dessin permet la spécification des réseaux et sous-réseaux. Une interface (à droite) permet un paramétrage plus précis. Par exemple, ici elle permet la modification des senseurs et des effecteurs associés à la transition 4. Les entités utilisées et leurs paramétrages possibles sont chargés directement à partir des bibliothèques. Tous les réseaux présentés dans ce Chapitre et dans les suivants sont réalisés directement grâce à cet outil. En plus de permettre la spécification de scénarios. Cet outil permet de suivre en direct l'état d'un moteur de scénarios et également d'agir dessus en déclenchant manuellement certaines transitions. Ainsi, il est possible pour une personne externe au système d'interagir de manière indirecte avec l'environnement. Ceci peut être intéressant, par exemple, pour un formateur qui souhaite placer un apprenant dans une situation spécifique tout en gardant le contrôle de la simulation.

Certaines propriétés nécessaires aux modèles de scénarios pour environnements virtuels dépendent du modèle utilisé pour spécifier l'environnement :

- Prendre en compte les ressources disponibles n'est possible que si elles sont modélisées dans l'environnement.
- Le contrôle et la prise en compte des acteurs dépendent de leur intégration dans l'environnement.

Dans le Chapitre suivant, nous proposons d'utiliser #SEVEN pour la spécification et l'exécution de scénarios en environnements virtuels collaboratifs. Comme certains travaux, tels que HCSM [Cremer et al., 1995] ou HPTS++ [Lamarche and Donikian, 2002], nous faisons l'hypothèse que définir les comportements possibles d'une entité, par exemple un objet dans un monde virtuel, est équivalent à définir un ensemble de scénarios. Comme #SEVEN ne fait pas d'hypothèses sur l'environnement dans lequel il est utilisé, nous l'utilisons également pour décrire et exécuter les comportements possibles de certaines entités de l'environnement. Pour ce faire, le contexte externe fournit les entrées et sorties, utilisées par des senseurs et effecteurs spécialisés, permettant de superviser l'état de l'entité ainsi que de déclencher des actions. Dans le Chapitre 5, nous utilisons cette propriété pour décrire, grâce à #SEVEN, les règles et habitudes associées à des équipes d'acteurs.

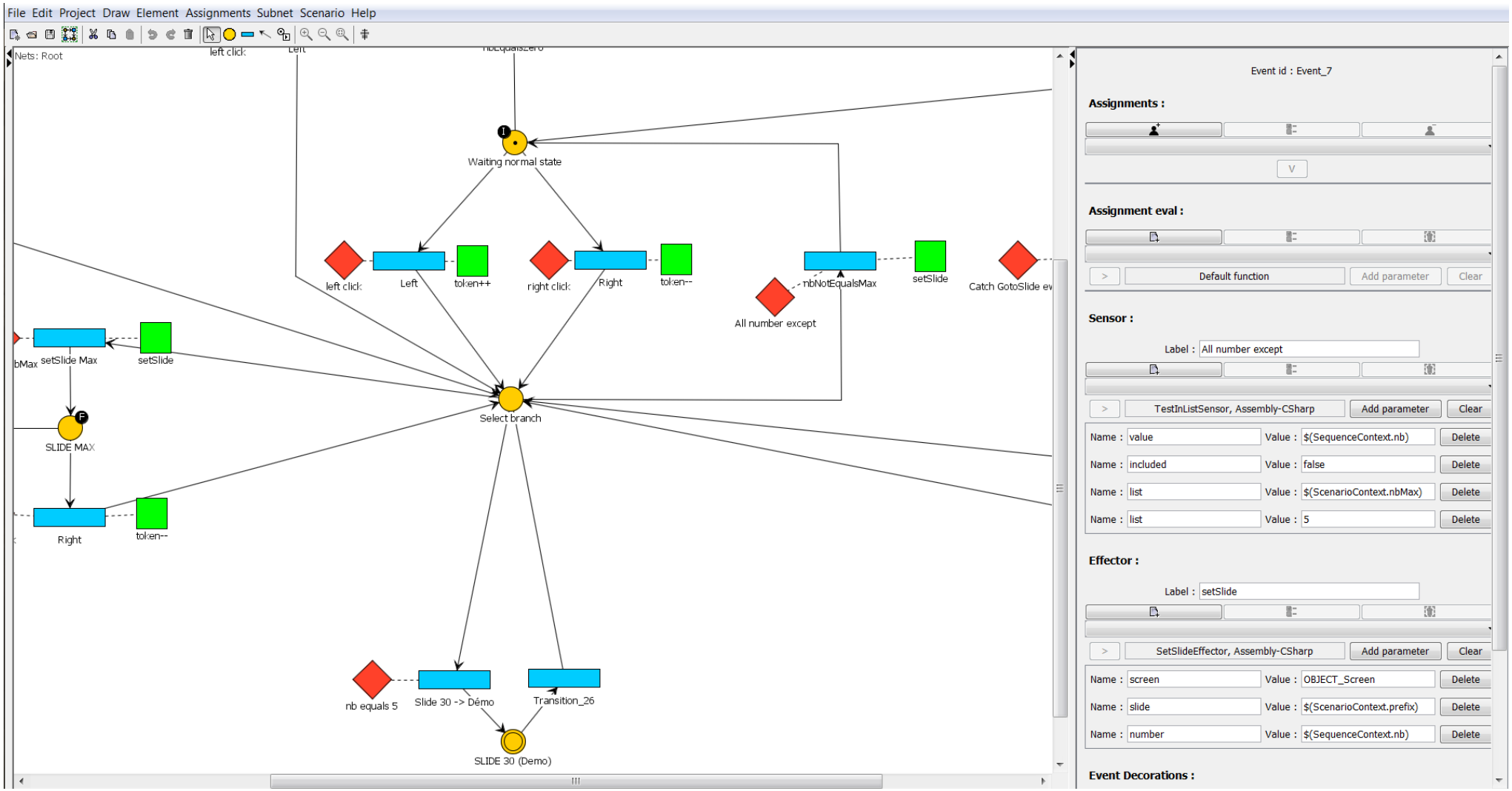


Figure 3.15 – Interface de l'éditeur de réseaux #SEVEN

Scénarios pour Environnements Virtuels 4

#SEVEN est un modèle générique pour la spécification et l'exécution de scénarios (c.f. Chapitre 3). Dans ce chapitre, nous nous intéressons spécifiquement à la spécification et à l'exécution de scénarios pour environnements virtuels. Voici comment #SEVEN se positionne par rapport aux propriétés que nous avons définies dans le Chapitre 2 comme nécessaires aux modèles de scénarios pour environnements virtuels.

- "*Décrire des agencements complexes (temporels et causaux) d'activités*" : Pour cela, #SEVEN utilise une mise en oeuvre des réseaux de Petri saufs hiérarchiques.
- "*Prendre en compte les ressources disponibles*" : #SEVEN ne propose pas explicitement de système de gestion de ressources. Elles pourront néanmoins être prises en compte dans la spécification des scénarios si elles font partie de l'environnement et si elles sont rendues accessibles par le contexte externe.
- "*Modifier l'environnement virtuel en dehors des actions des acteurs*" : #SEVEN est capable de modifier l'environnement dans lequel il est intégré grâce aux classes d'effecteurs, mais l'environnement doit lui fournir les moyens permettant de le modifier.
- "*Ne pas faire d'hypothèses sur le nombre ou la nature des acteurs impliqués (réels ou virtuels)*" : #SEVEN ne fait pas d'hypothèses sur la présence ou non des acteurs.
- "*Exprimer différents niveaux de guidage des acteurs*" : #SEVEN peut déclencher des actions qui impliquent des acteurs si l'environnement le permet. Les informations fournies aux acteurs dépendent des composants #SEVEN (senseurs, effecteurs) faisant le lien avec l'environnement.
- "*Simplifier la spécification de scénarios*" : #SEVEN permet la spécification compacte, structurée et graphique de scénarios complexes, notamment grâce aux réseaux de Petri sauf hiérarchiques.

Une partie de ces propriétés dépend du modèle utilisé pour représenter l'environnement virtuel sur lequel se greffe le moteur de scénarios. Dans le Chapitre 2, nous avons classé les modèles existants en trois familles : les modèles "*génériques*", les modèles à base d' "*objets synoptiques*" et les modèles "*objets-relations*". Nous proposons d'utiliser les modèles de la famille "*objets-relations*" pour la spécification d'environnements virtuels, car ils offrent les propriétés suivantes :

- Une représentation claire des objets et des actions. Les objets sont les entités de base et les relations décrivent les actions réalisables avec ces objets.
- Ils permettent de raisonner sur ces relations. Par exemple en répondant aux questions comme "*quels objets peuvent être utilisés pour réaliser l'action A ?*".
- Ils permettent la collaboration entre acteurs réels et virtuels au niveau le plus haut : la comanipulation d'objets.
- Ils prennent en compte le comportement des objets.

#FIVE [Bouville et al., 2015], également développé dans notre équipe, fait partie des modèles objets-relations et propose ces propriétés. Dans ce chapitre, nous présentons les fonctionnalités de #SEVEN en tant que modèle de scénarios pour environnements virtuels en nous appuyant sur les propriétés de #FIVE. #SEVEN se base sur des bibliothèques d'entités spécialisées pour permettre son intégration dans un environnement. Dans ce Chapitre nous présentons les éléments permettant de relier #SEVEN à #FIVE ainsi que les différentes propriétés qu'ils offrent, ensemble, pour la spécification et l'exécution de scénarios pour environnements virtuels. La Section 1 propose une présentation rapide de #FIVE. La Section 2 présente l'utilisation de #SEVEN dans un environnement #FIVE et leurs propriétés conjointes. La Section 3 s'intéresse plus précisément aux différents niveaux de guidage des acteurs possibles et La Section 4 s'intéresse à la gestion des ressources.

1 Le modèle pour environnements virtuels #FIVE

#FIVE repose sur un modèle permettant de décrire et de raisonner sur les objets qui peuvent participer à l'exécution d'une action.

- Un ***Objet #FIVE*** est une entité de l'environnement virtuel, pouvant avoir, ou non, une représentation 3D.
- Chaque objet possède un ou plusieurs ***types*** qui représentent ses propriétés. Par exemple, "*tranchant*" ou "*en acier*".
- Un ***Patron d'objet*** décrit un ensemble de types.
- Une ***Relation*** modélise une action. Elle décrit, par des patrons d'objets, les types nécessaires à différents objets pour pouvoir participer à cette action. Par exemple, la relation "*Inciser*" nécessite un objet "*tranchant*" et un objet "*découpable*". Un objet scalpel, qui porte le type "*tranchant*", est donc éligible pour participer à une réalisation de la relation "*inciser*" en tant qu'un des deux objets. Une relation ne vérifie pas la cohérence des états et des propriétés des objets. Ceci est délégué à la "*réalisation*".

- Une ***réalisation*** est une instance concrète de relation. Elle vérifie que l'état des objets rend l'action modélisée possible et exécute effectivement cette action. Par exemple, une lampe peut participer à l'action "*Allumer*" car elle est de type "*Allumable*". Néanmoins, la réalisation Allumer avec une lampe A ne peut être exécutée que si A n'est pas déjà allumée. Si la réalisation est exécutable, la lampe sera ensuite à l'état allumée. Elle ne pourra donc plus participer à une réalisation "*Allumer*", mais pourra participer à une relation "*éteindre*".

Le moteur de relations permet d'obtenir les réalisations possibles à partir d'un ensemble d'objets et de relations. Il gère un ensemble de requêtes que le système peut utiliser. #FIVE propose également un modèle d'interaction collaborative. Ce dernier permet un niveau de collaboration entre acteurs allant jusqu'à la comanipulation d'objets. L'objet de ce manuscrit n'est pas de présenter #FIVE, nous n'irons donc pas plus en détail dans son fonctionnement. Pour plus d'informations, #FIVE est présenté plus en détail dans l'Annexe B. #FIVE propose un ensemble d'interfaces, permettant d'implémenter ces concepts dans des moteurs bas niveaux gérant les éléments comme la physique, le rendu. Dans nos cas d'utilisation, nos systèmes reposent sur Unity3D [Technologies, 2016].

2 #SEVEN dans un environnement #FIVE

L'utilisation d'un moteur #SEVEN dans un environnement #FIVE repose sur trois classes d'entités principales, fournies dans une bibliothèque #SEVEN spécifique :

- La ***classe de contexte externe*** gère les entrées/sorties entre le moteur de scénarios et l'environnement. Dans le cadre d'un environnement #FIVE il réalise plusieurs fonctions.
  - Il offre ***un accès aux entités dans l'environnement***, il est ainsi possible de connaitre l'état d'un objet ou d'un acteur.
  - Il permet ***un accès aux relations existantes*** dans l'environnement.
  - Il offre ***un accès au moteur de relations***, qui permet notamment de créer des réalisations en utilisant les objets et les relations.
- ***Les senseurs de relations*** vérifient si une réalisation répondant à un ensemble de critères a eu lieu. Ces critères sont définis par les paramètres du senseur qui sont
  - "*Relation*" - Une relation #FIVE
  - "*Objects*" - un ensemble (potentiellement vide) d'objets associés.

  Par exemple, une relation "*Visser*" implique un objet de type "*Vissant*", un objet de type "*Vissable*" et un objet de type "*Emplacement*". Si l'environnement contient deux vis $V1, V2$, deux emplacements $E1, E2$, un tournevis T et une visseuse électrique S il existe 8

réalisations possibles de l'action Visser. Un senseur, dont le paramètre "*Relation*" indique la relation "*Visser*", attendra l'exécution d'une de ces réalisations. Dans la spécification des scénarios (XML réalisé par synthèse ou par l'outil auteur), préciser les paramètres "*Objects*" permet de réduire l'ensemble des réalisations possibles :

- préciser la vis ou l'emplacement réduit l'ensemble à quatre réalisations possibles. Par exemple $\{V1;E1;S\}$,$\{V1;E1;T\}$,$\{V2;E1;S\}$,$\{V2;E1;T\}$ répondent à la condition {Relation:Visser; Object:$E1$}
- Préciser les deux les réduit à deux réalisations. Par exemple $\{V2;E1;S\}$,$\{V2;E1;T\}$ répondent à la condition {Relation:Visser; Object:$E1$;Object:$V2$}.
- Préciser en plus que l'on utilise la visseuse réduit les réalisations possibles à une seule. Par exemple, seule $\{V2;E1;S\}$ répond à la condition {Relation:Visser; Object:$E1$;Object:$V2$;Object:S}.

Cette fonctionnalité est issue des possibilités offertes par le moteur de relations #FIVE. Les senseurs de relations l'utilisent au travers du contexte externe. Les acteurs (utilisateurs ou agents virtuels) sont intégrés dans le raisonnement des relations en tant qu'objets du type "*Acteur*". Ainsi si une action (c.-à-d. Réalisation) doit être exécutée par un acteur précis, l'instance d'acteur est spécifiée dans le paramètre "*objects*" du senseur. Si n'importe quel acteur peut réaliser cette action, le paramètre n'est simplement pas renseigné. Toutes les réalisations possibles seront alors supervisées par le senseur. Le senseur et le moteur de relations utilisent le type acteur pour raisonner. Ils ne font pas de différence entre un acteur contrôlé par un utilisateur et un acteur purement virtuel. Lorsqu'un senseur de relation est validé, son *contexte évènementiel* (c.f. Chapitre 3) retourne la réalisation correspondante. Ainsi, il est possible de l'utiliser pour configurer la suite de l'ensemble des scénarios au travers du *contexte local* ou du *contexte de scénarios*. Par exemple, s'il faut continuer d'utiliser le même outil pour visser d'autres éléments, l'outil utilisé dans la réalisation "*Visser*", capturée par le senseur, pourra être ajouté aux jetons et passé dans la suite du réseau.

- Les ***Effecteurs de Relations*** permettent d'agir sur l'environnement en déclenchant des réalisations. La réalisation à exécuter est définie par les paramètres de l'effecteur :
  - "*Relation*" - donne le type de relation à réaliser
  - "*Objects*" - associe à chaque patron d'objet de la relation un objet de l'environnement. Tous les patrons doivent être renseignés.

  L'effecteur interroge le moteur de relation, au travers du *contexte externe*, pour obtenir la réalisation correspondante. Cette réalisation est ensuite exécutée. Ainsi, le moteur de scénarios est capable d'agir sur l'environnement. La Figure 4.1 présente le principe d'utilisation des relations #FIVE par un moteur #SEVEN.

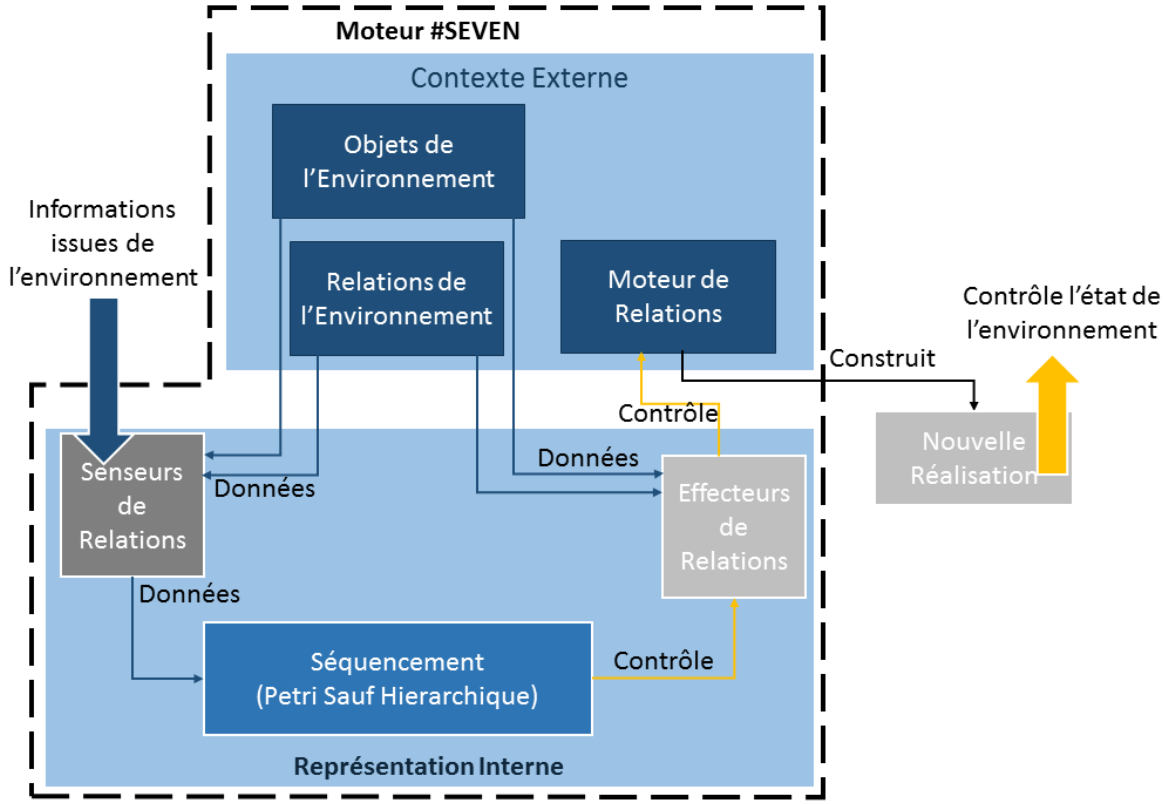


Figure 4.1 – Fonctionnement des composants principaux de la bibliothèque #SEVEN pour son intégration dans un environnement #FIVE.

Puisque le contexte externe offre un accès aux objets #FIVE, différentes classes de senseurs spécialisés peuvent vérifier la réalisation de conditions liées à leur état. Dans la suite de ce Chapitre, nous appelons ces senseurs des ***senseurs d'état***. Chaque classe permet de vérifier un ensemble de conditions liées à l'environnement. Voici une liste non exhaustive d'exemples de classes de senseurs :

- Les *senseurs de positions* vérifient qu'un objet se situe dans une zone spécifique dans la scène 3D,
- Les *senseurs de collision* vérifient que des objets sont en interpénétration.
- Les *senseurs de visibilité* vérifient qu'un acteur possède un objet donné dans son champ de vision.
- Les *senseurs de ressources* vérifient qu'un objet, nécessaire à la réalisation d'une action, n'est pas déjà utilisé.

Différents types de senseurs associés aux types de bases (p. ex. booléens, flottants) permettent de vérifier la valeur de certains attributs des objets. Par exemple un flottant qui indique la luminosité d'une lampe peut-être vérifiée par un senseur spécialisé dans les flottants. Ces senseurs font partie des bibliothèques de base #SEVEN et ne sont pas spécifiquement développés pour interagir avec #FIVE. Néanmoins, l'accès aux objets du monde n'est possible que grâce au contexte externe fourni dans la bibliothèque spécialisée pour #FIVE.

3 Niveaux de guidage des acteurs

Le niveau de guidage du moteur de scénarios sur les acteurs dépend des besoins liés à l'utilisation de l'environnement virtuel (c.f. Chapitre 2). Dans certaines simulations, il est préférable de laisser toute liberté d'action aux acteurs alors que dans d'autres il vaut mieux contraindre leurs actions. Par exemple, il est possible de former un utilisateur à une procédure en trois phases :

- Dans une première simulation en lui offrant uniquement la possibilité d'exécuter les actions correctes,
- dans un second temps en lui conseillant les actions correctes, mais en lui laissant possibles les autres actions,
- dans un troisième temps en lui laissant toute latitude dans le choix et l'exécution des actions pour évaluer ses connaissances.

Dans la littérature, les solutions existantes ne s'intéressent qu'à un ou deux niveaux de guidage. Pour mettre en place ce type de scénarios pédagogiques (descriptions des différentes étapes d'une formation), il faudrait donc utiliser plusieurs technologies différentes.

#SEVEN et #FIVE permettent d'adapter le niveau de guidage des acteurs en utilisant différentes spécifications de scénarios et en adaptant les informations fournies aux acteurs par le moteur de scénarios. Le fonctionnement général est présenté dans la Figure 4.2. Les réalisations possibles sont fournies au moteur de scénarios. Le moteur de scénarios informe ensuite les acteurs sur les conditions nécessaires pour avancer dans la simulation. Ces conditions peuvent être de deux de deux natures différentes :

- Réalisations possibles : ce cas l'acteur choisit parmi ces réalisations celle qu'il va exécuter.
- États à atteindre : dans ce cas l'acteur doit lui-même trouver une chaine d'actions qui permet d'atteindre cet état parmi les réalisations fournies.

Pour s'adapter au besoin, le moteur de scénarios propose trois modes d'utilisation, choisie dans sa configuration (description donnant notamment le fichier contenant la spécification des scénarios possibles et également le mode de fonctionnement utilisé) :

- ***Le mode bloquant*** ne fournit aucune réalisation aux acteurs,

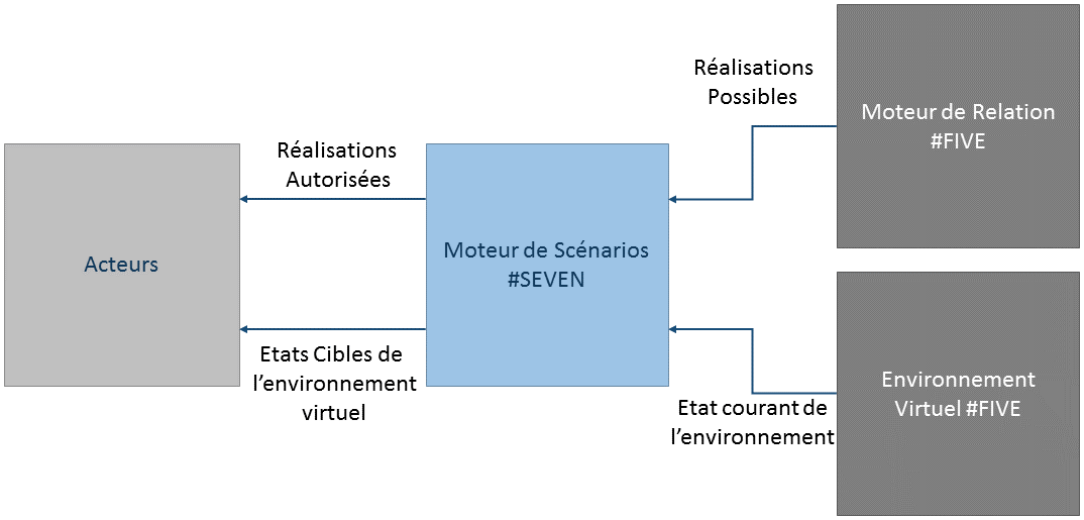


Figure 4.2 – Flux d'informations entre le moteur de relation, le moteur de scénarios et les acteurs.

- ***Le mode filtrage*** utilise les senseurs de relations actifs (dont la transition est sensibilisée) pour ne fournir aux acteurs que les réalisations exécutables qui correspondent à au moins une validation de condition d'un senseur.

- ***Le mode ouvert*** retourne toutes les relations exécutables fournies par le moteur de relations.

La configuration du moteur de scénarios définit également s'il fournit aux acteurs les autres conditions attendues, définies par les différents senseurs d'états.

Nous proposons quatre niveaux de guidage possibles qui dépendent du mode de fonctionnement du moteur de scénarios et de la spécification des scénarios possibles. Les niveaux "*non-guidé*" et "*Guidage faible*" supposent que le moteur de scénarios est en mode "*ouvert*". Les planifient alors eux-mêmes leurs actions. Les niveaux "*Guidage fort*" et "*Prise de contrôle*" supposent que le moteur est en mode "*filtrage*".

- ***Non Guidé*** : Le moteur de scénarios n'informe pas les acteurs et ne les guide pas dans leurs activités. Son rôle est uniquement de superviser l'état de l'environnement et de déclencher des actions si nécessaire.

- ***Guidage faible*** : Le moteur fournit les fonctionnalités du niveau "*non guidé*", mais indique les états que l'environnement doit atteindre pour passer à la suite. Par exemple, il indique que la machine doit être mise en route, mais n'indique par l'action à réaliser.

- ***Guidage fort*** : Le moteur fournit les fonctionnalités du niveau "*non guidé*", mais indique les relations à réaliser pour avancer.

- ***Prise de contrôle*** : Le moteur fournit les fonctionnalités du niveau "*Guidage fort*", mais les acteurs sont contrôlés dans certaines de leurs actions par le moteur de scénarios. Plutôt que d'indiquer ce qu'il y a à faire grâce un senseur, un effecteur déclenche certaines actions à la place des acteurs.

#SEVEN permet de fournir des niveaux intermédiaires grâce à l'ajout d'attributs (c.f. Chapitre 3, Section 3). Nous les appelons, niveaux intermédiaires, car ce sont des extensions des niveaux existants. À par les ajouts supplémentaires, les caractéristiques de leurs spécifications et le mode utilisé restent identiques à celles du niveau de base.

Une spécification de scénarios de niveau de "*guidage fort*" peut être étendue pour porter des attributs qui définissent une suite d'actions comme faisant partie d'une même activité. Ainsi, le déclenchement de la première action de la séquence par l'acteur peut être suivi par un déclenchement automatique des actions suivantes. Ce déclenchement automatique n'est pas obligatoire, contrairement au niveau "*prise de contrôle*", mais fournit une assistance à l'acteur. La suite d'actions peut être déclenchée directement par l'acteur ou par un autre composant du système comme un moteur pédagogique. Ceci peut être utile par exemple pour un utilisateur en formation qui veut pouvoir donner le contrôle au système dans une séquence d'actions précise, mais préfère rester assez libre dans ses actions le reste de la simulation.

La Figure 4.3 donne un exemple de spécification de scénarios à "*guidage fort*" utilisant des senseurs de relations. Ici la tâche est de réaliser un montage grâce à quatre vis. L'action "*Visser*" utilise une "*vis*" et un "*emplacement*" parmi A, B, C et D. tous les ordres de montage ne sont pas valides : les acteurs doivent réaliser les actions dans un des ordres prévus. Les vis sont montées en croix, donc l'emplacement A est suivi de l'emplacement C ou inversement, B est suivi de D ou inversement. Ce type de spécification peut être utilisé, par exemple, pour former des utilisateurs à une nouvelle procédure. Cette spécification de scénarios est utilisée pour un guidage fort.

La Figure 4.4 donne un exemple de spécification de scénarios qui utilise des senseurs d'états. Dans ce cas, les senseurs vérifient uniquement que les emplacements A, B, C et D ont bien tous reçu une vis. Les actions à réaliser ne sont pas définies, leur choix est à la charge des acteurs. Cette spécification est utilisée pour des niveaux de guidage absent ou faible. Les acteurs ne sont pas contraints dans la réalisation de l'activité. Seul l'état final du monde est pris en compte. Ce type de spécifications peut être utilisé par exemple pour de l'évaluation de compétence. Les acteurs sont libres, mais leurs actions peuvent être tracées, il est possible de vérifier en parallèle ou à postériori qu'elles ont été réalisées dans un ordre précis. Par exemple comme prévu dans les scénarios de la Figure 4.3

Il est possible de réaliser une spécification de scénarios qui utilise les senseurs de relation pour mettre en place un niveau de guidage équivalent à une absence de guidage ou à un guidage faible. Néanmoins, ceci demande de prévoir toutes les combinaisons possibles d'actions réalisables par les acteurs. Si ce n'est pas le cas, il y a de grands risques de perdre la cohérence entre l'état de la représentation interne du moteur de scénarios et l'état de l'environnement. Ceci est dû à la possibilité qu'ont les acteurs de réaliser n'importe quelles actions fournies par le moteur de

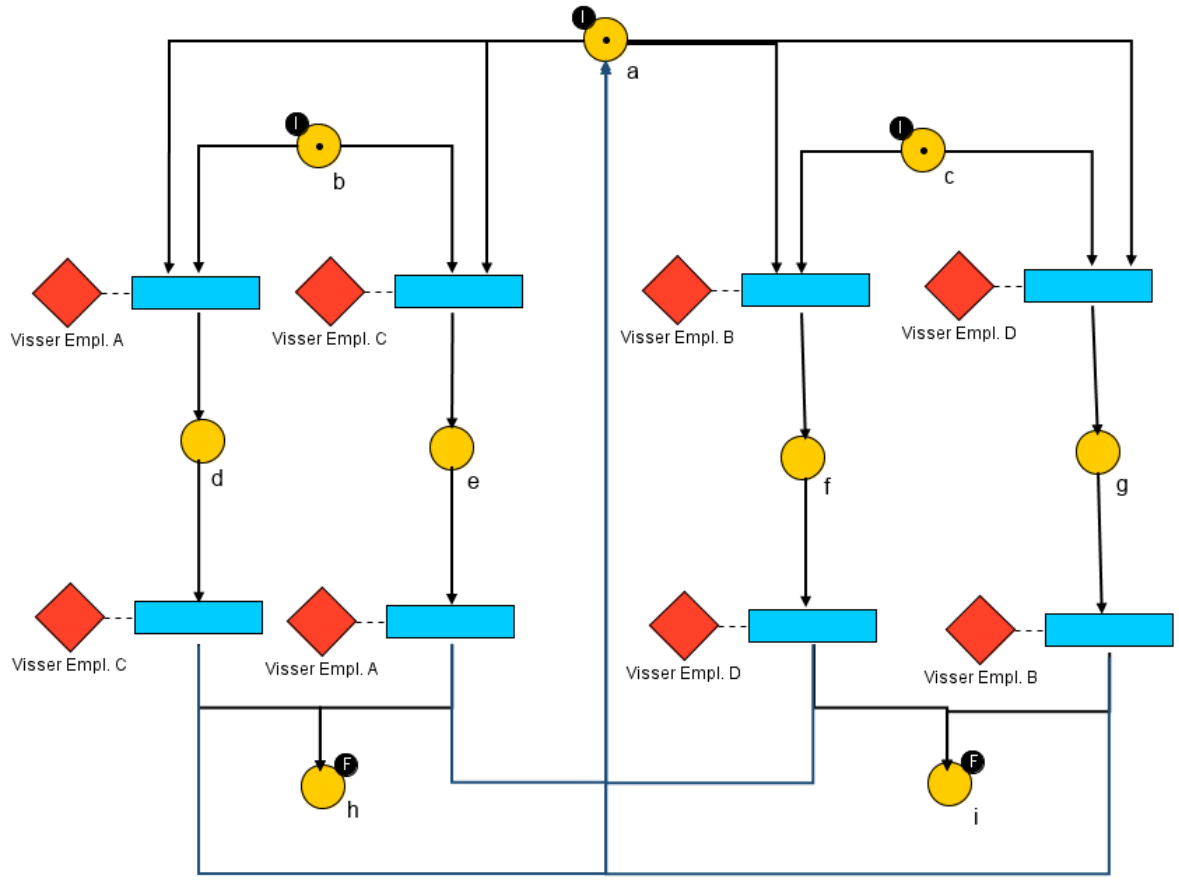


Figure 4.3 – Exemple de spécification de scénarios utilisée pour un guidage fort des actions des acteurs.

relations. Par exemple, si la spécification de la Figure 4.3 est utilisée sans guidage ou avec un guidage faible, l'acteur a accès à toutes les actions fournies par le moteur de relations ("*visser A*", "*visser B*", "*visser C*", "*visser D*"). Dans ce cas, après avoir réalisé l'action "*visser A*", rien ne l'empêche de réaliser "*visser B*" à la place de "*visser C*". Ce cas n'est pas prévu par le réseau, le moteur de scénarios ne met donc pas à jour son état, puisqu'il s'attend à "*visser C*". L'état du moteur de scénarios devient donc incohérent avec l'état de l'environnement.

Une spécification de scénarios utilisant des senseurs d'états pour réaliser un guidage fort n'est pas envisageable, car les acteurs doivent réaliser de la planification d'actions pour répondre à chaque condition définie. Ils choisissent donc eux même les actions à réaliser. Par conséquent, il n'est pas exclu qu'une de ces actions réponde en avance à une condition qui n'est vérifiée que plus tard. Il est donc difficile de contrôler l'agencement temporel des actions des acteurs.

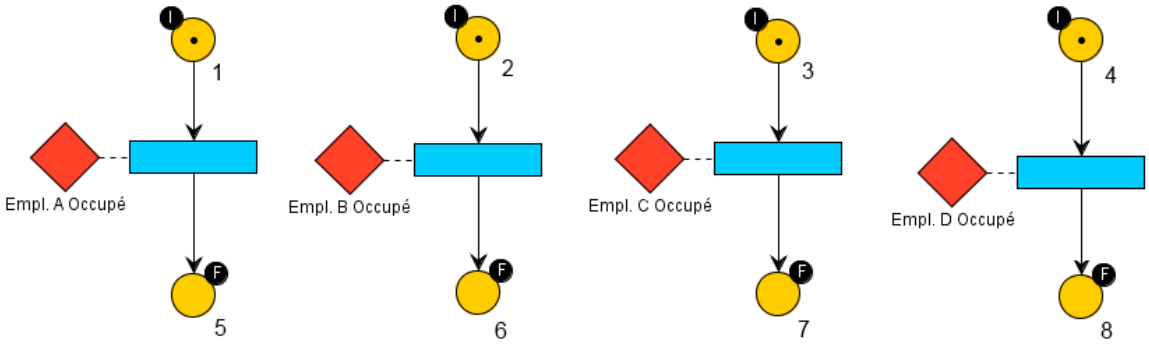


Figure 4.4 – Exemple de spécification de scénarios utilisée pour un guidage faible ou absent des actions des acteurs.

4 Gestion des Ressources

La problématique de ***gestion des ressources*** est prise en compte à différents niveaux par #FIVE ou par #SEVEN. Nous appelons ressource tout élément faisant partie de l'environnement, disponible en quantité limitée à un instant donné et participant à la réalisation des actions. Puisque les ressources existent en quantité limitée, elles interviennent dans la synchronisation des actions. Par exemple, si un outil est demandé par deux acteurs en même temps, mais qu'il n'y en a qu'un dans l'environnement, ils ne pourront pas agir en même temps. Dans le cas où les actions réduisent le nombre de ressources disponibles (par destruction ou par changement d'état), cette synchronisation a un impact fort sur le déroulement de la simulation puisque certaines actions ne pourraient ne jamais être réalisées.

Les ressources sont des objets ou des parties d'objets #FIVE, même si elles n'ont pas de représentation physique. Par exemple une quantité de puissance électrique fournie par un générateur. #FIVE les prend en compte dans les relations et dans les réalisations :

- Les relations indiquent le type d'une ressource nécessaire.
- Les réalisations vérifient que l'état de la ressource correspond aux contraintes de l'action modélisée.

Par exemple, une action "*prendre un objet*" nécessite un objet à prendre et une main libre. Si la main (partie de l'acteur qui réalise l'action) n'est pas libre ou que l'objet est déjà pris, il sera impossible de réaliser l'action. #FIVE est responsable de la gestion des accès aux ressources. Si deux acteurs tentent de réaliser en même temps une réalisation qui nécessite une même ressource, alors #FIVE doit prendre en compte la gestion des accès concurrents, en rejetant ou en mettant en attente une des deux réalisations.

Une propriété intéressante fournie par des modèles comme LORA++ est la représentation des séquences d'activités. Il s'agit d'une suite d'actions qui font toutes références à une même ressource, définie lors de l'exécution de la première action. Par exemple, si un acteur exécute la première action d'une tâche, il est possible de définir qu'il doit la réaliser jusqu'au bout. #SEVEN permet de définir ce type de séquences grâce aux contextes locaux et aux contextes évènementiels. Lors de la validation d'une condition d'un senseur de relation, son contexte évènementiel retourne la réalisation correspondante et permet donc de connaitre les ressources utilisées pour configurer la suite des scénarios possibles. Les ressources utilisées peuvent ensuite être passées du contexte évènementiel au contexte local. Il est donc possible de décrire des séquences d'actions comme devant utiliser jusqu'au bout une même instance de ressource. Ainsi, si l'acteur n'est pas précisé pour la première action, il est possible de l'extraire de la réalisation obtenue par le contexte évènementiel puis de le passer en référence dans les paramètres des senseurs suivants. Le filtrage des réalisations ne prendra plus en compte que les actions impliquant cet acteur.

5 Synthèse

Nous avons présenté les possibilités offertes par l'utilisation de #SEVEN dans un environnement #FIVE. L'utilisation conjointe des deux modèles est rendue possible grâce à l'utilisation des bibliothèques #SEVEN. Ensemble, ils permettent la spécification et l'exécution de scénarios en répondant à toutes les propriétés énumérées dans le Chapitre 2 :

- Les agencements complexes d'activités sont réalisables grâce aux propriétés de séquencement de #SEVEN (offertes par le modèle de réseaux de Petri) et la généricité de la représentation des actions proposée par #FIVE.
- La gestion des ressources est intégrée au niveau des deux modèles. #FIVE permet la représentation et le fonctionnement du système de ressources et #SEVEN inclut leur utilisation dans la spécification des scénarios.
- Les actions modélisées par #FIVE peuvent être exécutées par #SEVEN par l'intermédiaire d'une classe d'effecteurs spécialisée.
- Dans l'utilisation que #SEVEN fait du modèle de relations de #FIVEN, les acteurs sont impliqués dans l'environnement au même titre que les objets. #FIVE permet la spécification d'actions collaboratives et ne fait pas de différence entre des acteurs réels ou virtuels. Une réalisation peut être exécutée par un nombre quelconque d'acteurs (c.f. Annexe B).
- #FIVE représente directement les objets et les actions (réalisations, instances de relations entre objets). Ces concepts sont utilisés tels quels dans la spécification des scénarios

#SEVEN. Cette spécification utilise la représentation graphique fournie par les réseaux de Petri #SEVEN.

- L'utilisation de #SEVEN au dessus de #FIVE permet la spécification de scénarios pour différents niveaux de guidage de l'activité des acteurs. Les niveaux de guidage offerts vont de la liberté d'action totale sans assistance à la prise de contrôle des actions de l'acteur par le moteur de scénarios. Il est également possible de laisser l'acteur agir en lui conseillant une marche à suivre grâce à l'ajout informations supplémentaires.

En tant que modèle de scénarios pour environnements virtuels, #SEVEN (reposant sur le modèle #FIVE) propose l'ensemble des propriétés énoncées. Son principal avantage est qu'il est capable de s'adapter à l'utilisation de l'environnement virtuel, principalement en permettant de fournir différents niveaux de guidage, là où il serait nécessaire d'utiliser différents modèles. Il est également possible de l'utiliser pour décrire des scénarios sur la base de modèles pour environnements virtuels différents.

La réalisation d'activités collaboratives repose sur une organisation temporelle et causale des actions et sur la répartition de ces actions entre les différents acteurs [Gerbaud, 2008]. Dans ce Chapitre, nous avons traité de l'agencement des actions. Le Chapitre suivant s'intéresse à la répartition des actions entre les acteurs.

Distribution des Actions aux Acteurs 5

En accord avec les travaux de Gerbaud et al. [Gerbaud et al., 2009] nous considérons la distribution des actions aux acteurs comme ayant un impact important sur le déroulement de la simulation. En fonction des possibilités d'action des différents acteurs, la simulation peut se passer de manières totalement différentes. Par exemple, un ouvrier expérimenté devrait avoir accès à certaines actions plus difficiles à réaliser, mais qui permettent de gagner du temps, alors qu'un débutant ne devrait accéder qu'aux actions les moins difficiles.

En réalité virtuelle, la solution commune est de représenter la notion de rôle. Dans le Chapitre 2 nous avons défini trois propriétés qu'un modèle de rôles doit posséder :

- Il doit être capable de prendre en compte des concepts liés à l'acteur et qui permettent de connaitre son implication dans la simulation,

- Il doit être capable de faire évoluer les rôles des acteurs en accord aux changements ayant lieu dans la simulation. Notamment en tenant compte des règles définies dans le groupe d'acteurs.

- Les spécifications de rôles et de règles d'évolutions doivent pouvoir être réutilisées d'une simulation à l'autre.

Nous supposons que le modèle de scénarios utilisé permet à un acteur d'obtenir un ensemble d'actions réalisations en fonction de l'état de l'environnement virtuel et de l'avancement de la simulation à un instant donné. Un ***modèle de rôle*** permet à un acteur d'obtenir un sous-ensemble de ses actions. Ce sous-ensemble est le résultat d'une fonction de filtrage sur l'ensemble des actions possibles en fonction du rôle de l'acteur. Chaque rôle définit donc un sous-ensemble différent. Puisque le rôle d'un acteur peut évoluer au cours de la simulation, le sous-ensemble des actions auxquelles il a accès peut être modifié, même si l'ensemble de toutes les actions possibles reste identique.

Dans la Section 1 nous présentons le fonctionnement de notre système de filtrage des actions capable de prendre en compte différents concepts liés à l'acteur. La Section 2 propose un modèle pour l'évolution des rôles au cours de la simulation.

1 Système de filtrage des actions

Les travaux en théorie des rôles [Biddle and Thomas, 1966] définissent le rôle comme un ensemble de concepts permettant de raisonner sur ce qu'un acteur peut ou doit faire. Cette approche est utilisée dans certains travaux de storytelling [Cavazza et al., 2002; Szilas, 2003], mais est directement intégrée dans les actions. Ceci pose un problème pour réutiliser les actions dans des simulations différentes. D'autres travaux (p. ex. [Gerbaud et al., 2007], [Chevaillier et al., 2012]) proposent d'ajouter ces informations dans la spécification des scénarios. Ainsi, les actions restent génériques et le scénario est spécialisé pour des rôles précis.

Pour permettre la réutilisation des actions tout en nous rapprochant de la théorie des rôles, nous proposons de combiner ces deux approches. Pour cela nous ajoutons des attributs sur les actions (c.f. Chapitre 3) décrivant des conditions d'accès. Ces attributs sont ensuite utilisés par un filtre pour obtenir le sous-ensemble des actions qui concerne le rôle de l'acteur. La Figure 5.1 présente le fonctionnement général du filtre pour deux acteurs ayant des rôles différents. L'acteur 1 possède le rôle A et l'acteur 2 possède le rôle B. Le filtre calcule un sous-ensemble différent pour chaque acteur, car ils ne possèdent pas le même rôle. Si les rôles A et B sont identiques, alors les deux acteurs recevront le même ensemble d'actions. C'est alors à eux se mettre d'accord pour répartir et/ou synchroniser leurs actions. Si jamais ils décident de réaliser tous les deux la même action au même moment, c'est alors à l'environnement virtuel (ici #FIVE)de gérer les accès concurrents.

- Le filtrage est réalisé suivant un ensemble de concepts liés à l'acteur et non par un simple identifiant de rôle. Pour cela, nous proposons de définir le rôle comme un ensemble d'assertions décrivant l'acteur à un moment donné. Ces assertions peuvent être de différentes natures :

  - ***Internes à l'acteur*** : par exemple ses compétences, ou les droits qui lui sont accordés
  - Liées à ***l'acteur et à l'environnement*** : par exemple la position de l'acteur dans la scène, les ressources auxquelles il a accès

  Ces informations, liées au domaine d'application de la simulation et à l'acteur, peuvent avoir un impact sur le déroulement de situations. Leur spécification peut se baser sur des formalisations reconnues du domaine et/ou sur l'expérience des spécialistes. Par exemple, l'algorithme 4 définit le rôle d'un chirurgien en train de réaliser une procédure dans la salle d'opération. Dans un domaine de travail comme la chirurgie, les informations liées au rôle d'un acteur peuvent être par exemple :

  - son niveau de compétence ou de connaissance sur un sujet (utilisation d'un instrument, réalisation d'une technique particulière),

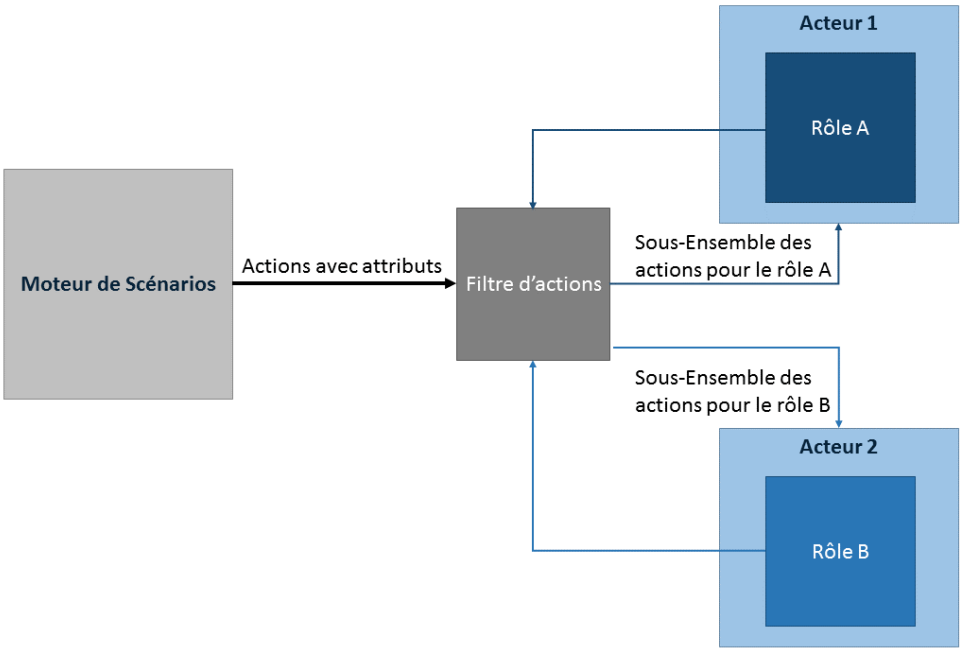


Figure 5.1 – Principe de base du filtrage d'actions.

  - l'école dans laquelle il a été formé (qui peut influer, par exemple, sur ses habitudes),
  - des parties de procédures qu'il a actuellement le droit de réaliser suivant son niveau de qualification (p. ex. apprenti, assistant, responsable).

- Le filtrage permet de fournir différents ensembles d'actions. L'appartenance d'une action à un ou plusieurs ensembles ajoute de l'information à son sujet à destination de l'acteur. Par exemple :

  - "*Les actions techniquement réalisables/impossibles*" : une action impossible ne peut simplement pas être réalisée par l'acteur.
  - "*Les actions autorisées/interdites*" : une action interdite peut être techniquement réalisable, mais l'acteur n'a pas le droit de l'exécuter.
  - "*Les actions prioritaires/secondaires*" : Une action peut-être plus ou moins urgente du point de vue de l'acteur. Ces priorités peuvent être différentes d'un acteur à un autre.

  Pour cela, le principe des attributs associés aux actions doit également être étendu afin de prendre en compte les définitions de rôle et pour définir différents sous-ensembles des actions. Un attribut lié au filtrage décrit :

- ***Un identifiant de sous-ensemble d'actions*** auquel l'action peut appartenir. Ce sous-ensemble possède une sémantique, connue de l'acteur, qui indique ce que l'acteur peut faire avec ces actions. Par exemple, un sous-ensemble "*Actions autorisées*" indique à l'acteur qu'il a le droit de réaliser les actions qui font partie de ce sous-ensemble.
- ***Une condition*** décrivant sous quelles conditions l'action qui porte cet attribut est ajoutée au sous-ensemble. Ces conditions portent sur le contenu du rôle de l'acteur. Par exemple, un acteur peut avoir le droit de réaliser une action que s'il maitrise l'utilisation de l'instrument nécessaire à sa réalisation.

Par exemple, l'attribut suivant décrit que l'action qui la porte ne doit être réalisée que par un chirurgien expert dans l'utilisation du trépan et uniquement s'il est celui qui exécute les tâches principales lors de la procédure.

- Sous-ensemble : Actions autorisées
- Condition : Expert en utilisation du Trépan et Chirurgien principal

Le filtre utilise les informations issues du rôle d'un acteur pour évaluer les conditions définies sur les attributs de l'action. Dans certains cas, l'échec d'interprétation d'une condition peut inclure l'action dans un autre sous-ensemble. Par exemple, le filtre peut gérer les Actions autorisées de la manière suivante :

- Si l'action ne possède pas d'attribut pour le sous-ensemble "*Action autorisée*" elle est y est ajoutée par défaut.
- Si l'action possède un attribut pour le sous-ensemble "*Action autorisée*" et que l'interprétation de la condition est vraie, elle est ajoutée à cet ensemble.
- Si l'action possède un attribut pour le sous-ensemble "*Action autorisée*" et que l'interprétation de la condition est fausse, elle est ajoutée à l'ensemble "*Actions interdites*".

Une action peut porter un nombre quelconque d'attributs. Elle peut ainsi être ajoutée à plusieurs sous-ensembles. Par exemple, une action peut être techniquement réalisable par un acteur (ensemble "*Actions possibles*"), mais faire partie des "*Actions interdites*". Les possibilités offertes aux acteurs en fonction des ensembles dans lesquels sont présentes les actions sont définies dans le système de décision de l'acteur (p. ex. un moteur #SEVEN utilisant des bibliothèques de composants spécialisés dans le comportement des acteurs). Par exemple, il est possible de laisser la liberté aux acteurs d'outrepasser leurs droits en cas d'urgence. L'acteur peut décider de réaliser une action interdite, mais techniquement possible. Ainsi, lors d'une opération chirurgicale, l'interne peut être amené à réaliser des actions réservées au chirurgien dans le but de sauver le patient, car le chirurgien est débordé.

Algorithm 4 Exemple de définition du rôle d'un acteur chirurgien.

```
<?xml version="1.0" encoding="utf-8" ?>
<actor xmlns="http://www.insa-rennes.fr/actor"
    classname="FSSurgeon, Assembly-CSharp">
  <role key="SKILL_USE_Trephine" value="EXPERT"/>
  <role key="SKILL_USE_Bi-Polar_Pliers" value="EXPERT"/>
  <role key="SKILL_Aneurysms_Cauterisation" value="INTERMEDIATE"/>
  <role key="SKILL_Trepanning" value="EXPERT"/>
  <role key="RIGHT_Perform_Incision" value="TRUE"/>
  <role key="RIGHT_Perform_Skull_Opening" value="FALSE"/>
  <role key="Training_School" value="RENNES"/>
  <role key="RESOURCE_Right_Hand" value="SCALPEL"/>
  <role key="RESOURCE_Left_Hand" value="PLIERS"/>
  <role key="RESOURCE_Self" value="BUSY"/>
  <role key="SCENE_POSITION" value="OPERATING_ROOM"/>
</actor>
```

Le niveau de détail utilisé pour définir les rôles des acteurs, et par conséquent les conditions, peut être adapté à la simulation. Par exemple, le rôle d'un chirurgien peut définir ses compétences dans l'utilisation des divers instruments (p. ex. Scalpel, trépan, pinces...) ou simplement indiquer un niveau général de maitrise des gestes techniques chirurgicaux. Ceci dépend des besoins liés à la répartition des actions entre les acteurs et au niveau de détail nécessaire. De plus, un niveau de détail très fin demandera beaucoup plus de travail de spécification, que ce soit au niveau des rôles ou des conditions. Le niveau de détail n'est pas imposé, mais il doit être cohérent entre la spécification du rôle et la spécification des conditions portées par les attributs des actions.

2 Évolution des rôles

Selon les travaux en théorie des rôles [Biddle and Thomas, 1966] et en accord avec certains modèles déjà existants dans la littérature (p. ex. Chevaillier et al. [Chevaillier et al., 2012]), le rôle d'un acteur peut évoluer au cours de la simulation. Ceci peut être dû à des changements dans son environnement physique ou social [Biddle and Thomas, 1966].

Certains changements sont directement liés aux actions des acteurs. Par exemple, si un acteur prend le volant d'un véhicule, il aura accès aux actions liées à la conduite. D'autres actions vont modifier indirectement le rôle d'un acteur. Par exemple, lors d'une procédure, l'acteur le plus proche de l'établi devient responsable de l'appui sur le bouton d'arrêt d'urgence. Si l'action "*marcher*" donne elle-même la responsabilité de l'utilisation de l'arrêt d'urgence à

l'acteur qui la réalise lorsqu'il s'approche de l'établi, elle n'est plus réutilisable dans une autre simulation. Cette modification du rôle n'est pas directement due à l'action marcher, mais à la relation que possède l'acteur avec son environnement physique et social (ici sa position dans l'espace par rapport à l'établi et à la position des autres membres de l'équipe). Il s'agit d'une règle associée à l'équipe.

Pour cela, nous proposons un modèle permettant de représenter les équipes d'acteurs et les comportements associés (p. ex. règles, habitudes) faisant évoluer les rôles des acteurs en fonction de

- leurs relations en tant que membres de l'équipe,
- leurs rôles respectifs,
- l'état de l'environnement,
- l'avancement de la procédure.

Tout en gardant la possibilité de modifier directement les rôles des acteurs grâce aux actions.

Ce modèle repose sur les entités supplémentaires suivantes :

- "*Les Postes*" représentent (partiellement) la situation d'un acteur vis-à-vis des objectifs et de la structure de l'équipe ainsi que de ses règles ou habitude.
- "*Les équipes*" sont des ensembles de postes. Une équipe fait partie de l'environnement. Elle réagit aux changements d'état de l'environnement et aux actions qui sont réalisées en modifiant son propre état ou les rôles des acteurs associés à ses postes.

Les relations fonctionnelles et structurelles entre les différentes entités sont représentées dans la Figure 5.2 : ***une équipe*** est un ensemble de ***postes***. Chaque poste est associée à au plus un acteur et à une seule équipe. Un acteur peut être associé à plusieurs postes. Le rôle de l'acteur est utilisé dans le fonctionnement des postes et de l'équipe. Il peut également être modifié par ces dernières. Un acteur est appelé membre d'une l'équipe s'il est associé à au moins un poste dans cette équipe.

Les équipes sont des entités réactives qui font partie de l'environnement virtuel. Elles fonctionnent suivant le principe de la boucle de perception (c.f. Chapitre 3). Une équipe peut être créée à partir d'une définition d'équipe. Il s'agit soit d'un fichier externe (p. ex. XML, Algorithme 5), soit d'une structure de données construite lors de la simulation. La définition d'équipe décrit comment sont associés les acteurs aux postes et définit les comportements possibles de l'équipe. Lors de sa création, une équipe associe des acteurs à des postes. Son comportement définit comment peuvent évoluer les rôles des acteurs associés à ces postes et sous quelles conditions ils peuvent être rattachés ou détachés de ces postes. Par exemple, si l'asepsie du chirurgien n'est plus garantie, il perd son poste de chirurgien principal.

L'attribution ou le retrait d'un poste à un acteur dépend de son rôle. Par exemple, un neurochirurgien ne pourra pas remplacer un anesthésiste, car ils ne possèdent par les mêmes

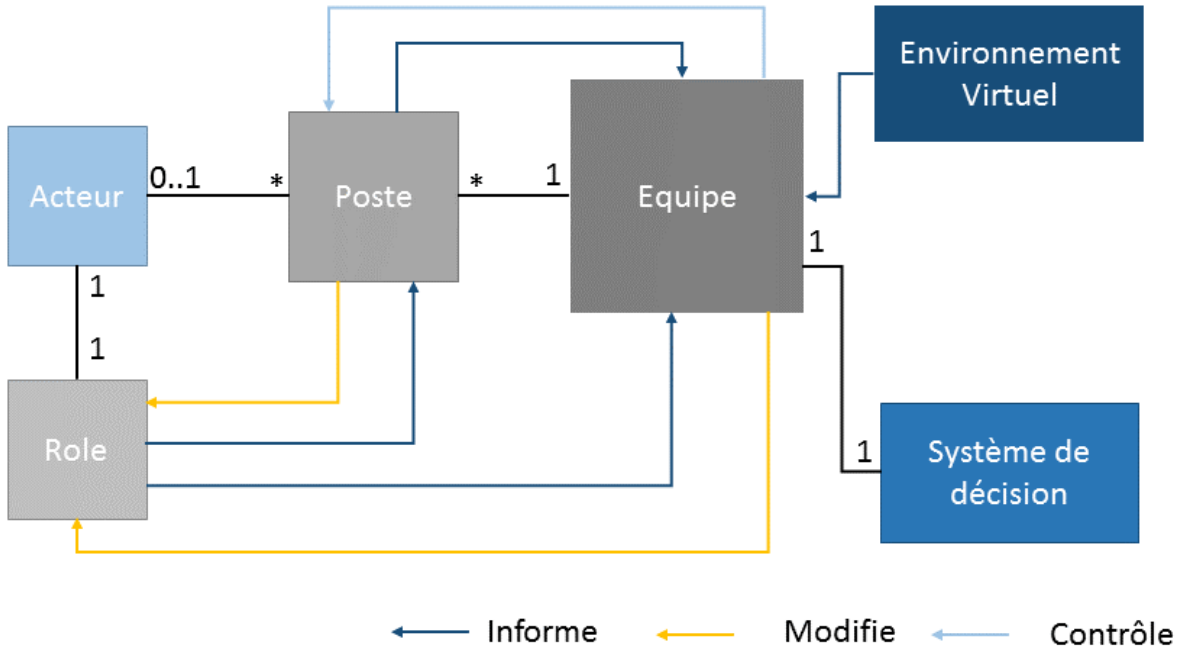


Figure 5.2 – Organisation fonctionnelle de notre modèle de rôles

compétences. De plus, le fait d'être attribué à un poste peut modifier le rôle d'un acteur. Par exemple, lorsqu'un acteur devient chef d'équipe, son rôle lui permet d'accéder aux actions permettant de donner des ordres. Pour gérer ce genre de situations, un poste propose plusieurs fonctionnalités : deux vérifications et deux comportements automatiques. Les vérifications permettent à différentes entités (p. ex. équipe, acteurs dans l'environnement virtuel) d'obtenir des informations au sujet d'un poste donné (p. ex. savoir qu'un acteur donné peut être associé à un poste). Les mises à jour sont déclenchées par le poste lui-même pour modifier les rôles des acteurs à certains moments.

- "*Le test d'entrée*" vérifie qu'un acteur peut être associé au poste. Il prend en compte l'état de l'équipe, le rôle de l'acteur entrant et le rôle d'un acteur déjà associé s'il existe. Par exemple, un poste "*Chirurgien*" dans une équipe médicale ne peut être occupé que par acteur dont le rôle décrit un chirurgien et si la poste n'est pas déjà occupée.
- "*le test de cohérence*" consiste à vérifier que le rôle de l'acteur associé répond à certains critères liés au poste. Par exemple, si le chirurgien entre en contacte avec un élément qui n'est pas aseptisé (p. ex. l'emballage d'un instrument), son rôle est mis à jour pour indiquer qu'il n'est plus aseptisé ce qui est incohérent avec son poste de chirurgien principal.
- "*La mise à jour d'entrée*" modifie le rôle d'un acteur au moment où il est associé au poste. Par exemple, un acteur peut obtenir le droit de donner des ordres aux autres acteurs, car il est devenu chef d'équipe.

- "*La mise à jour de sortie*" modifie le rôle d'un acteur au moment où il est dissocié d'un poste. Par exemple, un acteur ne peut plus donner d'ordre aux autres acteurs s'il n'est plus chef.

Un poste est une instance d'une classe qui implémente ces fonctionnalités. Lors de la création d'une équipe, chaque poste réalise un test d'entrée sur son acteur désigné puis utilise sa mise à jour d'entrée. Si le test d'entrée échoue, l'acteur n'est pas associé au poste. Durant la simulation, l'équipe peut à tout moment.

- Exécuter le test de cohérence,
- Retirer ou ajouter un acteur à un poste qui la compose.

Dans la réalité, l'organisation des acteurs n'est pas toujours constituée d'un seul groupe. Par exemple, deux équipes peuvent travailler sur un même projet, mais sur des tâches différentes. Pour permettre des représentations d'équipes plus complexes, une équipe contient des sous-équipes :

- Une sous-équipe est associée au même environnement virtuel que son équipe parente et possède son propre comportement.
- Les membres d'une sous-équipe doivent être membres de son équipe parente.
- Un acteur peut être membre de plusieurs sous-équipes.

Les sous-équipes sont créées par l'équipe parente, qui peut également modifier les membres de ses sous-équipes. L'organisation hiérarchique des équipes permet également de simplifier la spécification des règles d'évolution des rôles en offrant la possibilité de les diviser.

Les acteurs ne peuvent être associés à un poste d'une équipe que dans les conditions suivantes :

- Ils ont été associés lors de la création de l'équipe (indispensable s'il s'agit d'une équipe sans parents).
- Ils sont membres de l'équipe parente (indispensable pour les sous-équipes).

Pour simplifier la spécification et l'utilisation des équipes, nous conseillons de définir une équipe principale, possédant pour membre tous les acteurs de l'environnement. Cette équipe sera chargée à partir d'une définition réalisée dans un fichier externe. Son objectif est de répartir les acteurs dans des sous-équipes associées, par exemple, à des tâches précises. Ainsi, il est possible d'ajouter ou de retirer des acteurs dans les différentes équipes intervenant dans l'environnement. Par exemple, sur un chantier une équipe principale contient tous les acteurs et les différents ouvriers peuvent naviguer entre les différents postes, même s'ils ne sont pas directement liés (p. ex., monter un mur et réaliser du terrassement).

Enfin, une équipe peut ne plus être utile. Dans ce cas, elle peut être dissoute. Toutes ses sous-équipes sont alors dissoutes. Puis tous les acteurs sont dissociés de tous les postes qu'ils occupent dans l'équipe. Puisque les acteurs quittent une ou plusieurs postes, leurs rôles peuvent être modifiés par la "*mise à jour de sortie*". La dissolution d'une équipe peut être demandée par l'équipe elle-même, par l'équipe parente ou par une autre entité de l'environnement virtuel (p. ex. une action).

3 Système de décision et rattachement à l'environnement

Deux points importants doivent être pris en compte : comment spécifier le comportement des équipes et comment les intégrer dans l'environnement virtuel. Ces problèmes sont identiques à ceux rencontrés lors de la mise en place d'un moteur de scénarios ou de n'importe quelle entité réactive intégrée dans un environnement. Ces points sont déjà pris en compte par #SEVEN. Nous avons donc réalisé une bibliothèque d'entité #SEVEN spécifique pour permettre la spécification et le fonctionnement des équipes. Cette bibliothèque propose les classes d'entités suivantes :

- Une classe de ***contexte externe*** qui prend en compte l'équipe et sa structure (postes, membres, sous-équipes) et une référence vers l'environnement virtuel et ses différents composants.
- Une classe de ***senseurs d'équipe*** permettant de vérifier qu'une sous-équipe existe.
- Une classe de ***senseurs de postes***, permettant :
  - de vérifier qu'un acteur peut être assigné à un poste,
  - de savoir si un poste est occupé.
- Une classe de ***senseurs de rôles***, permettant de faire des tests sur le rôle d'un membre de l'équipe.
- Une classe d'***effecteurs de gestion des postes***, permettant d'assigner ou de retirer un membre à un poste.
- Une classe d'***effecteurs de gestion des équipes***, permettant de dissoudre ou de créer des équipes/sous-équipes.
- Une classe d'***effecteurs de rôles***, permettant de modifier le rôle d'un membre de l'équipe.

Ces classes sont complétées par les entités de base de #SEVEN (senseurs et effecteurs de manipulation des types de bases) et de la bibliothèque permettant l'intégration de #SEVEN dans un environnement #FIVE pour percevoir les actions réalisées ou l'état de l'environnement.

La spécification du comportement d'une équipe est donc réalisée graphiquement grâce à l'outil auteur #SEVEN.

L'algorithme 5 donne un exemple de spécification d'équipe de chirurgie. Le comportement de l'équipe principale crée dès le début de la simulation une équipe de chirurgie en assignant les postes aux acteurs comme définis dans la définition d'équipe XML. L'assignation de ces acteurs à différents postes met à jour leur rôle. Le chirurgien est responsable de l'équipe, il peut donner des ordres et réaliser les étapes principales de la procédure. L'interne obtient des droits liés à l'assistance du chirurgien et l'instrumentiste devient responsable de l'organisation et de l'entretien des instruments.

L'équipe principale supervise certaines conditions qui peuvent modifier les rôles de manière indirecte. Par exemple, comme présenté dans la Figure 5.3 si un acteur prend un objet (action prendre) et que cet objet n'est pas aseptisé, alors le rôle de l'acteur est mis à jour pour le marquer comme non aseptisé.

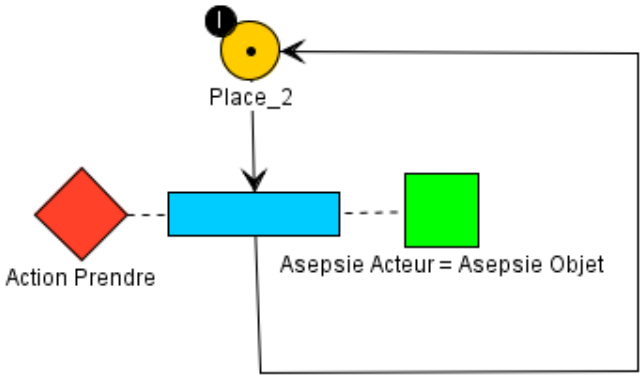


Figure 5.3 – Vérification de l'asepsie lors d'une action prendre.

En tant que personnel médical, les acteurs sont tous soumis à un ensemble de règles communes qui leur donnent le droit de pratiquer. Si une de ces règles est brisée (p. ex. par exemple l'asepsie), l'acteur concerné ne peut plus pratiquer. Comme présenté dans la Figure 5.4 cette règle est gérer de la manière suivante :

- L'équipe principale vérifie la cohérence de chaque poste de la classe "*PersonelBloque*" grâce à la vérification de cohérence qu'elle offre.
- Si le test échoue, l'équipe met à jour le rôle de l'acteur pour lui interdire de pratiquer.

Permettre simplifier la spécification de l'équipe, le réseau de la Figure 5.4 est défini une fois dans une spécification externe puis réutilisé et configuré grâce à la fonctionnalité de #SEVEN lui permettant de charger des sous-scénarios.

Algorithm 5 Exemple de définition d'une équipe avec une sous-équipée.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Equipe principale -->
<team id="EquipeBloque" xmlns="http://www.insa-rennes.fr/team"
  classname="Team, SEVEN.Team" behaviour="Intervention.xml">

  <poste classname="PersonelBloque, chirurgie" id="Chirurgien">
   <member id="Virtuel1" />
  </poste>
  <poste classname="PersonelBloque, chirurgie" id="Interne">
   <member id="Virtuel2" />
  </poste>
  <poste classname="PersonelBloque, chirurgie" id="Infirmier1">
   <member id="Utilisateur1" />
  </poste>

  <!-- Equipe Chirurgicale -->
  <subteam id="Chirurgie"
    classname="Team, SEVEN.Team" behaviour="Chirurgie.xml">
    <position classname="Chirurgien, chirurgie" id="Responsable">
    <member id="Virtuel1" />
    </position>
    <position classname="Interne, chirurgie" id="Assistant">
    <member id="Virtuel2" />
    </position>
    <position classname="Infirmier, chirurgie" id="Instrumentiste">
      <member id="Utilisateur1" />
    </position>
  </subteam>

</team>
```

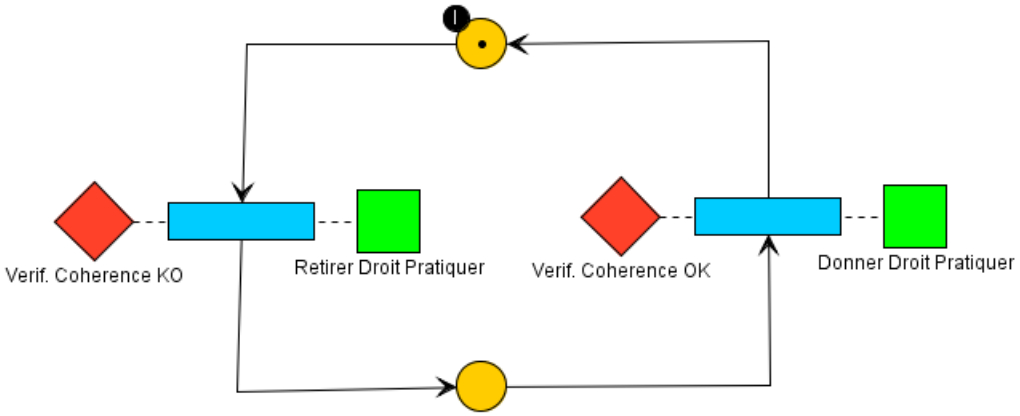


Figure 5.4 – Vérification de la cohérence d'un poste et mise à jour du rôle de l'acteur concerné si nécessaire.

4 Synthèse

Dans ce chapitre, nous proposons un modèle de rôles pour les acteurs en environnement virtuel collaboratif. Grâce à un principe de filtrage des actions, ce modèle permet de prendre en compte différents concepts liés à l'acteur qui vont influer sur son rôle comme ses compétences, ses droits, ses devoirs (par gestion des priorités). Bien que certains modèles existants aient déjà proposé des approches similaires [Cavazza et al., 2007; Szilas, 2003], notre modèle diffère sur le fait que son fonctionnement n'est pas lié à la spécification des actions, mais géré par un système d'attributs portés par les actions et fournis, notamment par notre modèle de scénarios #SEVEN (c.f. Chapitre 3). Ceci permet d'utiliser des composants définissant les actions plus génériques. Dans nos travaux, nous choisissons les informations liées au rôle des acteurs de manière arbitraire. Cependant, il serait intéressant les définir grâce à une formalisation du domaine telle qu'une ontologie. Ceci pour rendre les rôles plus facilement réutilisables, mais également pour proposer une assistance à la spécification. Enfin, l'utilisation des concepts liés à l'acteur n'est pas obligatoire, le système de filtrage permet également une utilisation plus simple en associant les actions directement aux rôles des acteurs comme peuvent également proposer d'autres modèles existants [Gerbaud et al., 2009], [Chevaillier et al., 2012].

Le modèle propose également un système d'évolution des rôles basé sur les structures d'équipes. Une équipe est un groupe organisé d'acteurs. L'implication de chaque acteur dans l'organisation de l'équipe et dans la simulation peut évoluer en fonction des actions ayant lieu dans l'environnement. Notre modèle est utilisable conjointement avec une évolution des rôles des acteurs contrôlée par les actions. Si le moteur de scénarios utilisé peut déclencher des actions (dans notre cas #SEVEN), alors le modèle fournit un équivalent à toutes les approches utilisées

dans la littérature pour représenter l'évolution des rôles (pilotées par actions, par scénarios ou par modèle d'équipe).

Les spécifications, réalisées dans des fichiers externes, sont facilement modifiables pour permettre de modifier le déroulement d'une simulation en proposant

- des rôles, et donc des comportements, différents pour les acteurs,
- des règles différentes d'évolutions des rôles

Ces composants sont également réutilisables dans d'autres simulations. Par exemple, dans nos simulations du projet S3PM, une équipe de chirurgie contient toujours un chirurgien, un interne et une infirmière instrumentiste quelle que soit la procédure réalisée, mais elle pourrait être étendue pour prendre en compte l'ensemble des participants comme l'équipe anesthésiste. Puisque les équipes sont des entités avec un comportement propre, nous proposons d'utiliser le modèle #SEVEN pour décrire et exécuter les comportements des équipes. Il est donc possible d'utiliser ses outils auteurs graphiques pour simplifier le travail de spécification.

Cas d'utilisations

6

Dans ce Chapitre, nous présentons des cas d'utilisations des modèles présentés dans ce manuscrit. L'accent est mis sur la simulation du projet S3PM qui est présentée en détail dans la Section 1. La vidéo https://youtu.be/qImpThoJiK0 présente de manière générale son exécution sur la plateforme immersive du laboratoire. D'autres travaux ayant utilisé nos modèles sont présentés rapidement dans la Section 2.

1 S3PM : simulation de craniotomie

Le projet S3PM se concentre sur la collaboration du personnel médical dans la salle d'opération lors d'une intervention. Contrairement à une majorité des travaux actuels en simulation médicale, nous ne nous intéressons pas au geste technique, mais à l'organisation des acteurs entre eux. La Section 1.1 présente l'intervention simulée. La Section 1.2 décrit la simulation réalisée. La Section 1.3 détaille la spécification des Scénarios possibles. La Section 1.4 détaille la distribution des actions entre les acteurs. Enfin, la Section 1.5 propose une synthèse du travail réalisé.

1.1 Description de la procédure

Nous avons travaillé avec des membres du personnel médical du service de neurochirurgie de CHU Pontchaillou de Rennes pour obtenir une description précise d'une opération de neurochirurgie. Afin de nous focaliser sur les éléments fondamentaux, seule une partie de l'intervention (allant du début de l'opération jusqu'à la fin de l'ouverture du crâne du patient) a été modélisée. Ce temps opératoire, appelé ouverture, est déjà conséquent, car elle contient notamment des étapes liées à l'installation du patient, à la vérification de son identité et de l'opération à réaliser. Nous avons donc restreint nos travaux à une partie de l'ouverture, la craniotomie.

La craniotomie (ou craniectomie) est un acte de neurochirurgie bien connu des neurochirurgiens (c.f. Figures 6.1) qui consiste à découper les os du crâne pour permettre l'accès à une zone du cerveau à soigner. La craniotomie est en général précédée de l'installation du patient, de la mise en place de l'opération et d'une incision pour permettre l'accès au cerveau. Comme toute procédure de neurochirurgie, la craniotomie demande la collaboration de plusieurs personnes (c.f. 6.2) Elle implique principalement trois acteurs : le chirurgien, l'interne et l'instrumentiste. Le chirurgien est responsable des principales étapes techniques et de la coordination de l'équipe chirurgicale. L'interne est en quelque sorte la seconde paire de mains

Figure 6.1 – Craniotomie réalisée le 5 aout 1900 au congrès international de médecine. Origine [Doyen, 1902]

du chirurgien et vient l'assister sur certaines manipulations. L'instrumentiste est une IBODE (Infirmière de Bloc Opératoire Diplômée d'État) responsable du passage des instruments au chirurgien et à l'interne ainsi qu'à l'organisation des instruments sur la table et à leur nettoyage.

Le déroulement général d'une craniotomie est décrit par l'Algorithme 6.

Au cours de la procédure, des saignements peuvent apparaitre. En fonction de leur abondance, ils seront traités de deux manières différentes. Dans le cas d'un saignement léger, le chirurgien applique une cire sur le saignement pour arrêter le flux. Dans le cas d'un saignement abondant, une procédure de cautérisation plus complexe est réalisée. L'Algorithme 7 décrit cette procédure.

En discutant avec le personnel médical, nous avons appris que chaque chirurgien et/ou chaque équipe de chirurgie possède ses habitudes lors des différentes étapes d'une procédure chirurgicale.

Algorithm 6 Déroulement général d'une Craniotomie

1: Instrumentiste donne "Moteur" au Chirurgien
2: Interne et Chirurgiern : Percer crane (x 1 à 4)
3: Chirurgien pose "Moteur" sur le "Dais"
4: Instrumentiste donne "Décolle Dure-mère" au Chirurgien
5: Chirurgien décolle Dure-mère (x 1-4 trous)
6: Chirurgien rend "Décolle Dure-mère" à Instrumentiste
7: Instrumentiste donne "Moteur" au Chirurgien
8: Interne et Chirurgien : Découpe Volet Osseux (x 1-4 trous)
9: Chirurgien pose "Moteur" sur le "Dais"
10: Instrumentiste donne "Décolle Dure-mère" au Chirurgien
11: Chirurgien décolle "Volet Osseux"
12: Chirurgien saisit "Volet Osseux" avec sa main
13: Instrumentiste tend "Cupule" au Chirugien
14: Chirurgien rend "Décolle Dure-mère" à Instrumentiste

Algorithm 7 Déroulement Cautérisation en cas de saignement abondant.

1: Instrumentiste donne « Pince bipolaire » (PB) au Chirurgien
2: Chirurgien fait contact vaisseau sanguin avec PB
3: Chirurgien annonce « Feu »
4: Interne appuie sur Pédale
5: Chirurgien annonce « Stop »
6: Interne relâche Pédale
7: Chirurgien rend PB à Instrumentiste

Figure 6.2 – Opération de neurochirurgie réalisée au CHU de Rennes.

Un des exemples qui nous ont été donnés est que, dans la procédure de cautérisation, l'acteur qui déclenche la mise à feu de la pince bipolaire avec la pédale n'est pas toujours l'interne, mais peut, par exemple, être l'instrumentiste. Pour représenter ces cas, nous proposons donc différentes organisations d'équipes lors de la cautérisation de saignements abondants. Voici des exemples d'organisations possibles :

- **A** - Le chirurgien ne donne que les ordres, l'interne utilise la pince bipolaire et déclenche la mise a feu.
- **B** - Le chirurgien ne donne que les ordres, l'interne utilise la pince bipolaire et l'instrumentiste déclenche la mise à feu.
- **C** - Le chirurgien utilise la pince bipolaire et déclenche la mise à feu.
- **D** - Le chirurgien donne les ordres et utilise la pince bipolaire, l'interne déclenche la mise a feu.
- **E** - Le chirurgien donne les ordres et utilise la pince bipolaire, l'instrumentiste déclenche la mise a feu.
- **F** - Si le chirurgien utilise le trépan, alors A, sinon D.

Un autre exemple d'habitude, liée à l'expérience du chirurgien, est le nombre de trous percés pour préparer la découpe du volet osseux (c.f. ligne 2, Algorithme 6). Un chirurgien débutant va plutôt préférer réaliser quatre trous, mais un expert peut vouloir en réaliser moins pour des raisons d'habitudes ou de préférences personnelles. L'équipe est en règle générale au courant de ces habitudes et sait les prendre en compte dans son organisation.

1.2 Présentation de la simulation

Nous avons réalisé un prototype d'application, proposant un environnement virtuel collaboratif pour la formation des infirmières instrumentistes à la craniotomie. Nous aurions pu nous intéresser à n'importe quel membre de l'équipe, mais nous avons choisi de travailler sur la formation des instrumentistes, car nous avons un accès facilité à de nombreuses infirmières grâce à l'école et au service de neurochirurgie de Rennes. Nous avons choisi la craniotomie, car il s'agit d'une opération réalisée régulièrement, ce qui facilite les acquisitions de données pour la génération des modèles de procédure. Notre approche est de mettre en situation l'apprenant en tant qu'instrumentiste lors de la procédure. Pour ce faire, il collabore avec deux acteurs virtuels : l'interne et le chirurgien. Ces deux acteurs suivent les actions à réaliser fournies par le moteur de scénarios, obtenus en utilisant les paramètres des senseurs pour filtrer l'ensemble des réalisations possibles fournies par le moteur de relations #FIVE. Une seconde passe de filtrage est réalisée grâce à notre modèle de rôle. Ils reçoivent ainsi chacun un sous-ensemble des actions à réaliser. Le comportement des acteurs virtuels est assez simple. Nous avons défini trois groupes d'actions : les actions de base (prendre ou poser un objet, tendre la main, annoncer un mot-clé), les actions de procédure et les actions de la cautérisation. Ceci est réalisé grâce à une gestion des niveaux de priorité des actions décrits grâce à des attributs #SEVEN. Les actions (réalisations #FIVE) déclenchent les animations des personnages virtuels.

Le formé a, de son côté, la tâche de maintenir la table d'opération organisée et de fournir les bons instruments au bon acteur et au bon moment. Lorsqu'un des acteurs virtuels a besoin d'un instrument, il tend la main et attend que l'utilisateur le lui fournisse. Si l'instrument est le bon, l'acteur virtuel exécute alors son action. Si l'instrument n'est pas le bon ou si l'acteur n'en a plus besoin, il essaye de le reposer sur un emplacement prévu sur la table. Cette action n'est réalisable que si l'emplacement prévu est libre. Si l'emplacement n'est pas libre alors l'acteur attend qu'il soit libéré par l'utilisateur. La table dispose d'un certain nombre d'instruments et d'emplacements prévus pour les déposer.

Lors de la simulation, un saignement peut avoir lieu de manière aléatoire. Dans ce cas, la procédure de cautérisation de saignement abondant se déclenche. Les acteurs virtuels considèrent cette tâche prioritaire par rapport à la procédure principale. Si une action de la cautérisation doit être réalisée, alors ils vont chercher à le faire, même si la procédure principale est en cours, car le niveau de priorité est plus élevé. La simulation peut prendre en compte les différentes organisations d'équipes telles que proposées dans la Section 1.1. La gestion du nombre de trous à réaliser au minimum par le chirurgien en fonction de son expérience est également prise en compte (c.f. Section 1.1 et Algorithme 6, ligne 2). Enfin, lors de la simulation, l'acteur

chirurgien peut ne pas donner les ordres "*feu*" et "*stop*". Ceci est réalisé en les retirant de la spécification des scénarios. L'acteur virtuel ne les prend donc plus en compte dans ses prises de décision. Dans ce cas, l'utilisateur peut prendre la décision de déclencher quand même la mise à feu en outrepassant ses droits.

Pour des questions de temps (durée de la simulation, mais surtout de réalisation du système), nous simplifions la procédure réalisée. La procédure commence par une simple incision et termine par le stockage du volet osseux dans une cupule. L'interne assiste le chirurgien lors du perçage et de la découpe en lubrifiant l'os. L'utilisateur à accès à tout moment aux actions prendre, poser/donner n'importe quel objet et déclencher/arrêter la pince bipolaire. Ceci permet à l'utilisateur de voir ses erreurs et d'en tirer leçons. Par exemple, si l'utilisateur ne fournit pas le bon instrument, il est immédiatement reposé sur la table par le chirurgien ou l'interne. Dans le cas contraire, l'acteur virtuel l'utilise dans la réalisation d'une étape de la procédure.

Nous avons identifié plusieurs difficultés techniques liées à la spécification de cet environnement virtuel.

- La première est liée à la synchronisation des actions des différents acteurs. L'exemple type est la lubrification par l'interne de la zone de travail du chirurgien. Tant que l'interne arrose le crane, le chirurgien peut travailler. L'interne doit par contre s'interrompre lorsque suffisamment de trous ont étés réalisés, pour laisser le chirurgien utiliser le décolle dure-mère, puis reprendre pour permettre la découpe du volet osseux.

- La spécification des scénarios doit fournir une certaine souplesse : le nombre de trous réalisés par le chirurgien est inconnu (mais vaut au moins 1), tout en étant dirigiste pour ne pas laisser les acteurs sortir de la procédure. Il s'agit également d'une difficulté liée à la spécification des scénarios, mais qui est complétée par la configuration du comportement de l'acteur chirurgien.

- La gestion des droits, et donc l'évolution des rôles des acteurs, dépend des situations. Les ordres du chirurgien modifient indirectement les rôles des acteurs en leur offrant ou retirant des droits d'action. Dans certains cas, le chirurgien peut ne pas donner d'ordre :

  - soit c'est un cas prévu et "l'équipe à l'habitude" les rôles doivent donc être modifié automatiquement par l'équipe,
  - soit c'est un "oubli" de la part du chirurgien (simulé en retirant les actions associées de la spécification) et l'utilisateur peut alors outrepasser ses droits pour permettre de continuer la procédure.

- L'organisation de l'équipe pour la cautérisation n'est pas connue à l'avance lors de la spécification des scénarios, elle dépend des habitudes que le veut simuler. Par contre la procédure principale et la procédure de cautérisation sont connues. La spécification des scénarios doit donc être indépendante des règles de l'équipe. De plus, ces dernières peuvent être complexes (par exemple cas F, Section 1.1).

- Enfin, le niveau de guidage de tous les acteurs n'est pas identique. Les acteurs virtuels sont fortement guidés par le moteur de scénarios alors que l'utilisateur est beaucoup plus libre dans ses actions.

Grâce à #FIVE [Bouville et al., 2015] (c.f. Annexe B et Chapitre 4), nous avons réalisé un environnement virtuel réactif offrant de grandes possibilités d'interactions. Il contient.

- les instruments nécessaires à la procédure de craniotomie,
- de nombreux autres objets interactifs, comme des meubles sur roulette, des tiroirs contenant d'autres instruments et des lampes articulées.

Les actions réalisables par les acteurs sont celles liées à la procédure, mais aussi quelques actions, plus génériques et réalisables à tout moment par l'utilisateur, permettant par exemple d'ouvrir ou fermer un tiroir, de déplacer de l'équipement sur roulettes, d'ajouter et supprimer des instruments sur la table d'opération ou d'orienter les lampes.

Nous avons réalisé la scène à partir de photos et des plans d'un bloc opératoire existant du département de neurochirurgie du CHU Pontchaillou de Rennes et de modèles 3D d'équipements existant dans la salle réelle. La Figure 6.3 présente cette scène projetée dans le dispositif immersif "*Immersia*" du laboratoire et la Figure 6.4 présente, à la troisième personne, le point de vue d'un utilisateur.

Figure 6.3 – Bloc opératoire virtuel du projet S3PM.

1.3 Spécification de l'ensemble des scénarios

Dans cette section nous décrivons comment a été réalisée la spécification de l'ensemble des scénarios possibles à partir de la description de la procédure. Cette étape ne s'intéresse qu'à

Figure 6.4 – Point de vue de l'utilisateur dans l'environnement virtuel collaboratif pour la Formation des Infirmières Instrumentistes.

l'organisation logique et temporelle de la procédure, indépendamment du rôle des acteurs devant réaliser les actions. La distribution des actions est réalisée à l'étape suivante.

L'objectif de cette simulation est de former un utilisateur au poste d'instrumentiste à une procédure bien définie. Pour cela les acteurs virtuels ne doivent pas sortir de la procédure et la suivre scrupuleusement. Les acteurs virtuels suivent donc les actions fournies par le moteur de scénarios, qui leur fournit uniquement les actions réalisables en fonction de la spécification des scénarios.

Les actions de l'utilisateur ne sont contraintes que par l'environnement virtuel grâce aux relations et réalisations #FIVE. Les actions sont directement fournies par le moteur de relations. L'utilisateur peut déclencher toutes les réalisations qui l'impliquent. Comme l'IBODE ne doit pas réaliser de gestes chirurgicaux, nous n'avons simplement pas rendu possible leur déclenchement par l'interface utilisateur. Par contre, il peut agir sur les objets de la scène (instruments, mobilier) et passer les instruments de sa main à la main d'un autre acteur.

Les actions de base sont des actions qui sont utilisables à tout moment de la simulation par les acteurs. Il s'agit des actions liées au passage d'instrument (attendre un objet, prendre et poser) et des ordres de base du chirurgien (annoncer *feu* ou *stop*). Comme les acteurs virtuels ne peuvent réaliser des actions que si elles sont proposées par le scénario, nous avons rendu ces actions réalisables sans contraintes autres que l'état de l'environnement (grâce à #FIVE). Dans leur processus de décision, les acteurs virtuels choisissent parmi ces actions s'ils sont incapables

de réaliser une action de la procédure. Par exemple, si le chirurgien ne peut pas réaliser l'étape suivante de la procédure, il va en premier lieu poser l'instrument qu'il tient, puis tendra la main pour obtenir l'instrument suivant de la part de l'utilisateur. Pour simplifier la spécification, un réseau #SEVEN générique est défini dans un fichier externe et est paramétré plusieurs fois différemment pour décrire les différentes actions de base. Ce réseau est présenté dans la Figure 6.5. Les paramètres (Relation et objets #FIVE) sont définis dans un jeton placé directement sur la place sous-réseau.

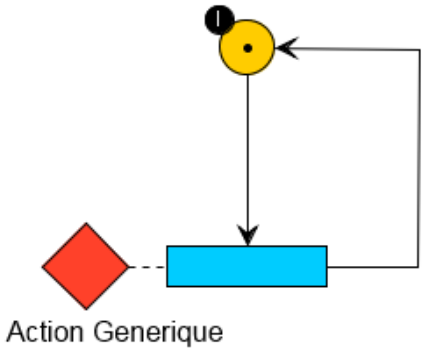


Figure 6.5 – Du point de vue du réseau, l'action est exécutable sans contraintes temporelles ou causales.

La procédure principale est décrite par le réseau #SEVEN présenté par la Figure 6.6. Cette dernière impose de réaliser au moins un trou. De plus, les actions liées au perçage et à la découpe ne sont réalisables que si l'os est lubrifié (place "*Skull Watered*"). Le fonctionnement est inverse pour l'action de décoller la dure-mère : elle n'est réalisable que si la lubrification n'est pas en cours. Une fois la découpe réalisée, la fin est une séquence d'actions basiques. Le perçage doit être réalisé au moins une fois (Transition 17). Les perçages supplémentaires sont associés à la Transition 51 qui peut être déclenchée indéfiniment jusqu'au décollage de la dure-mère. Maintenir le décompte des perçages via l'action percer n'est pas une bonne solution, car cela spécialise trop la relation et la rend difficilement réutilisable pour d'autres simulations. Nous avons intégré le décompte de perçage dans le comportement de l'équipe. Lorsque le nombre prévu est atteint, le rôle du chirurgien est modifié pour lui autoriser la suite de la procédure et lui interdire la réalisation d'un trou supplémentaire. Il nous aurait également été possible de laisser l'acteur virtuel compter le nombre de trous à réaliser et choisir, quand passer à l'étape suivante.

La gestion de la cautérisation, représentée par la Figure 6.7, est réalisée lorsqu'un saignement intervient. Nous avons décidé d'intégrer le déclenchement du saignement dans la spécification des scénarios. La Transition 28 est déclenchée de manière aléatoire (durée entre deux déclenchements, définie entre deux bornes de temps minimum et maximum). Le scénario agit sur le patient et déclenche un saignement. Ce saignement est également contrôlé par le moteur de scénarios lorsque la procédure de cautérisation est complétée. Il aurait été possible d'utiliser

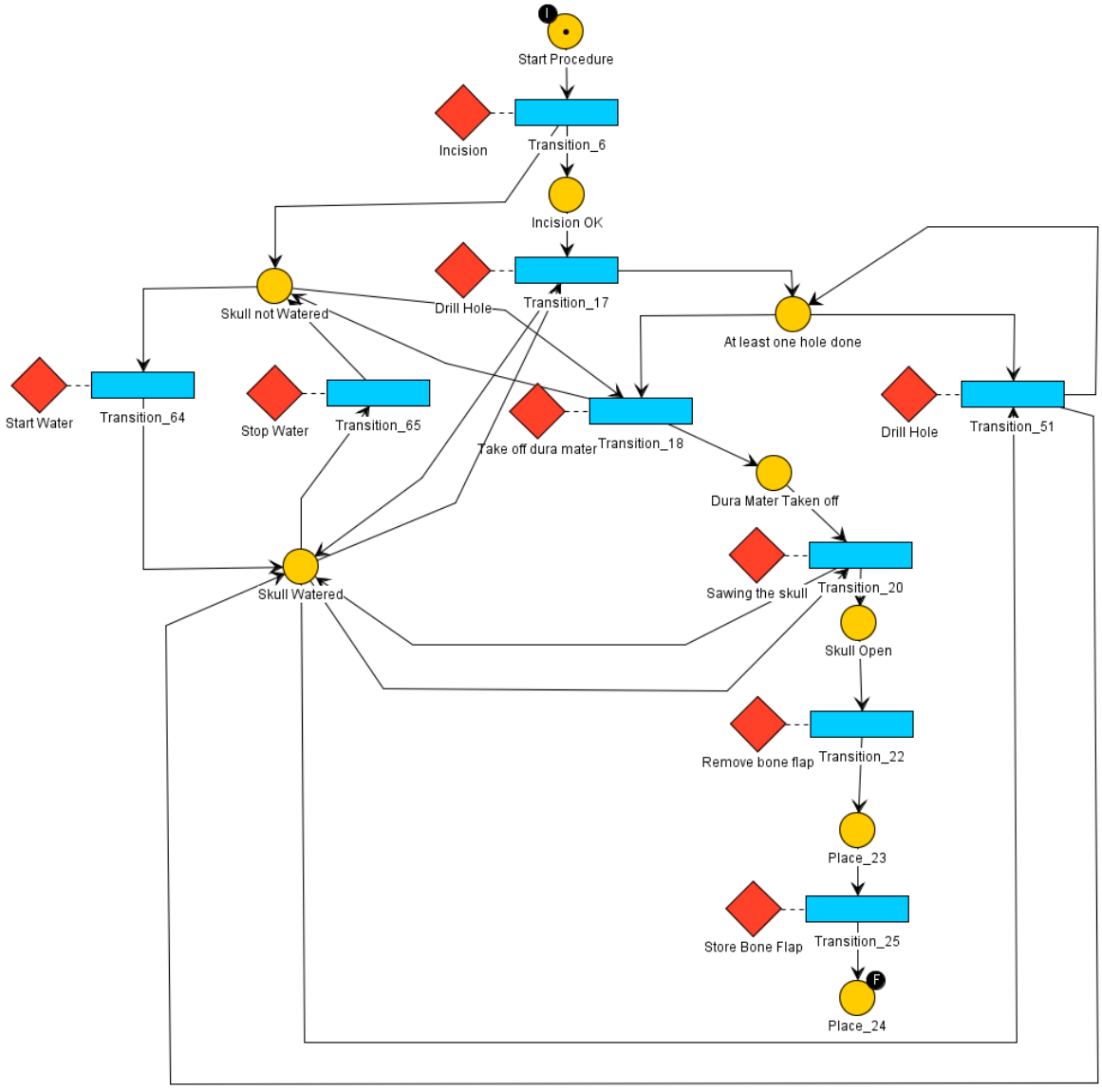


Figure 6.6 – Procédure principale de la simulation S3PM.

un comportement associé à l'objet patient qui déclencherait ce saignement. L'avantage de notre approche est qu'il est possible de modifier la fréquence ou de simplifier la procédure en retirant les saignements sans avoir à impacter l'Environnement virtuel. Cela offre également la possibilité d'ajouter la gestion des saignements légers en complétant le réseau.

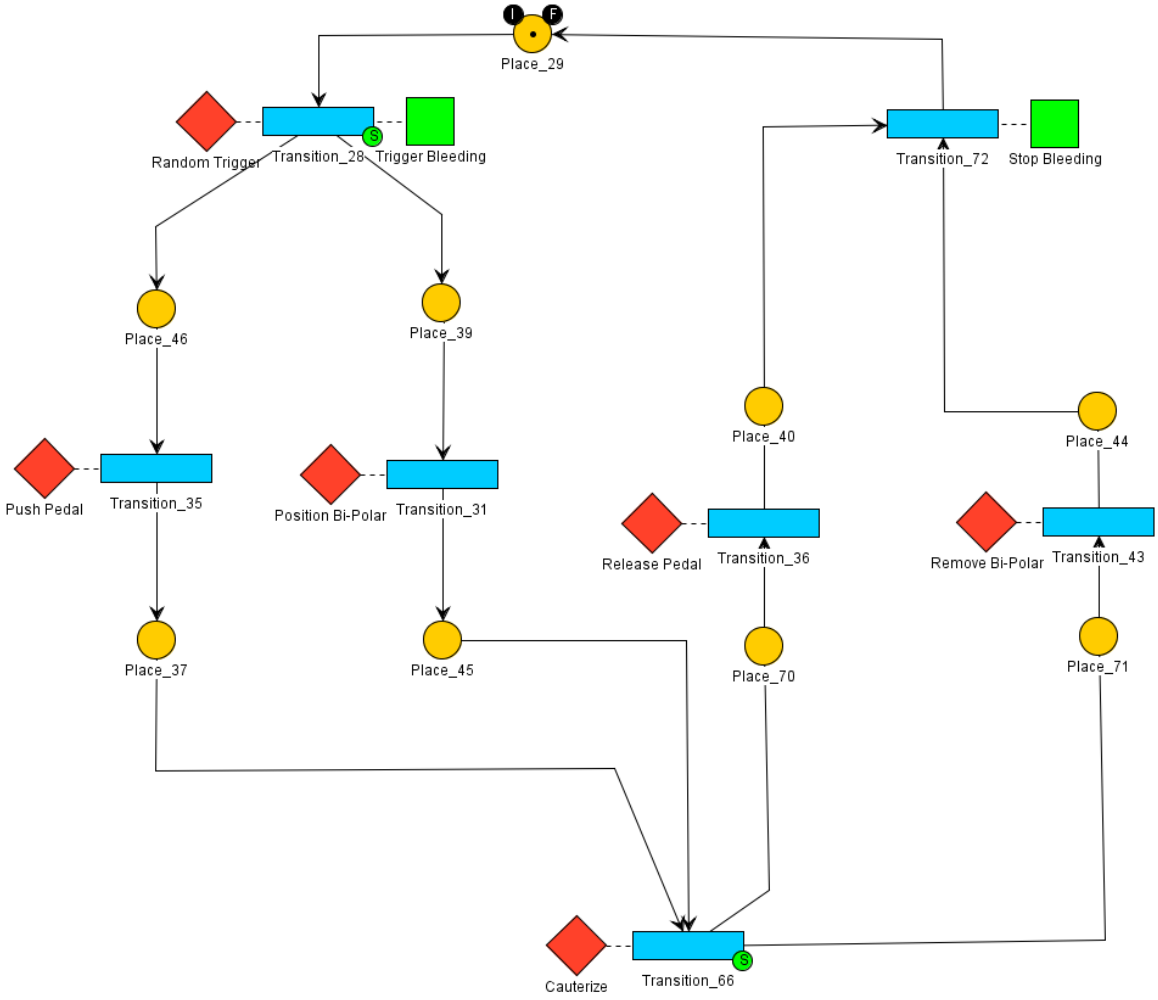


Figure 6.7 – Procédure de cautérisation lors de saignements importants.

1.4 Distribution des actions entre les acteurs

La spécification des rôles initiaux de chaque acteur indique son niveau de compétence en chirurgie. Cette compétence nous est utile pour filtrer les actions réalisables par l'interne et par le chirurgien. Les acteurs sont associés à une position dans l'équipe : Chirurgien, Interne et Instrumentiste ce qui leur donne certains droits au début de la simulation. Par exemple, l'interne est autorisé à lubrifier la zone de travail du chirurgien dès que l'action devient disponible d'après moteur de scénarios, seule l'instrumentiste a le droit de réaliser l'action "*prendre un*

instrument" et le chirurgien a le droit de percer. Les ressources nécessaires aux actions des deux procédures (principale et cautérisation) sont également indiquées par des attributs portés par les actions pour permettre aux acteurs de savoir pourquoi une action n'est pas réalisable. Par exemple, la relation #FIVE inciser nécessite un instrument tranchant, mais l'attribut indique précisément qu'il faut un scalpel pour l'effectuer. Le chirurgien attendra donc un scalpel de la part de l'instrumentiste. Ainsi, l'action inciser peut en réalité être réalisée avec n'importe quel instrument tranchant, mais la spécification de la procédure conseille l'utilisation d'un scalpel. Ces conseils sont suivis à la lettre par les acteurs virtuels.

Une partie du comportement de l'équipe s'occupe de la synchronisation des actions de l'interne et du chirurgien. Il s'agit de savoir quand l'interne a le droit de commencer au d'arrêter de lubrifier la zone de travail et quand le chirurgien a le droit de réaliser certains gestes techniques comme le perçage, la découpe ou décoller la dure-mère. Pour ce faire, le comportement de l'équipe pour ce point est décrit par le réseau de la Figure 6.9. L'équipe donne et retire des droits aux acteurs en fonction de l'avancement de la procédure. Par exemple, l'équipe se repose sur le nombre de trous que le chirurgien à l'habitude de réaliser (défini dans son rôle). Lorsque ce nombre de trous est atteint (perception de l'environnement par l'équipe), le chirurgien perd le droit d'utiliser l'action "*percer*" et gagne le droit d'utiliser l'action décoller "*dure-mère*". Dans le même temps, l'interne perd le droit de lubrifier la zone de travail. Ainsi, la simulation s'adapte facilement à différentes habitudes de chirurgiens sur ce point, juste en adaptant le nombre de trous que ces derniers préfèrent réaliser. Ici encore, les acteurs virtuels considèrent l'absence de droit comme une interdiction totale de réaliser l'action. Ils cherchent à réaliser la prochaine action qu'ils sont autorisés à faire. L'intérêt principal d'utiliser une représentation des droits est de pouvoir remplacer facilement l'acteur virtuel par un autre acteur, par exemple un utilisateur, qui aura une interprétation différente des droits et des possibilités d'action.

Une autre partie de l'équipe gère la procédure de cautérisation. La Figure 6.8 représente le comportement d'une équipe qui répond à la règle **F** proposée dans la section 1.1. En fonction de l'instrument utilisé par le chirurgien, l'équipe principale crée une sous-équipe qui gère l'organisation de la procédure de cautérisation. La sous-équipe cautérisation possède trois positions : chef, exécutant et assistant. Le chef donne les ordres, l'exécutant utilise la pince bipolaire (le geste technique) et l'assistant déclenche la mise à feu. L'équipe principale va attribuer ces rôles de manières différentes en fonction de l'instrument utilisé par le chirurgien au moment du saignement (transition 6 ou 7 de la Figure 6.8). Nous utilisons une sous-équipe pour modifier les droits des acteurs plutôt que directement des actions. Ainsi, nous pouvons réutiliser l'action "*Annoncer*" avec différents mots-clés (feu, stop, etc.) et dans différents contextes. Ceci est important, car la sémantique associée à l'annonce d'un mot-clé est relative au contexte. Par exemple, annoncer stop peut demander l'arrêt d'un geste technique de l'interne contrôlé par le chirurgien (par exemple une incision) ou demander l'arrêt d'un instrument électrique par l'instrumentiste plus loin dans la même procédure. Nous pouvons donc réutiliser la même action "*annoncer*" à différents moments d'une simulation ou dans d'autres simulations en adaptant le comportement de l'équipe.

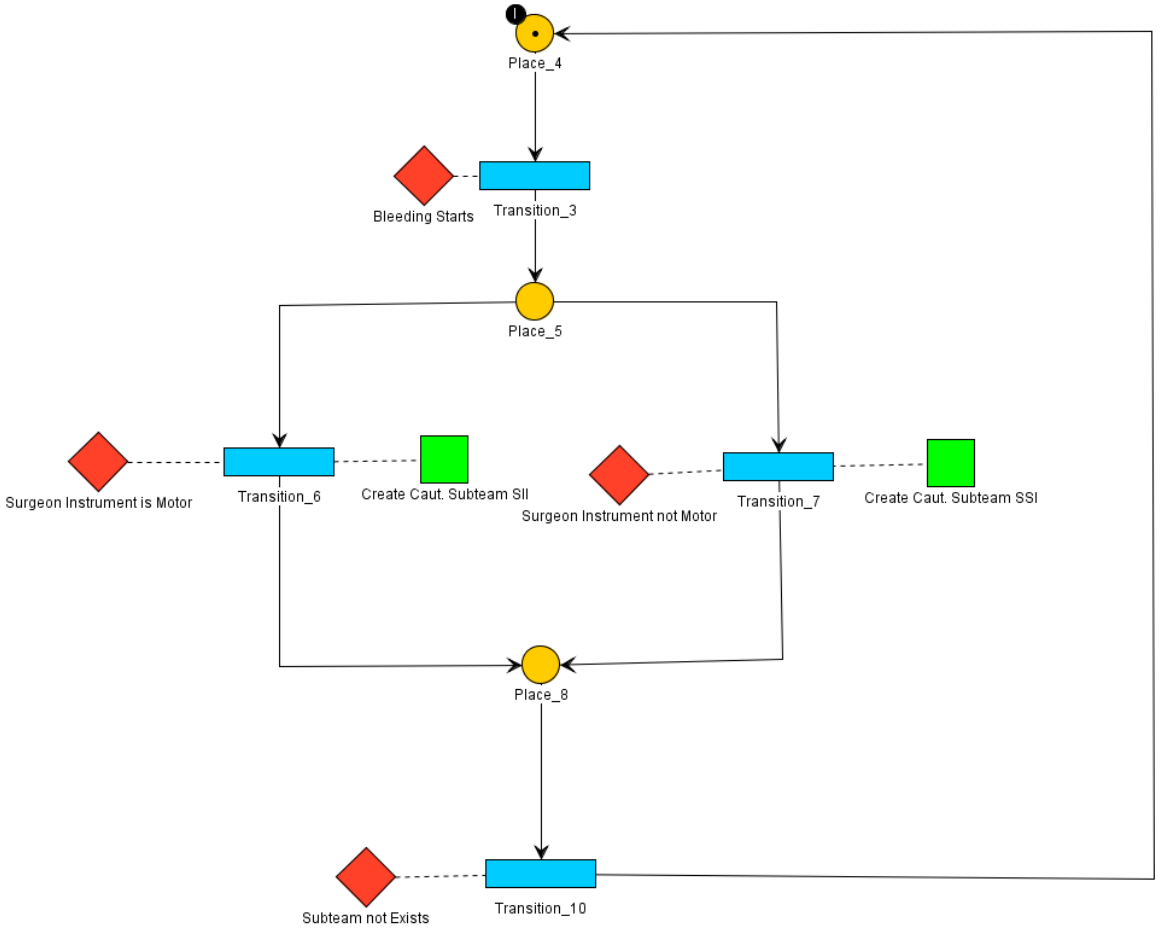


Figure 6.8 – Création de la sous-équipe cautérisation en fonction de l'instrument utilisé par le chirurgien.

1.5 Synthèse

Nos modèles de scénarios, de rôles et d'environnement permettent une grande flexibilité dans la réalisation de l'Environnement virtuel collaboratif pour la Formation du projet S3PM. Il est possible de répondre à des problèmes complexes tels que la représentation des habitudes des équipes de chirurgie ou les niveaux de contrôle asymétriques des acteurs. Nous n'avons pas rencontré de difficulté majeure dans la réalisation de cette simulation. Tous les poings difficiles ont étés levés facilement. Nous avons été capables d'exécuter cette simulation sur différents équipements tels que de simples écrans (c.f. Figure 6.10) , des casques immersifs ou des salles

d'immersion (c.f. Figures 6.3 et 6.4) dans le cadre de nos tests ou de manifestations publiques.

L'utilisation de ces modèles permet également de réduire fortement la spécialisation des composants tels que les actions à une simulation en particulier. Les actions, objets, sous-scénarios, équipes et définitions des acteurs deviennent alors des composants facilement réutilisables d'une simulation à l'autre.

2 Autres Travaux et Conclusion

Lors de nos travaux nous avons également utilisé nos modèles pour réaliser d'autres Environnements Virtuels Collaboratifs. Nous avons mis en place.

- une simulation de maintenance de machine industrielle collaborative entre deux salles d'immersion et donc deux acteurs réels (c.f. Figure 6.11).
- une simulation d'atelier avec trois acteurs réels.
- un système de répétition pour le monde du cinéma entre un acteur réel et des acteurs virtuels [Bouville et al., 2016] (c.f. Figure 6.12).
- Un appartement virtuel équipé en domotique 6.13 pour la réhabilitation de personnes handicapées. Cette simulation propose plusieurs scénarios comme répondre au téléphone ou accueillir un invité. Ces travaux ont étés réalisés avec le centre de rééducation et de réadaptation de Kerpape (Ploemeur, 56).

D'après notre expérience en développement d'environnements virtuels, nous proposons d'intégrer le processus de développement suivant :

1. Obtenir une description du déroulement de la simulation avec les experts du domaine :
   - Objectifs de l'utilisation de l'environnement virtuel
   - Scénarios possibles, nécessaires, interdits...
2. Spécifier les Objets et des Relations (c.f. Chapitres 2, 4 et Annexe B)
3. Spécifier les scénarios possibles grâce au point 1, sans prendre en compte la distribution des actions aux acteurs.
4. Définition des acteurs : quels acteurs réels, quels acteurs virtuels
5. Choix d'un niveau de guidage des acteurs (c.f. Chapitre 4)
6. Spécifier la distribution des actions :
   (a) Définir les conditions d'accès aux actions en accord avec le point 1 : capacités, droits, devoirs, priorités, ressources nécessaires...

(b) Spécifier les rôles initiaux des acteurs.

(c) Spécifier l'organisation de l'équipe et des sous-équipes ainsi que les règles d'évolution des rôles.

Lors de la mise en place de nos différents environnements virtuels, nous n'avons pas rencontré de problèmes de conception qui soient liés à l'utilisation de nos modèles. Nous avons pu mettre en place les différentes simulations facilement. La démonstration S3PM propose plusieurs versions :

- une version prenant en compte les habitudes de l'équipe et la gestion de la cautérisation,
- une version sans cautérisation ne tenant compte que de la procédure principale,
- une version simplifiée de la procédure qui tourne en boucle pour permettre une utilisation lors de démonstrations publiques ou lors de conférences.

La réalisation de ces différentes versions n'a demandé qu'une édition rapide de la spécification des scénarios grâce à l'outil auteur #SEVEN.

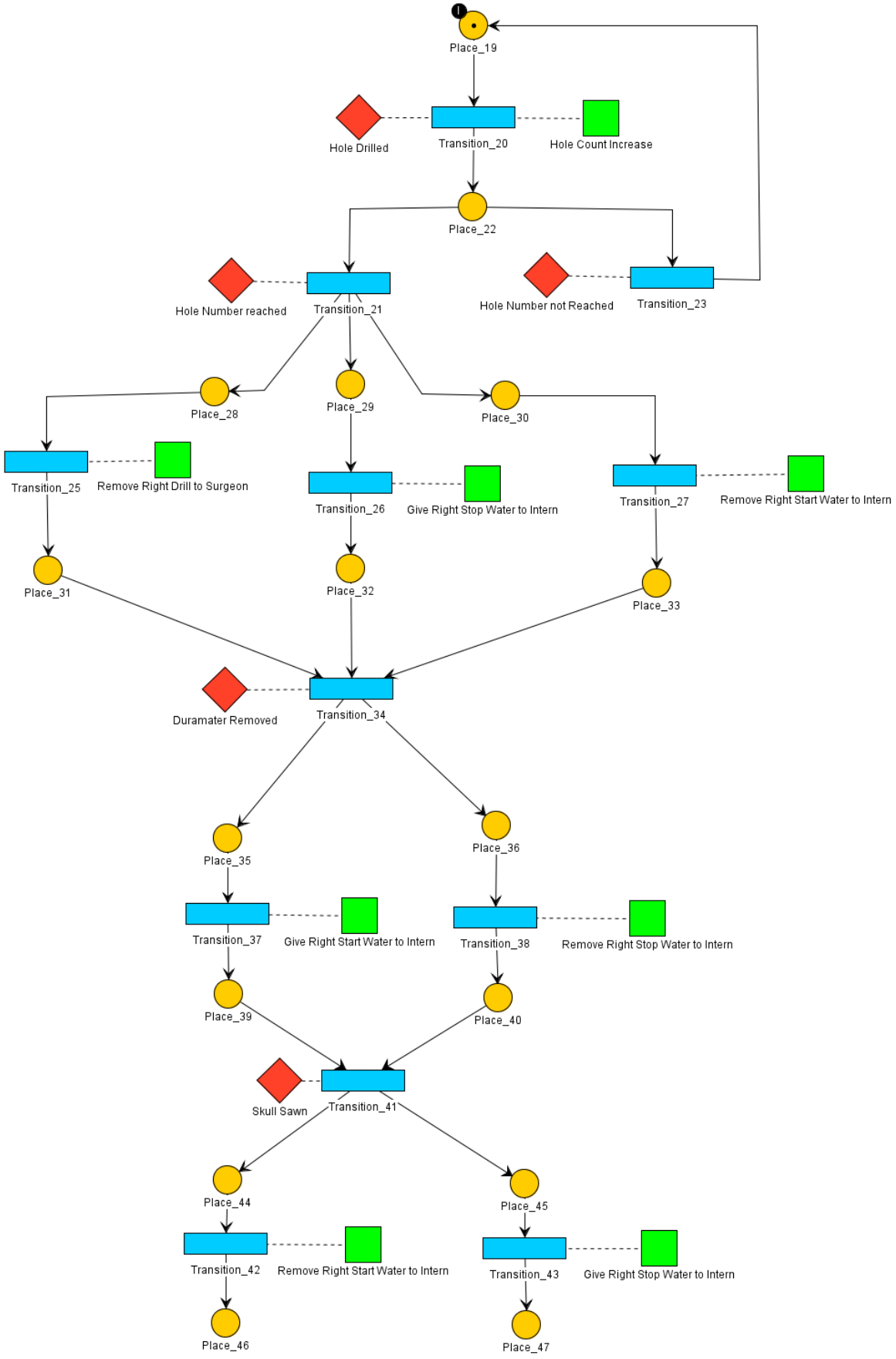


Figure 6.9 – Gestion des droits de l'interne et du chirurgien en fonction du déroulement de la procédure principale.

Figure 6.10 – Utilisation de la simulation S3PM sur un grand écran.

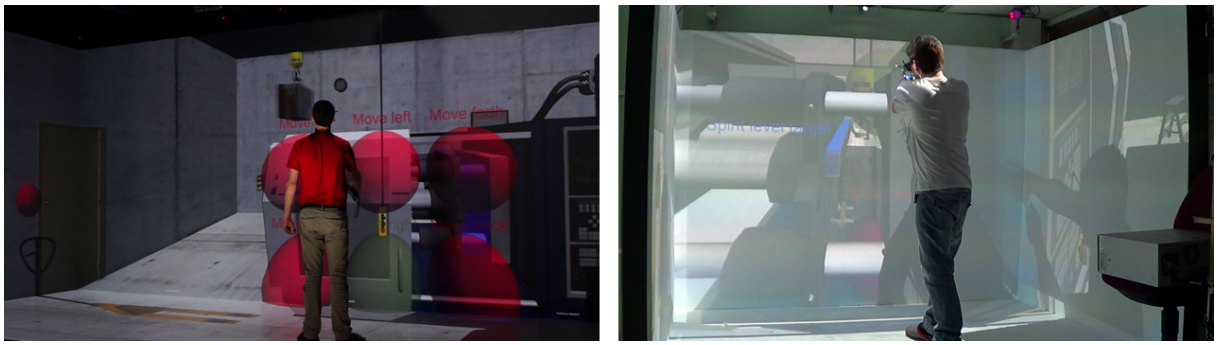


Figure 6.11 – Environnement Virtuel Collaboratif de Formation à la maintenance de machine industrielle avec deux acteurs réels dans deux salles d'immersion reliées entre elles.

Figure 6.12 – Scène utilisée pour la répétition immersive des acteurs de cinéma

Figure 6.13 – Un appartement virtuel pour la réhabilitation de personnes handicapées.

Conclusion

Dans ce manuscrit, nous avons présenté #SEVEN, un modèle de scénarios ne présupposant pas de la nature de l'environnement dans lequel il est utilisé. Pour permettre son intégration dans un environnement, un moteur #SEVEN utilise différentes classes d'entités fournies au travers de bibliothèques. Ces classes permettent de définir des interactions d'un haut niveau d'abstraction entre le moteur et son environnement. Le moteur peut ainsi percevoir l'état de l'environnement, adapter sa représentation interne des scénarios possibles et déclencher des actions dans l'environnement. La représentation interne d'un moteur #SEVEN peut être de différentes natures. Nous avons proposé une implémentation des réseaux de Petri saufs hiérarchiques qui permet la description d'agencements complexes d'actions tout en offrant une représentation graphique utilisable au travers d'un outil auteur et différentes propriétés permettant de simplifier le travail du développeur lors de la spécification. Outre la spécification de l'ensemble des scénarios, notre outil auteur permet également la supervision et le contrôle à de moteurs #SEVEN à l'exécution.

Nous proposons une utilisation de #SEVEN en tant que modèle de scénarios pour environnements virtuels. Nous intégrons #SEVEN dans un environnement défini grâce à #FIVE, un modèle "*objets-relations*". L'utilisation de #SEVEN au-dessus de #FIVE permet de prendre en compte différents aspects des environnements virtuels, notamment :

- Les scénarios spécifiés ne sont pas impactés par le fait que les acteurs soient réels ou virtuels ni par leur nombre.
- Les scénarios peuvent offrir différents niveaux de guidage des acteurs allant de la liberté totale sans guidage à la prise de contrôle des acteurs par le moteur de scénarios.
- Les ressources sont prises en compte pour la synchronisation de l'état du moteur de scénarios et de l'environnement virtuel

Nous avons ensuite proposé un modèle pour la représentation des rôles des acteurs basés sur les recherches en théorie des rôles. Grâce à un système de filtre, ce dernier permet de distribuer les actions des acteurs en fonction des informations qui les définissent comme leurs capacités, leurs droits ou même des informations liées à leur historique, comme l'école dans laquelle ils ont étés formés. Ce système n'est pas intégré directement dans la spécification des actions afin de les laisser génériques pour une réutilisation dans d'autres environnements. Nous proposons également un modèle de gestion de l'évolution des rôles des acteurs au cours de la simulation. Ce

modèle se base sur l'organisation des équipes d'acteurs en permettant de représenter différentes règles et habitudes. Ainsi, un acteur peut voir son rôle modifié par des changements dans son environnement social ou physique.

Puisqu'il ne fait pas d'hypothèse sur la nature de l'environnement dans lequel il est intégré, #SEVEN peut également être utilisé pour décrire les comportements possibles d'une entité dans un environnement virtuel. Dans notre cas, nous avons utilisé cette propriété pour décrire le comportement des équipes d'acteurs dans notre modèle de rôle.

Ces travaux ont étés réalisés dans le contexte du projet S3PM (Synthesis and Simulations of Surgical Process Models). Son objectif est la génération de scénarios pour environnements virtuels pour la formation du personnel médical à partir de l'observation de cas réels. Pour cela, il propose une génération d'un modèle de procédure chirurgicale à base de réseaux "*Test and Flip*". #SEVEN permet l'intégration de ces modèles de deux manières différentes. La première consiste à remplacer le modèle de réseaux de Petri proposé par une adaptation du modèle "*Test and Flip*". La seconde consiste à générer un réseau de Petri au format #SEVEN grâce aux outils du projet ou à traduire un "*Test and Flip*". Actuellement, seule la seconde solution est implémentée. Nous avons également utilisé nos solutions pour la spécification de scénarios par un développeur pour le prototype actuel de simulation du projet.

Perspectives

#SEVEN peut être intégré dans d'autres environnements. Il serait intéressant de le confronter aux travaux réalisés dans différents domaines comme la domotique ou la robotique. Dans le cadre des environnements virtuels de formation, une fonctionnalité intéressante serait d'intégrer la possibilité de réaliser des retours en arrière pour annuler certaines actions des acteurs et permettre aux formés d'essayer d'autres agencements d'actions.

Actuellement, un moteur de scénarios #SEVEN ne peut pas suivre l'exécution d'une procédure spécifique tout en laissant les acteurs libres de leurs actions. Nous travaillons actuellement à l'utilisation de deux moteurs #SEVEN en parallèle : un suivant la procédure et un supervisant l'état de l'environnement. Lorsque le moteur suivant la procédure devient incohérent avec l'état du monde il pourrait être possible de le resynchroniser avec l'environnement plus tard dans la simulation grâce aux marquages du réseau du moteur qui supervise l'état de l'environnement.

Le modèle de rôles que nous proposons repose sur le fait que tous les acteurs possèdent la même représentation des règles de l'équipe et des rôles des acteurs. En réalité, de nombreuses erreurs peuvent venir du fait que les acteurs ne possèdent pas tous une même représentation. Il peut être intéressant d'essayer de modéliser ces situations, par exemple pour détecter certaines erreurs en simulations avant qu'elles n'arrivent sur le terrain.

Dans ce manuscrit, nous avons fait référence à un autre composant des environnements virtuels de formation : le moteur de pédagogie. Ce dernier a également son rôle à jouer dans le déroulement de la simulation. Il peut, par exemple, détecter et prendre en compte les erreurs des utilisateurs. En collaboration avec le moteur de scénarios, il peut également déclencher des actions à la place d'un acteur pour l'aider à avancer ou encore autoriser ou interdire certaines actions en cours de simulation, pour adapter le déroulement de la simulation aux objectifs pédagogiques. Il peut également fournir de l'aide à l'acteur à partir des informations issues du moteur de scénarios, par exemple en mettant en évidence certains éléments. Enfin, il peut tenir compte des erreurs et fournir un retour à l'utilisateur sur la qualité de sa prestation. Toutes ces propriétés sont issues de la possibilité d'un moteur pédagogique à interagir avec l'environnement virtuel, les acteurs et le moteur de scénarios. Ceci semble tout à fait réalisable en utilisant un moteur #SEVEN pour définir des scénarios pédagogiques.

La représentation des réseaux #SEVEN ne semble pas très adaptée aux non-informaticiens. Un outil auteur spécialisé plus accessible serait un plus pour l'utilisation de nos outils. Par exemple, parmi les outils proposés par la suite logiciels SurgeTrack se trouve le logiciel SurgePlan, qui permet de décrire le déroulement d'une opération chirurgicale. Le formalisme utilisé a été

développé en collaboration avec des chirurgiens. Il serait intéressant de l'utiliser directement pour définir des scénarios manuellement. Peut-être peut-il s'adapter à d'autres domaines, comme l'industrie.

Publications de l'auteur

Claude, G., Gouranton, V., and Arnaldi, B. (2015a). Roles in Collaborative Virtual Environments for Training. In *International Conference on Artificial Reality and Telexistence Eurographics Symposium on Virtual Environments* .

Claude, G., Gouranton, V., and Arnaldi, B. (2015b). Versatile Scenario Guidance for Collaborative Virtual Environments. In *International Conference on Computer Graphics Theory and Applications*.

Claude, G., Gouranton, V., Bouville Berthelot, R., and Arnaldi, B. (2014). Short Paper: #SEVEN, a Sensor Effector Based Scenarios Model for Driving Collaborative Virtual Environment. In *International Conference on Artificial Reality and Telexistence, Eurographics Symposium on Virtual Environments*.

Claude, G., Gouranton, V., Caillaud, B., Gibaud, B., Arnaldi, B., and Jannin, P. (2016). Synthesis and simulation of surgical process models. *Studies in health technology and informatics*, 220.

Table des figures

Annexes

Introduction aux Réseaux de Petri

A

Les réseaux de Petri sont issus de la thèse de Carl Adam Petri [Petri, 1962] [Petri, 1966]. Il s'agit d'un outil graphique pour la description et l'analyse de processus concurrents dans un système [Ladet, 1989]. Ils sont très utilisés car ils possèdent de nombreuses propriétés mathématiques connues [Glynn, 1987] [Murata, 1989].

1 Formalisme, Représentation et Fonctionnement

Un réseau de Petri est un graph orienté composé de **places** P et de **transitions** T. Les places sont reliées à des transitions et les transitions à des places. Une place est dite *en amont* d'une transition si un arc va de la place vers la transition. A l'inverse, elle est dite *en aval* si un arc va de la transition vers la place. Une place peut être en amont et/ou en aval d'une ou plusieurs transitions (voir figure A.2). Une place peu contenir des **jetons**. Une transition représente un *évènement instantané pouvant avoir lieu dans le système*. Enfin, un arc possède un poids. Par défaut, le poids d'un arc est 1.

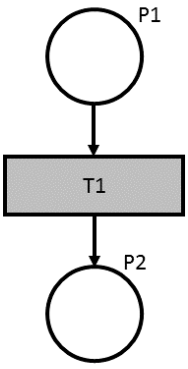


Figure A.1 – Un réseau de Petri simple. P_1 est une place amont de T_1. P_2 est une place aval de T_1.

Une transition est dite *sensibilisée* si toutes ses places en amont possèdent un nombre de jetons au moins égal au poids de l'arc. Si une transition est sensibilisée, alors elle peut être

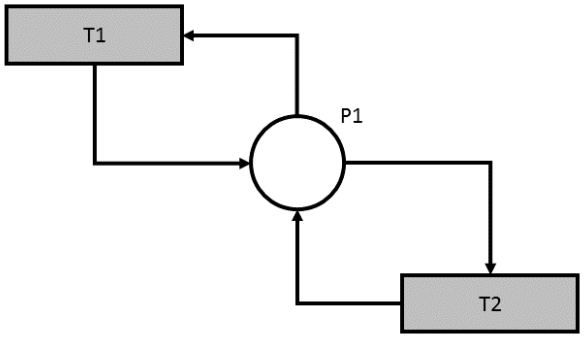


Figure A.2 – Ici P_1 est amont et aval de T_1 et T_2.

déclenchée. Dans ce cas, elle consomme dans chaque place amont un nombre de jetons égal au poids de l'arc. Puis elle produit un nombre égal au poids de l'arc dans chaque place en aval. Chaque déclenchement d'une transition modifie le nombre de jetons dans un certain nombre de places. La figure A.3 montre une exécution d'un réseau simple dont les arcs sont tous pondérés à 1.

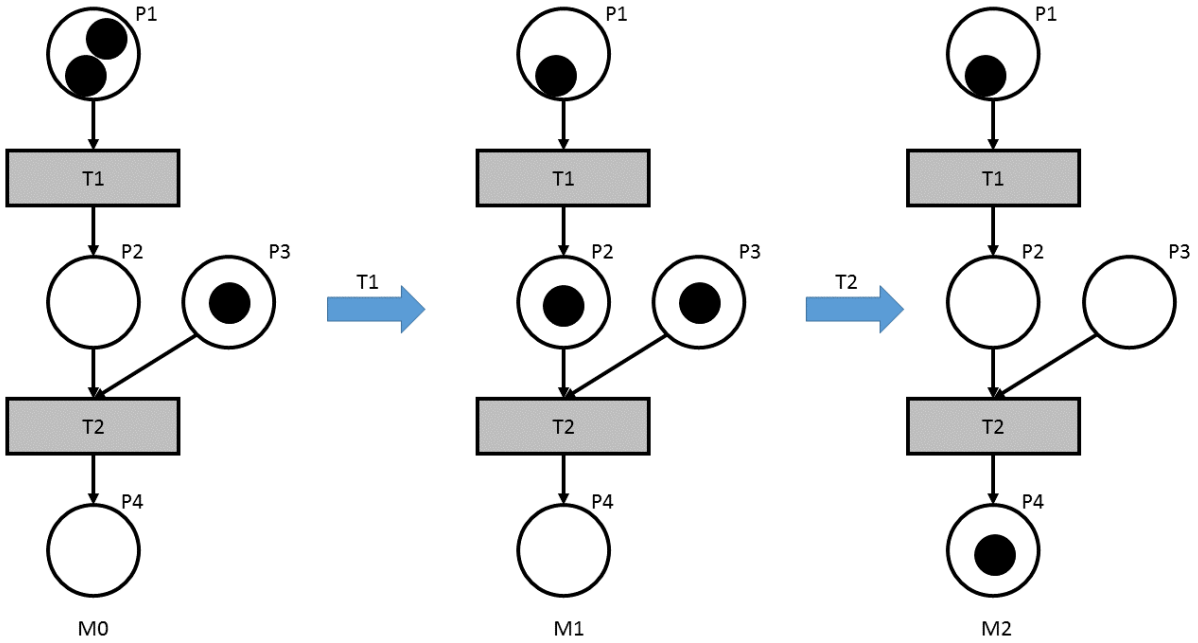


Figure A.3 – Exemple d'exécution d'un réseau de Petri.

L'état du réseau peut être représenté par un vecteur qui associe à chaque place le nombre jetons présents. Ce vecteur M est appelé *marquage* du réseau. Le marquage initial du réseau est

celui qui définit son état de départ. L'ordre d'activation des transitions définit les marquages atteints successivement par le réseau. Par exemple, dans la figure A.3 la transition T_1 est toujours sensibilisée tant qu'il y a au moins un jeton en P_1. Le marquage M_1 est atteint par le déclenchement de T_1 en partant du marquage M_0. Le marquage M_2 est atteint par le déclenchement de T_2 depuis le marquage M_1. Dans M_0 et M_2, T_2 n'est pas sensibilisée. Dans les marquages présents, T_1 est toujours sensibilisée. Il est possible d'obtenir d'autres marquages que M_1 et M_2 en fonction de l'ordre d'activation de T_1 et T_2. Initialement, les transitions d'un réseaux de Petri sont activées aléatoirement. Mais il est possible d'utiliser d'autres algorithmes afin, par exemple, de rendre l'exécution d'un réseau déterministe.

Un des avantages des réseaux de Petri est qu'ils sont capables de représenter la concurrence et le parallélisme sans ajouter d'autres composants. Plusieurs transitions ayant au moins une place amont en commun sont en concurrence. Une transition ayant plusieurs places en aval permet de représenter un parallélisme potentiel. La figure A.4 présentent des cas de concurrence et de parallélisme. Dans Le réseau R_c T_{12} est en concurrence avec la sous séquence $T_{11}, T13$. Dans le réseau R_p T_{12} et T_{13} sont en parallèle. Cependant, les deux doivent être déclenchées pour pouvoir sensibiliser T_{14}.

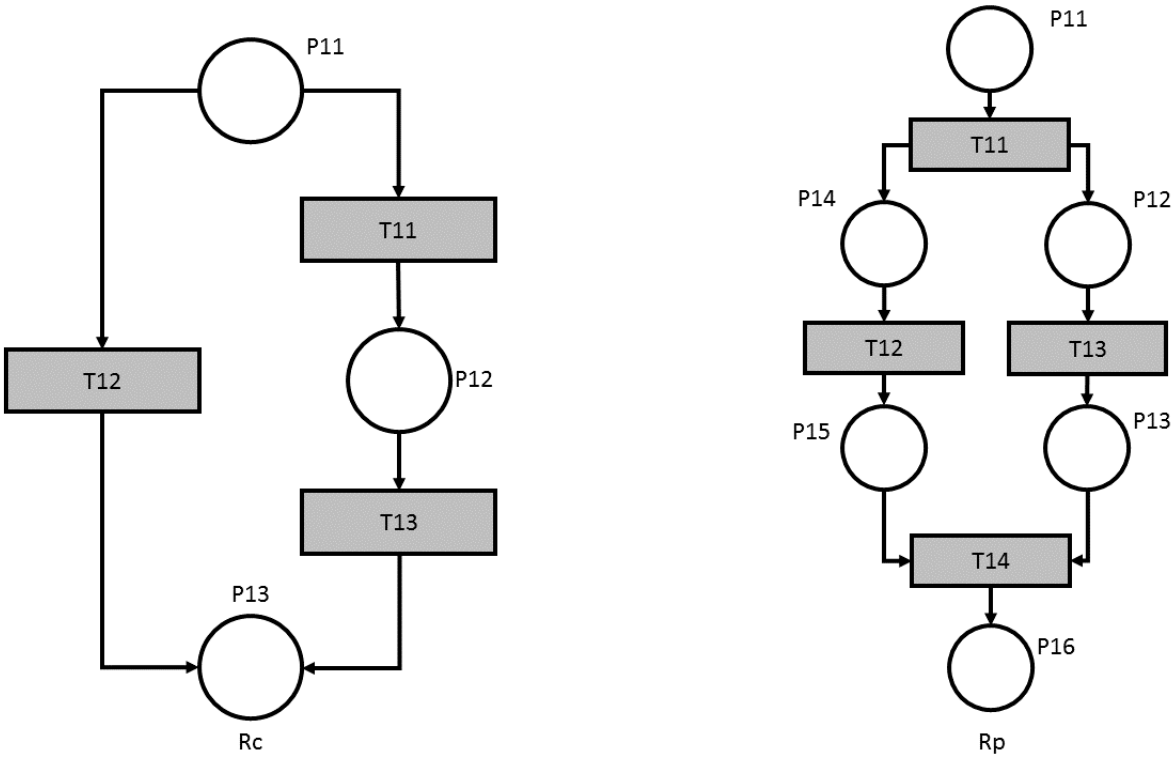


Figure A.4 – Concurrence et parallélisme dans les réseaux de Petri.

2 Réseaux Bornés

Une place d'un réseau de Petri est bornée si elle ne peut accueillir qu'un nombre fini de jetons n. Un réseau de Pétri est n-borné si toutes ses places sont bornées à n pour un marquage initial m_0 donné. C'est-à-dire que, quel que soit l'exécution du réseau, il n'atteindra jamais un marquage ou une de ses places porte plus de n jetons. Par conséquent, le poids de ses arcs n'est jamais supérieur à n. Comme le nombre de places dans un réseau est fini, si un réseau est borné alors le nombre de marquages possibles du réseau est fini. Dans ce cas, le réseau est donc un automate fini. Un réseau de Petri 1-borné est appelé un réseau de Petri **sauf**.

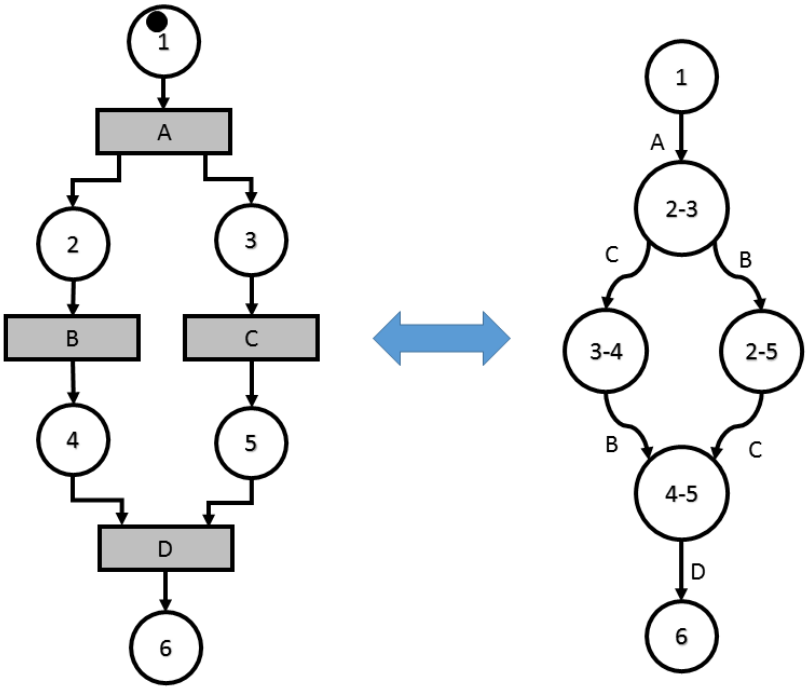


Figure A.5 – Un réseau de Petri sauf et une machine à états équivalente.

Les réseaux de Petri saufs sont équivalents à des machines à états finis. Néanmoins, ils sont capables d'exprimer clairement le parallélisme ce qui permet des représentations plus simples dans des cas complexes (utilisant de nombreux parallélismes et concurrences). La Figure donne un exemple basique d'un réseau de Petri sauf utilisant du parallélisme et une machine à états équivalente. Le parallélisme est géré au niveau de la machine à état en représentant tous les séquencements de B et C possibles. Les états ne sont pas équivalents aux places du réseau, mais à ses marquages. L'état 2-3 représente le marquage du réseau après le déclenchement de la transition A, les marquages 3-4 et 2-5 représentent respectivement l'état du réseau après le déclenchement en premier de C ou de B. L'état 4-5 représente l'état du réseau après B et C dans

n'importe quel ordre. Dans le cas de réseaux complexes le nombre de marquages possibles (et donc le nombre de séquencements possibles) peut être très élevé comparativement au nombre de places. On assiste alors à une explosion combinatoire du nombre d'états de la machine à états qui devient alors bien plus difficile à utiliser.

3 Calcul d'accessibilité

Un marquage M_j est dit accessible depuis un marquage M_i s'il existe au moins une séquence S_{ij} de déclenchement de transitions (et donc de marquages successifs) qui mène de M_i à M_j. Des solutions existent pour calculer l'accessibilité d'un marquage [Mayr, 1984] [Ezpeleta et al., 1995]. Ceci permet, notamment, de prévoir des inter-blocages.

4 Réseaux Équivalents

Une des propriétés des réseaux de Petri qui nous intéresse est la possibilité de calculer des réseaux équivalents. Ceci permet de simplifier la représentation d'un réseau en gardant ses propriétés mathématiques. Notamment le calcul d'accessibilité.

4.1 Pliage et Dépliage d'un réseau

Nous appelons une *classe d'équivalence* d'évènement un ensemble d'évènement qui possèdent des caractéristiques communes. Si un réseau R contient une transition T qui est déclenchée par n'importe quel évènement une classe d'évènement C dont les instances existantes sont a et b alors il existe un réseau R_{eq} équivalent à R ou T est remplacée par deux transitions T_a et T_b avec les même places amont et aval que T et qui sont déclenchées respectivement par a et par b et réciproquement. Nous appelons R_{eq} le réseau *déplié* de R. La figure A.6 montre les réseaux équivalents R et R_{eq}.

4.2 Réseaux Hiérarchiques

Un réseau hiérarchique contient des sous-réseaux dans certaines de ses places ou transitions. Dans ce cas, un réseau équivalent est calculable en remplaçant la place ou la transition hiérarchique par le sous-réseau. Dans le cas d'une place hiérarchique P_h certaines places du sous-réseau sont considérées comme en aval de toutes les transitions précédant P_h et d'autres sont considérées comme en amont de toutes les transitions suivant P_h. Dans le cas d'une transition hiérarchique T_h, le sous réseau commence et termine par deux transitions différentes T_{in} et T_{out}. Les places en amont de T_h sont alors en amont de T_{in} et les places en aval de T_h sont en aval de T_{out}.

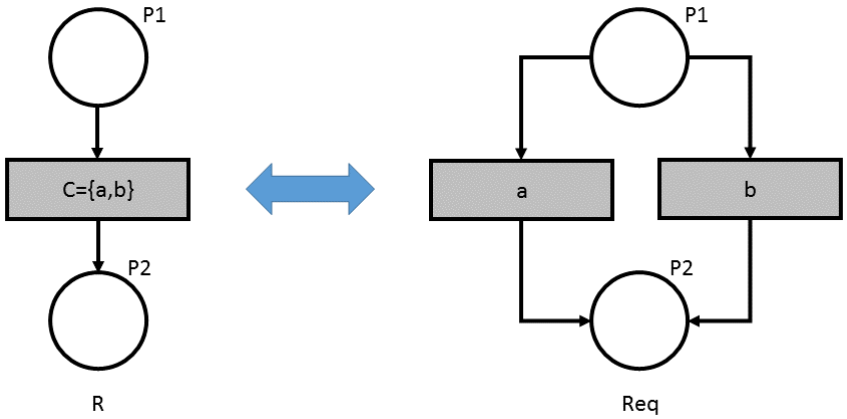


Figure A.6 – R_{eq} est équivalent au réseau R. On nous appelons R_{eq} le réseau R *déplié*.

#FIVE : Modèle de Représentation de l'Environnement Virtuel

B

#FIVE (Framework for Interaction in Virtual Environements) [Bouville et al., 2015] propose des abstractions et des mécanismes pour représenter les comportements d'un Environnement Virtuel. Il est divisé en deux modules : *Le Moteur de Relations* et *le Moteur d'Interactions Collaboratives*.

1 Moteur de Relations

Le moteur de relations permet de raisonner sur ce qu'il est possible de faire avec les objets de l'Environnement Virtuel. Il repose sur les entités suivantes :

- ***Les Types*** - Propriétés d'un objet et qui est requise pour son implication dans une relation. Par exemple *Tranchant* ou *en acier*.
- ***Les Objets*** - Éléments constituants l'Environnement Virtuel et pouvant intervenir dans des relations. Par exemple *un scalpel* ou *une clef*.
- ***Les Patrons d'Objets*** - Ensemble de Types liées qui décrivent un ensemble d'objets. Par exemple le patron *instrument d'incision* est constitué des types *tranchant*, *en acier* et *prenable*.
- ***Les Relations*** - Modèles d'actions. Définit les patrons nécessaires aux objets pouvant intervenir dans la réalisation d'une action. Par exemple l'action *inciser* demande deux objets distincts : un correspondant au patron *instrument d'incision* et l'autre au patron *cible d'incision*.
- ***Les Réalisations*** - Instance Concrète d'une relation. Utilise des instances spécifiques des objets de l'environnement. Une réalisation permet :
  - De vérifier que l'action utilisant un ensemble spécifique d'objet est réalisable. Par exemple en vérifiant que l'état de ces objets correspond à certaines pré-conditions.
  - De réaliser effectivement cette action en exécutant un comportement issu des différents composants impliqués.

Le moteur de relation permet de répondre à des questions telles que "*Dans quelles relations ces objets peuvent-ils intervenir ?*", "*Quels objets peuvent réaliser cette relation (en prenant tous les objets déclarés) ?*" ou "*Quels objets me manquent pour réaliser cette relation (spécification préalable d'une partie des objets) ?*". La Figure B.1 donne un exemple d'utilisation du moteur de relations : la relation R peut être réalisée par deux ensembles d'objets différents : A et C ou A et D. A la question "*Quelles sont les réalisations de R possibles ?*", le moteur répondra donc Ra et Rb.

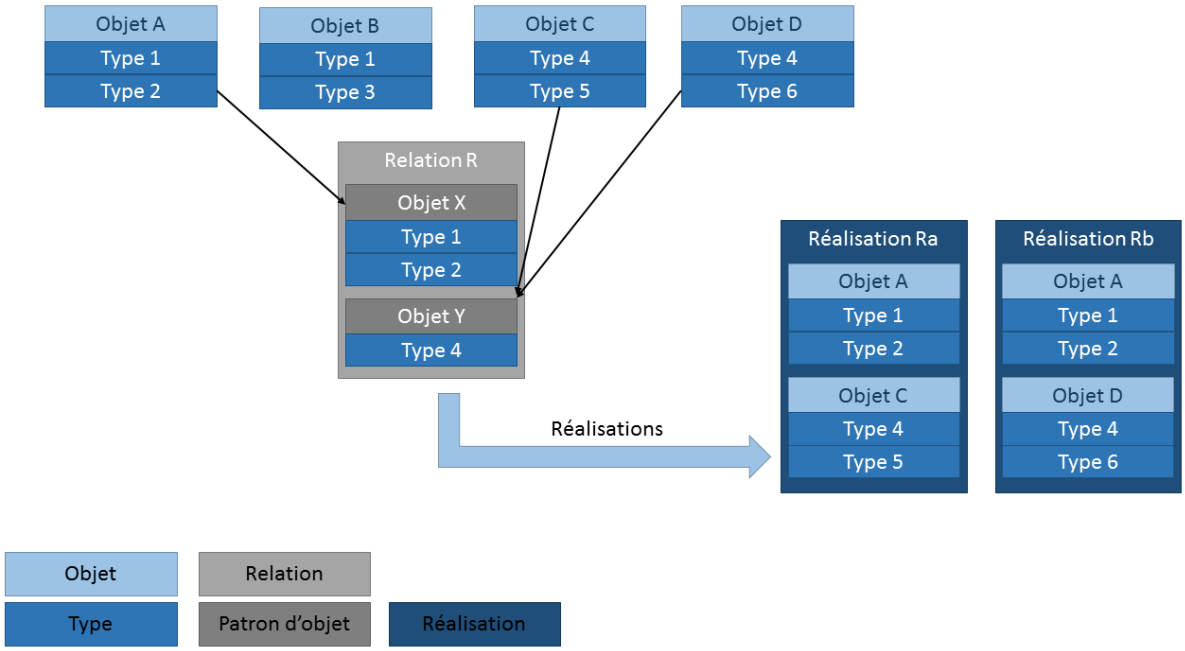


Figure B.1 – Le moteur de relations permet de raisonner sur les actions possibles en utilisant des ensembles d'objets de l'Environnement Virtuel.

2 Moteur d'Interaction Collaboratives

Le Moteur d'Interactions Collaboratives utilise des modèles et mécanismes pour le contrôle collaboratif d'environnements Virtuels. Il repose sur différents types d'entité :

- ***Les Objets Interactifs*** sont des objets de l'environnement qui possèdent des *paramètres contrôlables*. Par exemple une position dans l'espace.
- ***Les Paramètres Contrôlables*** sont des paramètres qui peuvent être modifiés par une interaction. Par exemple la position d'un objet sur l'axe X. Il est défini par un identifiant (e.g. x_pos), un type (e.g. nombre flottant), un *gestionnaire d'accès* et *un unificateur*.

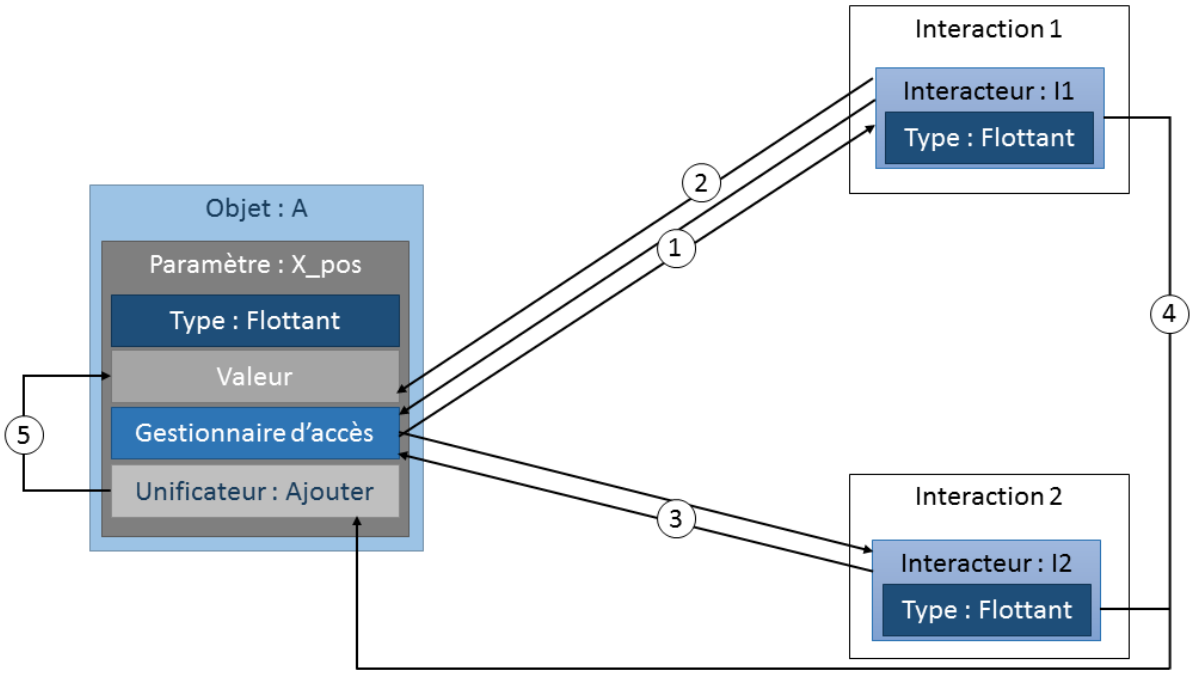


Figure B.2 – Le moteur d'interaction permet de réaliser des interaction collaborative comme la co-manipulation d'objets.

- ***Les Gestionnaires d'accès*** régissent les accès aux paramètres.
- ***Les Unificateurs*** décrivent les règles à appliquer lorsque des accès concurrents à un paramètre sont induits par des interactions collaboratives. Par exemple deux accès à x_pos : un acteur tire l'objet et l'autre le pousse.
- ***Les Interacteurs*** peuvent modifier un *paramètre contrôlable* (voir Figure B.2). Un interacteur possède un type. Ce type définit le type de paramètre que l'interacteur peut modifier. Lors d'une tentative d'interaction collaborative, un interacteur demande l'autorisation au gestionnaire d'accès (1) de pouvoir modifier un paramètre. L'interacteur définit lui même comment il modifie le paramètre (2). Si au moins un autre interacteur demande l'accès au même paramètre (3), les règles définies par l'unificateur sont appliquées aux valeurs fournies par les interacteurs (4) et le résultat est appliqué à la valeur du paramètre (5).

3 Conclusion

Le moteur de Relation est le module qui est le plus utilisé par les contributions présentées dans ce manuscrit. Le moteur d'Interaction Collaborative offre la possibilité de représenter

dans l'Environnement Virtuel les interactions collaboratives comme les co-manipulations par plusieurs acteurs.

Bibliographie

Amokrane, K., Lourdeaux, D., and Michel, G. (2014). Serious game based on virtual reality and artificial intelligence. In *ICAART (1)*.

Arnaldi, B., Fuchs, P., and Guiton, P. (2006). Introduction à la réalité virtuelle. In *Les Applications de la Réalité Virtuelle*, volume 4 of *Le Traité de la Réalité Virtuelle*. Les Presses de l'Ecole de Mines de Paris.

Aylett, R. S., Louchart, S., Dias, J., Paiva, A., and Vala, M. (2005). Fearnot!–an experiment in emergent narrative. In *Intelligent virtual agents*. Springer.

Badawi, M. and Donikian, S. (2004). Autonomous agents interacting with their virtual environment through synoptic objects. *CASA*.

Badouel, E., Bernardinello, L., and Darondeau, P. (2016). *Petri Net Synthesis*. Springer.

Barot, C. (2014). *Sc narisation d'environnements virtuels. Vers un quilibre entre controle, coh rence et adaptabilit* . PhD thesis, UTC.

Barot, C., Lourdeaux, D., Burkhardt, J.-M., Amokrane, K., and Lenne, D. (2013). V3s: A virtual environment for risk-management training based on human-activity models. *PRESENCE: Teleoperators and Virtual Environments*, 22(1).

Berthelot, R. B., Lopez, T., Nouviale, F., Gouranton, V., and Arnaldi, B. (2014). Corvette: Collaborative environment for technical training and experiment. In *iEEE Virtual Reality*.

Biddle, B. and Thomas, E. (1966). Role theory: concepts and research.

Biddle, B. J. (1986). Recent development in role theory. *Annual review of sociology*.

Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129(1–2).

Bouville, R., Gouranton, V., and Arnaldi, B. (2016). Virtual reality rehearsals for acting with visual effects. In *International Conference on Computer Graphics & Interactive Techniques, GI*.

Bouville, R., Gouranton, V., Boggini, T., Nouviale, F., and Arnaldi, B. (2015). #five: High-level components for developing collaborative and interactive virtual environments. In *SEARIS*.

Brom, C. and Abonyi, A. (2006). Petri-nets for game plot. In *Proceedings of AISB artificial intelligence and simulation behaviour convention*, volume 3.

Brom, C., Šisler, V., and Holan, T. (2007). Story manager in europe 2045 uses petri nets. In *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling*. Springer.

Buche, C., Querrec, R., De Loor, P., and Chevaillier, P. (2003). Mascaret: pedagogical multi-agents systems for virtual environment for training. In *Cyberworlds*.

Caillaud, B. (2013). Surgical process mining with test and flip net synthesis. In *Application of Region Theory*.

Carpentier, K. (2015). *Scénarisation personnalisée dynamique dans les environnements virtuels pour la formation.* PhD thesis, UTC.

Carpentier, K., Lourdeaux, D., and Thouvenin, I. (2013). Dynamic selection of learning situations in virtual environment. In *5th International Conference on Agents and Articificial Intelligence*.

Cavazza, M., Donikian, S., Christie, M., Spierling, U., Szilas, N., Vorderer, P., Hartmann, T., Klimmt, C., André, E., Champagnat, R., et al. (2008). The iris network of excellence: Integrating research in interactive storytelling. In *Interactive storytelling*. Springer.

Cavazza, M., Lugrin, J.-L., Pizzi, D., and Charles, F. (2007). Madame bovary on the holodeck: immersive interactive storytelling. In *ACMMM*.

Cavazza, M. O. M., Charles, F. F., and Mead, S. J. S. (2002). Character-based interactive storytelling. *IEEE Intelligent Systems*, 17(4).

Ceusters, W. (2012). An information artifact ontology perspective on data collections and associated representational artifacts. In *MIE*.

Chevaillier, P., Trinh, T.-H., Barange, M., De Loor, P., Devillers, F., Soler, J., and Querrec, R. (2012). Semantic modeling of virtual environments using MASCARET. In *Software Engineering and Architectures for Realtime Interactive Systems*.

Collinot, A. and Drogoul, A. (1998). Using the cassiopeia method to design a robot soccer team. *Applied Artificial Intelligence*, 12(2-3).

Cremer, J., Kearney, J., and Papelis, Y. (1995). HCSM: A framework for behavior and scenario control in virtual environments. *ACM Trans. Model. Comput. Simul.*, 5(3).

DAVID, A. (1989). Du grafcet aux réseaux de petri.

Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems. In *EuroCog*.

Donikian, S. (2001). Hpts: a behaviour modelling language for autonomous agents. In *Proceedings of the fifth international conference on Autonomous agents*.

Dougiamas, M. and Taylor, P. (2003). Moodle: Using learning communities to create an open source course management system.

Doyen, E. (1902). Les progrès récents de la chirurgie. *La revue Illustrée*, 9.

Eco, U. (1985). *Lector in fabula*. Grasset.

Ezpeleta, J., Colom, J. M., and Martinez, J. (1995). A petri net based deadlock prevention policy for flexible manufacturing systems. *Robotics and Automation, IEEE Transactions on*, 11(2).

Ferber, J. and Gutknecht, O. (2000). Operational semantics of multi-agent organizations. In *Intelligent Agents VI*.

Ferber, J., Gutknecht, O., and Michel, F. (2004). From agents to organizations: An organizational view of multi-agent systems. In *Agent-Oriented Software Engineering IV*.

Garraud, C., Gibaud, B., Penet, C., Cazuguel, G., Dardenne, G., and Jannin, P. (2014). An ontology-based software suite for the analysis of surgical process model.

Gerbaud, S. (2008). *Contribution à la formation en réalité virtuelle: scénarios collaboratifs et intégration d'humains virtuels collaborant avec des utilisateurs réels*. PhD thesis, INSA Rennes.

Gerbaud, S., Gouranton, V., and Arnaldi, B. (2009). Adaptation in collaborative virtual environments for training. In *Learning by Playing. Game-based Education System Design and Development*, volume 5670 of *Lecture Notes in Computer Science*.

Gerbaud, S., Mollet, N., and Arnaldi, B. (2007). Virtual environments for training: From individual learning to collaboration with humanoids. In *Technologies for E-Learning and Digital Entertainment*, volume 4469 of *LNCS*. Springer Berlin Heidelberg.

Gerbaud, S., Mollet, N., Ganier, F., Arnaldi, B., and Tisseau, J. (2008). Gvt: a platform to create virtual environments for procedural training. In *IEEE Virtual Reality Conference*.

Gibaud, B., Penet, C., and Jannin, P. (2014). Ontospm: a core ontology of surgical procedure models. SURGETICA.

Glynn, W. (1987). Petri nets, algebras, morphisms, and compositionality. *Information and Computation*, 72(3).

Hall, J., Ellis, C., and Hamdorf, J. (2003). Surgeons and cognitive processes. *British Journal of Surgery*, 90(1).

Hannoun, M., Boissier, O., Sichman, J. S., and Sayettat, C. (1999). Moïse: Un modèle organisationnel pour la conception de systèmes multi-agents. *JFIADSMA*, 99.

Hannoun, M., Boissier, O., Sichman, J. S., and Sayettat, C. (2000). Moise: An organizational model for multi-agent systems. In *Advances in Artificial Intelligence*.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3).

Hill Jr, R. W., Gratch, J., Marsella, S., Rickel, J., Swartout, W. R., and Traum, D. R. (2003). Virtual humans in the mission rehearsal exercise system. *KI*, 17(4).

Hosseini, M. and Georganas, N. D. (2001). Collaborative virtual environments for training. In *Proceedings of the ninth ACM international conference on Multimedia*.

Johnson, W. L. and Rickel, J. (1997). Steve: An animated pedagogical agent for procedural training in virtual environments. *ACM SIGART Bulletin*, 8.

Kallmann, M. and Thalmann, D. (1999). Modeling objects for interaction tasks. In *Computer Animation and Simulation'98*.

Koper, R. and Olivier, B. (2003). Representing the learning design of units of learning.

Kubera, Y., Mathieu, P., and Picault, S. (2011). Ioda: an interaction-oriented approach for multi-agent based simulations. *Autonomous Agents and Multi-Agent Systems*, 23(3).

Ladet, P. (1989). Réseaux de petri. *Techniques de L'ingénieur, r7252*, Agroalimentaire(Archives).

Lamarche, F. and Donikian, S. (2002). Automatic orchestration of behaviours through the management of resources and priority levels. In *International Joint Conference on Autonomous Agents and Multiagent Systems*.

Lanquepin, V., Carpentier, K., Lourdeaux, D., Lhommet, M., Barot, C., and Amokrane, K. (2013). Humans: a human models based artificial environments software platform. In *Proceedings of the Virtual Reality International Conference: Laval Virtual*.

Larousse, D. (2016). *Définition : scénario, scénarios ou scénarii*. http://www.larousse.fr/.

Le Corre, F., Hoareau, C., Ganier, F., Buche, C., and Querrec, R. (2014). A pedagogical scenario language for virtual environment for learning based on uml meta-model. application to blood analysis instrument. In *International Conference on Computer Supported Education (CSEDU)*.

Lugrin, J.-L. and Cavazza, M. (2006). AI-based world behaviour for emergent narratives. In *International Conference on Advances in Computer Entertainment Technology*.

Luna, A. S., Gouranton, V., Lopez, T., and Arnaldi, B. (2013). The perceptive puppet: Seamless embodiment exchange between real and virtual humans in virtual environments for training. In *GRAPP, International Conference on Computer Graphics Theory and Applications*.

Lutz, M. (2010). *Programming python*. " O'Reilly Media, Inc.".

Magerko, B., Laird, J., Assanie, M., Kerfoot, A., and Stokes, D. (2004). Ai characters and directors for interactive computer games. *Ann Arbor*, 1001(48).

Mallot, H. A. (1997). Behavior oriented approaches to cognition: Theoretical perspectives. *Theory in Biosciences*, 116.

Margery, D., Arnaldi, B., and Plouzeau, N. (1999). A general framework for cooperative manipulation in virtual environments. In *Virtual Environments*, Eurographics.

Marion, N., Querrec, R., and Chevaillier, P. (2009). Integrating knowledge from virtual reality environments to learning scenario models-a meta-modeling approach. In *CSEDU (1)*.

Marion N., Septseault C., B. A. and Querrec, R. (2007). GASPAR : Aviation management on an aircraft carrier using virtual reality. In *Cyberworlds, 2007. CW '07. International Conference on*.

Mateas, M. and Stern, A. (2002). A behavior language for story-based believable agents. *IEEE Intelligent Systems*, 17(4).

Mayr, E. W. (1984). An algorithm for the general petri net reachability problem. *SIAM Journal on computing*, 13(3).

Mollet, N. and Arnaldi, B. (2006). Storytelling in virtual reality for training. In *Technologies for E-Learning and Digital Entertainment*. Springer.

Mollet, N., Gerbaud, S., Arnaldi, B., et al. (2007). Storm: a generic interaction and behavioral model for 3d objects and humanoids in a virtual environment. In *IPT-EGVE the 13th Eurographics Symposium on Virtual Environments*.

Moreau, G. (2006). Modèles géométriques des environnements virtuels. In *Les Outils et Modèles Informatiques des Environnements Virtuels*, volume 3 of *Le Traité de la Réalité Virtuelle*. Les Presses de l'Ecole de Mines de Paris.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4).

Oliveira, J. C., Hosseini, M., Shirmohammadi, S., Cordea, M., Petriu, E., Petriu, D., and Georganas, N. D. (2000a). Virtual theater for industrial training: A collaborative virtual environment. In *WORLD MULTICONFERENCE on Circuits, Systems, Communications & Computers*.

Oliveira, J. C., Shen, X., and Georganas, N. D. (2000b). Collaborative virtual environment for industrial training and e-commerce. *IEEE VRTS*, 288.

Oliveira, J. C., Shirmohammadi, S., and Georganas, N. D. (2000c). A collaborative virtual environment for industrial training. In *Virtual Reality, 2000. Proceedings. IEEE.*

Paiva, A., Machado, I., and Prada, R. (2001). Heroes, villians, magicians : dramatis personae in a virtual story creation environment. In *ACM IUI.*

Peterson, J. L. (1981). Petri net theory and the modeling of systems.

Petri, C. A. (1962). *Kommunikation mit automaten.* PhD thesis.

Petri, C. A. (1966). *Communication with automata.* PhD thesis.

Piaget, J. (1976). *Le comportement, moteur de l'évolution*, volume 354.

Pimentel, K. and Teixeira, K. (1993). Virtual reality through the new looking glass.

Ponder, M., Herbelin, B., Molet, T., Schertenlieb, S., Ulicny, B., Papagiannakis, G., Magnenat-Thalmann, N., and Thalmann, D. (2003). Immersive VR decision training: Telling interactive stories featuring advanced virtual human simulation technologies. In *Workshop on Virtual Environments*, EGVE. ACM.

Querrec, R., Buche, C., Maffre, E., and Chevaillier, P. (2003). Sécurévi: virtual environments for fire-fighting training. In *5th virtual reality international conference (VRIC 03).*

Querrec, R., Buche, C., Maffre, E., and Chevaillier, P. (2004). Multiagents systems for virtual environment for training. application to fire-fighting. *International Journal of Computers and Applications (IJCA)*, 1(1).

Querrec, R. and Chevaillier, P. (2001). Virtual storytelling for training: An application to fire fighting in industrial environment. In *Virtual Storytelling Using Virtual Reality Technologies for Storytelling.* Springer.

Querrec, R., Reignier, P., and Chevaillier, P. (2001). Humans and autonomous agents interactions in a virtual environment for fire figthing training. In *Virtual Reality International Conference.*

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH computer graphics*, volume 21.

Rickel, J., Marsella, S., Gratch, J., Hill, R., Traum, D., and Swartout, W. (2002). Toward a new generation of virtual humans for interactive experiences. Technical report, DTIC Document.

Riedl, M. O. and Bulitko, V. (2012). Interactive narrative: An intelligent systems approach. *AI Magazine*, 34(1).

Rodriguez-Artacho, M. (2002). Palo language overview. *Universidad Nacional de Education a Distancia, Technical Report TR-2002-01 [Disponible en http://sensei. lsi. uned. es/palo/PALO-TR. pdf]*.

Rodríguez-Artacho, M. and Maíllo, M. F. V. (2004). Modeling educational content: the cognitive approach of the palo language. *Journal of Educational Technology & Society*, 7(3).

Rosse, C., Mejino Jr, J. L., et al. (2003). A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of biomedical informatics*, 36(6).

Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual*. Pearson Higher Education.

Saraos Luna, A., Gouranton, V., and Arnaldi, B. (2012). Collaborative Virtual Environments For Training: A Unified Interaction Model For Real Humans And Virtual Humans. In *International Conference on Serious Games and Edutainment*.

Shawver, D. M. (1997). Virtual actors and avatars in a flexible user-determined-scenario environment. In *vrais*.

Smith, B. and Grenon, P. (2002). Basic formal ontology. *Draft. Downloadable at http://ontology. buffalo. edu/bfo*.

Smith, S., Duke, D., and Massink, M. (1999). The hybrid world of virtual environments. In *Computer Graphics Forum*, volume 18.

Steuer, J. (1992). Defining virtual reality: Dimensions determining telepresence. *Journal of communication*, 42(4).

Sugiyama, K., Tagawa, S., and Toda, M. (1981). Methods for visual understanding of hierarchical system structures. *Systems, Man and Cybernetics*, 11(2).

Sutherland, I. E. (1965). The ultimate display. *Multimedia: From Wagner to virtual reality*.

Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*.

Swartout, W., Gratch, J., Hill, R., Hovy, E., Lindheim, R., Marsella, S., Rickel, J., and Traum, D. (2005). Simulation meets hollywood. In *Multimodal Intelligent Information Presentation*. Springer.

Swartout, W., Hill, R., Gratch, J., Johnson, W. L., Kyriakakis, C., LaBore, C., Lindheim, R., Marsella, S., Miraglia, D., and Moore, B. (2006a). Toward the holodeck: Integrating graphics, sound, character and story. Technical report, DTIC Document.

Swartout, W. R., Gratch, J., Hill Jr, R. W., Hovy, E., Marsella, S., Rickel, J., Traum, D., et al. (2006b). Toward virtual humans. *AI Magazine*, 27(2).

Szilas, N. (2003). Idtension: a narrative engine for interactive drama. In *Technologies for Interactive Digital Storytelling and Entertainment*, volume 3.

Taoum, J., Querrec, R., Saunier, J., and Blandin, B. (2015). East: Environnements d'apprentissage scientifiques et techniques. In *Les journées de l'AFRV 2015.*

Technologies, U. (2016). Unity3d. https://unity3d.com/.

Thalmann, D. (2000). Challenges for the research in virtual humans. In *Proc. AGENTS 2000*, number VRLAB-CONF-2007-064.

Theune, M., Faas, S., Heylen, D., and Nijholt, A. (2003). The virtual storyteller: Story creation by intelligent agents.

Traum, D., Rickel, J., Gratch, J., and Marsella, S. (2003). Negotiation over tasks in hybrid human-agent teams for simulation-based training. In *International joint conference on Autonomous agents and multiagent systems.*

van de Panne, M. and Fiume, E. (1993). Sensor-actuator networks. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93.

Willans, J. S. and Harrison, M. D. (2001a). Prototyping pre-implementation designs of virtual environment behaviour. In *Engineering for Human-Computer Interaction*, volume 2254 of *LNCS.*

Willans, J. S. and Harrison, M. D. (2001b). Verifying the behaviour of virtual environment world objects. In *Interactive Systems Design, Specification, and Verification*, volume 1946 of *LNCS.*

Xiang, Z., Mungall, C., Ruttenberg, A., and He, Y. (2011). Ontobee: A linked data server and browser for ontology terms. In *ICBO.*

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Séquencement d'actions en environnement virtuel collaboratif

Nom Prénom de l'auteur : CLAUDE GUILLAUME

Membres du jury :
- Monsieur JANNIN Pierre
- Monsieur ARNALDI BRUNO
- Madame GOURANTON Valérie
- Madame THOUVENIN Indira
- Monsieur GUITTON Pascal
- Monsieur CHEVAILLIER Pierre

Président du jury : Pierre JANNIN

Date de la soutenance : 12 Juillet 2016

Reproduction de la these soutenue

Thèse pouvant être reproduite en l'état
~~Thèse pouvant être reproduite après corrections suggérées~~

Fait à Rennes, le 12 Juillet 2016

Signature du président de jury
Pierre JANNIN

Le Directeur,

M'hamed DRISSI

Résumé

An interactive virtual environment alone cannot handle all the needs related to its usage. In Virtual Reality for training, it is important to guide the activity of the trainee to offer an efficient education. The problem is the same in interactive fictions: alone, the environment only offers a limited interest.

We are interested in the problem of the specification of the sequencing of actions in collaborative virtual environments. It is about defining, then controlling what must or can occur during the simulation in a potentially multi-user context. It is done, partly by the specification (and the execution) of a set of possible scenarios during a simulation session, and partly by the distribution of the feasible actions between the different actors (reals or virtuals) in the simulation.

First, we present #SEVEN, a Petri nets based model, allowing to describe causal and temporal sequencing of actions in an environment. #SEVEN is then used to answer to the problem of the specification of the possible scenarios and of the distribution of the actions between the actors. #SEVEN's properties made it a model able to be adapted to the needs of the specification of scenarios, especially because it can provide different guidance levels. As an example, it can define precisely the actions that need to be performed or indicates the changes that must occur in the state of the environment without defining precisely the actions to perform. Then, we address the problem of the distribution of the actions from perspective of the role theory, proposing a team model allowing to model the behaviours and rules related to a group of actors. This model allows to make evolve the possibilities of actions offered to the actors during the simulation depending on their abilities, their position in the team or the resources they have access to. The model also takes into account the state of the environment and the other team members.

These models have been implemented in a framework for the sequencing of actions in collaborative virtual environments used to develop several simulators, including those of the S3PM project (Synthesis and Simulation of Surgical Process Models). This project aims at building training systems for the collaborative work of medical staff in operating rooms from real cases observations.

Abstract

Un environnement virtuel interactif à lui seul ne permet pas de répondre à tous les besoins liés à son utilisation. Dans le cadre de la formation par Réalité Virtuelle, un cadrage de l'activité de l'apprenant est important pour offrir une formation efficace. Le problème est le même dans les fictions interactives : seul, l'environnement n'offre qu'un intérêt limité.

Nous nous intéressons ici au problème de la spécification du séquencement des actions dans un environnement virtuel collaboratif. Il s'agit de définir puis de contrôler ce qui peut ou doit se passer au cours de la simulation dans un contexte potentiellement multiutilisateur. Ceci passe, entre autres, par la spécification (puis l'exécution) d'un ensemble de scénarios possibles lors d'une session de simulation ainsi que par la distribution des actions réalisables entre les différents acteurs (réels ou virtuels) intervenant dans la simulation.

Nous présentons en premier lieu #SEVEN, un modèle fondé sur les réseaux de Petri, permettant de décrire des agencements temporels et causaux des actions dans un environnement. #SEVEN est ensuite utilisé pour répondre aux problèmes de la spécification de l'ensemble des scénarios possibles et de la distribution des actions entre les acteurs. Les propriétés de #SEVEN en font un modèle capable de s'adapter facilement aux besoins de la spécification de scénarios notamment, car il permet de fournir différents niveaux de guidage. Il peut par exemple définir précisément les actions à réaliser et l'ordre qu'elles doivent avoir ou encore indiquer les changements d'état de l'environnement devant avoir lieu sans préciser les actions nécessaires. Ensuite, nous abordons le problème de la distribution des actions à partir de la théorie des rôles en proposant un modèle d'équipe permettant de modéliser les comportements et règles liés à l'organisation d'un groupe d'acteur. Ce modèle permet de faire évoluer les possibilités d'action offertes aux acteurs au cours de la simulation suivant leurs compétences, leur position dans l'équipe, les ressources auxquelles ils ont accès. Ce modèle prend également en compte l'état de l'Environnement et les autres membres de l'équipe.

Ces modèles sont implémentés dans un framework pour le séquencement d'actions dans des environnements virtuels collaboratifs, utilisé dans la réalisation de simulateurs, notamment ceux du projet S3PM (Synthesis and Simulation of Surgical Process Models). Ce projet vise à réaliser des systèmes de formations de travail en équipe pour le personnel médical en salle d'opération à partir de l'observation de cas réels.

N° d'ordre : **16ISAR 13 / D16 - 13**

Institut National des Sciences Appliquées de Rennes
20, Avenue des Buttes de Coësmes - CS 14315 - F-35043 Rennes Cedex
Tél : 02 23 23 82 00 – Fax : 02 23 23 83 96