CHRYSAOR : un Système Tutoriel Intelligent pour les Environnements Virtuels d'Apprentissage Humain. Application à la formation au matériel de laboratoire en hémostase : application à la formation au matériel de laboratoire en hémostase

Frédéric Le Corre

université de bretagne occidentale

Thèse / Université de Bretagne occidentale
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de
Docteur de l'Université de Bretagne occidentale
Mention : Informatique
École doctorale Sicma

présentée par
Frédéric Le Corre
préparée au Centre Européen
de Réalité Virtuelle
Laboratoire LAB-STICC

CHRYSAOR : un Système Tutoriel Intelligent pour les Environnements Virtuels d'Apprentissage Humain. Application à la formation au matériel de laboratoire en hémostase.

Thèse soutenue le 12 juillet 2013
devant le jury composé de :

Dorin-Mircea Popovici (rapporteur)
Enseignant-chercheur, Ovidius, Roumanie

Simon Richir (rapporteur)
Professeur des Universités, ENSAM, Angers

Philippe Allain (examinateur)
Professeur des Universités, CHU, Angers

Vincent Rodin (examinateur)
Professeur des Universités, UBO

Querrec Ronan (directeur)
Maître de conférence, ENIB

Cédric Buche (encadrant)
Maître de Conférence, ENIB

Sébastien Kerdélo (invité)
Responsable de service R&D
DIAGNOSTICA STAGO, Gennevilliers

Université de Bretagne Occidentale

— Thèse —

Section informatique

CHRYSAOR : un Système Tutoriel Intelligent pour les Environnements Virtuels d'Apprentissage Humain. Application à la formation au matériel de laboratoire en hémostase.

Frédéric Le Corre

Soutenue le 12 juillet 2013 devant le jury :

Rapporteurs			
Dorin-Mircea	Popovici	Enseignant-chercheur	Université Ovidius, Roumanie
Simon	Richir	Professeur des universités	ENSAM, Angers
Examinateurs			
Philippe	Allain	Professeur des universités	CHU, Angers
Vincent	Rodin	Professeur des universités	UBO, Brest
Encadrant			
Cédric	Buche	Maître de conférences (HDR)	ENIB, Brest
Directeur			
Ronan	Querrec	Maître de conférences (HDR)	ENIB, Brest
Invité			
Sébastien	Kerdélo	Responsable de service R&D	DIAGNOSTICA STAGO, Gennevilliers

Frédéric Le Corre

125-131 Avenue Louis Roche
92230, Gennevilliers
tel : +33 (0)2 98 05 89 68
e-mail : frederic.lecorre@stago.com

Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance (Lab-STICC)

Centre Européen de Réalité Virtuelle (CERV)

Technopôle Brest-Iroise
25, rue Claude Chappe
BP 38 F-29280 Plouzané
tel : +33 (0)2 98 05 89 50
url : http://www.cerv.fr
url : http://www.labsticc.fr

Table des matières

Table des figures

Introduction

Contexte industriel

Cette thèse en informatique est réalisée en partenariat avec la société Stago [1].

Diagnostica Stago (Stago) est une société de l'industrie du diagnostic *in vitro* entièrement dédiée à l'exploration de l'hémostase (processus physiologique qui permet d'interrompre le saignement pour éviter l'hémorragie) et de la thrombose (formation d'un caillot dans un vaisseau). Fondée en 1945, elle est aujourd'hui *leader* mondial dans le secteur de l'hémostase. Les services proposés par cette entreprise sont la vente d'instruments de diagnostic, tel que le STA-R® (Figure 1), mais également des consommables (cuvettes de test, reducers, etc.) et des réactifs (produits chimiques utilisés par l'instrument pour réaliser les tests).

Figure 1 – Un instrument de diagnostic en hémostase : le STA-R®

1. http ://www.stago.fr/

Comme dans tous les domaines à forte technicité, dans le domaine du biomédical il est nécessaire de former les personnes à l'utilisation et à la maintenance des instruments et automates de diagnostic. Cependant la formation à ces appareils présente plusieurs contraintes, notamment lorsque cette dernière a lieu chez le client. Par exemple, avant chaque déplacement, une décontamination est nécessaire (à cause des risques biologiques), ce qui demande du temps et l'intervention de plusieurs techniciens. De plus, certains instruments de bioanalyse sont très volumineux, à chaque transport il est donc nécessaire de faire intervenir à nouveau plusieurs personnes. Enfin, ce sont des instruments ayant un coût relativement important, il y a donc également un risque de frais supplémentaires en cas de casse au cours du transport.
Pour éviter ces problèmes, il existe des formations chez le client qui dispose déjà du système, et dans ce cas plusieurs employés y ont accès. Cependant dans les laboratoires comme dans les hôpitaux, il y a énormément de roulements chez les employés, et les derniers rentrants doivent se former sur le tas, sans aucune formation préalable.
Au vu de ces contraintes, les formations se font généralement sur le site du constructeur : c'est la méthode de formation pratiquée la plus courante dans le domaine biomédical. Dans ces formations, plusieurs apprenants se retrouvent dans une classe avec un formateur. Malheureusement la majorité du personnel du client ne peut pas participer à cette formation. Les deux types de formation actuellement proposées par Stago sont celles présentées précédemment, c'est-à-dire formation par un instructeur sur le site de l'entreprise ou formation chez le client.

Actuellement, les technologies de réalité virtuelle sont très utilisées dans la formation et l'éducation pour une multitude de domaines (*e.g.* armée [Gerbaud et al., 2008], aviation [Buche et Querrec, 2011], écoles [Popovici et al., 2004], ...). Un des avantages des environnements virtuels de formation est qu'il y a une abstraction de toutes les contraintes citées précédemment. En effet, grâce à la réalité virtuelle, les apprenants peuvent se former sur un simple ordinateur où ils le souhaitent et quand ils le souhaitent : il n'y a donc plus autant de contraintes de temps ou de lieu. Ils peuvent répéter ces formations en environnement virtuel aussi souvent qu'ils le souhaitent et manipuler les instruments sans aucun risque de dégâts matériels ou humains. Il y a donc une plus grande liberté de formation pour les utilisateurs des instruments Stago. Bien plus, les environnements virtuels de formation sont particulièrement appropriés dans la mesure où ils proposent des situations à la fois réalistes, complexes et incertaines.

Nous proposons donc d'utiliser la réalité virtuelle pour la formation aux instruments de diagnostic biomédical Stago. Notons cependant que cette formation n'a pas pour vocation de remplacer la formation traditionnelle, mais qu'elle apparaît plutôt comme une formation complémentaire. Dans un premier temps nous avons choisi de baser notre application de formation par la réalité virtuelle sur un seul instrument : le STA-R®, qui est l'instrument haut de gamme Stago.

Contribution

L'utilisation de la réalité virtuelle pour la formation est relativement bien répandue et plusieurs travaux de recherche y sont consacrés [Rickel et Johnson, 1999; Lourdeaux, 2001; Popovici et al., 2004]. Ces environnements virtuels permettent d'avoir une très bonne

représentation de la réalité. Cependant, la simulation de la situation réelle seule ne suffit pas, il convient d'incorporer un scénario pédagogique pour construire l'apprentissage [Henri et al., 2007].

Scénario Pédagogique pour construire la formation

Nous souhaitons que le formateur puisse décrire lui même la situation d'apprentissage. Par exemple, le formateur pourrait vouloir organiser sa formation en plusieurs exercices :
- présentation des différentes sous-parties du système à étudier (*e.g.* un sous ensemble du STA-R®) ou des objets de l'environnement (*e.g.* les réactifs),
- réalisation d'une procédure par le formateur (*e.g.* la procédure de test),
- réalisation d'une procédure par l'apprenant : la réalisation de cette procédure dans un contexte pédagogique lui permet de bénéficier d'assistances pédagogiques.

Ce type de stratégie pédagogique est propre à chaque formateur, elle ne peut pas être directement contenue dans l'environnement virtuel. Elle doit donc être considérée comme une donnée d'entrée. Il s'agit donc d'externaliser sa conception. Ceci permet également à un même scénario d'être utilisé sur plusieurs systèmes.

Le scénario pédagogique dans l'Environnement Virtuel d'Apprentissage Humain (EVAH) permet d'organiser la formation dans un but pédagogique, cependant le scénario est conçu pour tout apprenant sans tenir compte des individualités, ce qui peut être un frein à l'apprentissage.

Système Tutoriel Intelligent pour individualiser la formation

Habituellement, le formateur décrit sa pédagogie dans le scénario. Cependant, il le fait de manière générale, c'est-à-dire sans différencier les apprenants. Pour pallier cela nous souhaitons coupler notre environnement virtuel à un Système Tutoriel Intelligent (STI). Cela permettra que la pédagogie s'adapte dynamiquement à l'apprenant au cours de la formation : la stratégie pédagogique sera propre à chaque individu. Un STI est un système informatique qui peut guider l'utilisateur de manière personnalisée et sans intervention humaine. Nous voulons notre STI générique, c'est-à-dire qu'il soit indépendant de l'application de réalité virtuelle. Le terme générique est un point crucial pour nous, car nous souhaitons que notre STI puisse être utilisé dans n'importe quel environnement virtuel de formation, que ce soit dans le domaine biomédical ou dans un tout autre domaine.

Verrous scientifiques

Dans la littérature, il existe de nombreux modèles de STI, cependant ils ne répondent pas à nos besoins.

Premièrement, il y a un manque de généricité et de modularité, par conséquent il est souvent nécessaire de modifier les STI dès que l'on change l'environnement, comme l'explique Buche

et Querrec [2011]. Par exemple, il faudrait modifier le STI à chaque fois que l'on souhaiterait changer d'exercice ou d'instrument de diagnostic. Le fait d'avoir un STI générique et modulaire le rendrait donc totalement indépendant du domaine de l'application et facilement modifiable.

Deuxièmement, afin que notre modèle de STI puisse correctement exploiter toutes les connaissances mises à sa disposition (*e.g.* les connaissances du scénario pédagogique), nous devrons travailler sur plusieurs caractéristiques importantes : le scénario pédagogique, l'individualisation et l'adaptabilité. Notons qu'il y a un réel manque pour le scénario pédagogique dans les modèles existants. Par exemple, nous pourrions souhaiter que notre STI participe au scénario pédagogique, soit en y proposant des aides, soit en se comportant directement comme le formateur. Tout cela nous amènera à utiliser différents concepts et différentes technologies, tels que les systèmes multi-agents. Nous pourrions également nous baser sur un modèle organisationnel afin de structurer notre scénario pédagogique et notre STI, grâce à la notion de rôle dans la structure organisationnelle.

Modèle : CHRYSAOR

Notre proposition, regroupant notre STI générique et le scénario pédagogique, porte le nom de Chrysaor. Concrètement, Chrysaor a des connaissances sur l'environnement virtuel et sur les interactions que peuvent avoir les apprenants avec cet environnement. Cela lui permet de raisonner sur ces connaissances et de proposer des aides à l'apprenant (comme par exemple guider l'apprenant s'il a fait une mauvaise action dans l'environnement). Chrysaor peut raisonner sur la tâche à effectuer, sur les connaissances de l'apprenant ainsi que sur le scénario pédagogique. Cela nous conduit à devoir abstraire toutes ces connaissances et les classer : les connaissances métier (Mascaret [Querrec et al., 2011]), les connaissances sur l'apprenant [El-Kechai, 2007] et les connaissances sur le scénario pédagogique [Laforcade, 2004; Marion et al., 2009]. Chrysaor peut raisonner sur toutes ces connaissances, c'est pourquoi il faut un modèle pour décrire :
– l'apprenant,
– l'organisation pédagogique,
– le scénario pédagogique,
– l'activité métier,
– l'environnement.
Pour cela, nous nous basons sur un méta-modèle tel que Mascaret. Chrysaor nous permet d'avoir un lien entre notre scénario pédagogique et le STI, car celui-ci pourra jouer un des rôles du scénario pédagogique.

Application

Notre proposition est appliquée à un environnement virtuel basé sur le STA-R®, appelée StarApplication (figure 2). L'apprenant peut naviguer dans la scène, interagir avec l'instrument et les divers objets présents, tout cela dans le but de se former à l'utilisation et à la maintenance du système. Cet environnement virtuel est associé à Chrysaor.

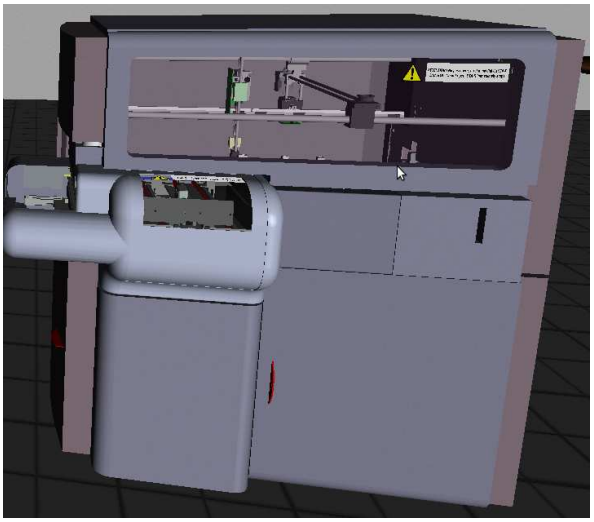

Figure 2 – Image de l'environnement virtuel représentant le STA-R® : l'application StarApplication

Pour évaluer notre proposition et notre application, nous avons réalisé deux expérimentations en collaboration avec une équipe de psychologues de l'Université de Bretagne Occidentale (UBO).

La première a été menée sur dix-huit étudiants issus d'une école d'ingénieurs, avec pour objectif de confirmer la mémorisation à long terme d'une procédure acquise par la réalité virtuelle. Les étudiants devaient réaliser plusieurs fois la procédure et ensuite revenir au bout d'une semaine pour la refaire. Afin de valider notre expérimentation, nous avons comparé différents résultats tels que le nombre de consultations des instructions, le nombre d'erreurs ou encore le temps total de réalisation de la procédure.

La seconde expérimentation a été menée sur soixante étudiants en formation BTS en analyse de biologie médicale (lycée Paul Eluard, Saint-Denis, 93) pendant quinze jours, avec cette fois-ci pour objectif de comparer l'efficacité de la formation par la réalité virtuelle (avec un STI) vis à vis de la formation traditionnelle (figure 3).

Pour notre application StarApplication, le scénario pédagogique a été décrit de manière à ce qu'il n'y ait qu'un seul exercice (une procédure de test biologique de routine). Cet exercice est composé du rôle apprenant joué par un étudiant et du rôle formateur joué par Chrysaor. Ces expérimentations nous ont donc permis de vérifier l'efficacité de la formation par la réalité virtuelle, ainsi que de conforter Stago à utiliser cette technologie.

Grâce à la modularité de Chrysaor, nous avons pu facilement décrire les différents comportements de notre tuteur (en l'occurence le blocage des mauvaises actions et le système de logs). Par la suite, si nous souhaitons obtenir un STI plus complexe, il nous suffira de modifier son organisation et ses comportements. Il y a également un autre point important : notre modèle de scénario pédagogique nous a permis de très facilement décrire un exercice et les rôles (apprenant et formateur) associés à la procédure. Grâce au lien de ces rôles avec Chrysaor, notre STI a pu jouer le rôle formateur décrit dans le scénario.

Figure 3 – Formation par la réalité virtuelle lors de l'expérimentation à Saint-Denis, 93

Plan

La suite de ce manuscrit est organisée de la manière suivante :

Dans le `chapitre 1`, nous présentons l'état de l'art. Nous explicitons notre choix des caractéristiques importantes pour les STI. Cela nous permettra par la même occasion de faire un bilan des différents STI existants.

Dans le `chapitre 2`, nous décrivons notre proposition. Nous présentons nos travaux qui se focalisent sur notre STI générique (appelé Chrysaor). Nous avons basé notre travail sur le méta-modèle Mascaret [Querrec et al., 2011] qui est également présenté dans ce même chapitre. Nos travaux portent sur l'utilisation des Systèmes Multi-Agents avec pour objectif de doter le STI de modularité, ainsi que sur le couplage avec un modèle de scénario pédagogique pour la description d'exercices de formation.

Dans le `chapitre 3`, nous présentons une application d'un environnement virtuel (StarApplication) utilisant Chrysaor comme exemple d'utilisation de notre STI. Nous y présentons également toutes les données sur notre expérimentation : du protocole aux résultats.

Pour terminer, une `conclusion` nous permet de dresser un bilan de ce qui a été fait, et les perspectives nous permettrons de définir les points restants à améliorer dans notre modèle Chrysaor, notamment au sujet de l'individualisation de la formation.

Chapitre 1

État de l'art

Ce chapitre propose de faire un état de l'art de plusieurs domaines importants pour nos travaux. Nous allons tout d'abord présenter dans la section 1.1 les possibilités de formation par la réalité virtuelle, appliquées au domaine du diagnostic *in vitro*. Dans la section 1.2 nous ferons un point sur les scénarios pédagogiques dans la littérature. De la même manière nous ferons un bilan sur les STI dans la section 1.3. Nous souhaitons utiliser les système multi-agents et permettre aux agents de raisonner sur les connaissances. Pour cela nous proposons un bilan sur la représentation de ces connaissances dans la section 1.4. Afin de représenter ces connaissances, nous nous appuyerons sur Mascaret que nous présenterons dans la section 1.5. Nous terminerons sur une synthèse de cet état de l'art dans la section 1.6.

1.1 Formation par la réalité virtuelle dans le domaine du diagnostic *in vitro*

1.1.1 Domaine du diagnostic *in vitro*

Le diagnostic *in vitro* définit le fait d'utiliser des produits, des réactifs, des matériaux, des instruments et systèmes, leurs composants et accessoires, afin de réaliser des examens d'échantillons provenant du corps humain. Ces tests fournissent une information concernant un état physiologique ou pathologique, avéré ou potentiel. Le marché du diagnostic *in vitro* a réalisé en 2011 un chiffre d'affaires de 1.79 milliards d'euros. Il se divise en deux grandes catégories, le marché des laboratoires de biologie qui représente 80% du secteur d'activité, et celui de l'autosurveillance glycémique qui en représente 20%.

Stago, qui finance cette thèse, se positionne sur le marché des laboratoires de biologie. En effet, Stago vend des instruments de diagnostic, des consommables (cuvettes de test, reducers, etc.) et des réactifs (produits chimiques utilisés par l'instrument pour réaliser les tests) afin que les laboratoires de biologie puissent tester la coagulation du sang de leurs patients.

Les clients des industriels du diagnostic *in vitro* sont répartis en quatre principales catégories dont trois représentent le marché des laboratoires de biologie. Ce marché est constitué des laboratoires de biologie médicale privés, des centres hospitaliers publics ou privés et des centres de transfusion sanguine, ce qui représente au total près de 79% des clients. Avec environ 20% de parts de marché, la catégorie des ventes aux grossistes répartiteurs constituent la deuxième grande activité de ce secteur.

1.1.2 Formations pratiquées

En ce qui concerne la formation dans ce domaine, nous avons étudié uniquement la manière dont elle est actuellement effectuée chez Stago et ses principaux concurrents. Ces entreprises ont des propositions de formation presque identiques. Il s'agit essentiellement de formation classique sur le site de l'entreprise, de formation chez le client ou de *e-learning*. Pour les formations aux personnels internes à Stago, il s'agit essentiellement de formations classiques sous forme de classe. Le *e-learning* (ou formation en ligne) consiste en un apprentissage par le biais de lecture d'un document (bien souvent de type Powerpoint) mis à disposition en utilisant Internet.

Formation classique sur site ou chez le client

Le plus souvent dans le cas de petites entreprises, la formation est effectuée dans les locaux de la société. Nous pouvons citer en exemple l'entreprise française Biomérieux [1]. En effet, ce type de formation est le moins coûteux : plusieurs clients se déplacent au même endroit. Cela permet donc de grouper les formations. Cependant cela force les clients à se déplacer.

Dans le cas d'entreprises de taille plus conséquente, les formations peuvent également se dérouler chez le client. Cela apporte un confort pour le client, mais des coûts plus importants pour l'entreprise.

Ces deux types de formation (sur le site de l'entreprise et chez le client) sont les plus répandus, on les retrouve chez la plupart des leaders du diagnostic *in vitro*, ainsi que dans dans les entreprises de domaines différents.

Un autre type de formation a commencé à devenir relativement courant depuis quelques années, il s'agit du *e-learning*.

Formation *e-learning*

En plus de la formation classique, certaines entreprises, telles que Siemens, Abbot, Sysmex ou Beckman Coulter proposent des formations par *e-learning*. La plupart des formations *e-learning* permettent aux utilisateurs de mettre à jour leurs connaissances quand ils le souhaitent et avec la durée qu'ils souhaitent. De plus, les apprenants n'ont pas besoin de se déplacer, ils peuvent avoir accès à cette formation de leur lieu de travail ou de leur domicile.

1. Site Web : http ://www.biomerieux.fr

Ce type de formation complémentaire est essentiellement proposée par des grandes entreprises, car ils ont les moyens financiers de diversifier leurs formations. Tout comme la majorité de ses concurrents, Stago propose les deux types de formation les plus répandus : traditionnelle (chez le client et sur le site de l'entreprise) et *e-learning*. Le *e-learning* chez Stago se présente sous la forme de documents type présentation Powerpoint qu'on peut consulter sur le site de l'entreprise.

Nous avons pu constater également que certaines entreprises font évoluer leurs formations de type *e-learning*.

Dans certaines entreprises la formation de type *e-learning* peut comporter également un module de discussion (texte ou vidéo) avec un formateur en ligne, comme par exemple dans l'entreprise Sysmex (Figure 1.1).

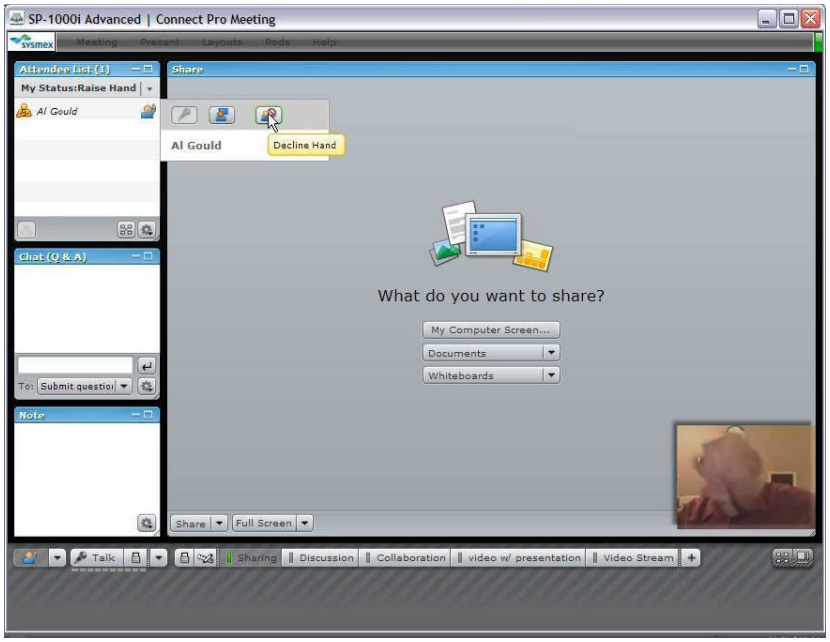

Figure 1.1 – Exemple de *e-learning* chez Sysmex

Dans cet exemple, le formateur intervient directement dans la formation. Les apprenants pouvant discuter avec le formateur et ainsi lui poser des questions sur le contenu de la formation. L'inconvénient est qu'il faille prendre rendez-vous avez le formateur.

1.1.3 Bilan

Le *e-learning* commence donc à se développer, mais n'est pas encore extrêmement répandu. Dans le domaine du diagnostic *in vitro* il n'y a, à notre connaissance (du moins de manière rendue publique), aucune formation en lien avec les environnements virtuels. Notons également que ces formations *e-learning* ne sont que des formations complémentaires et qu'une formation traditionnelle en amont est indispensable. Dans le cas de la formation aux

instruments Stago, il s'agit de former à la manipulation d'un système. Il est donc nécessaire d'aller au delà du *e-learning* car il faut que l'apprenant puisse effectivement réaliser les tâches. C'est sur ce point que se trouve l'intérêt d'utiliser les environnements virtuels pour la formation.

Comme indiqué dans le chapitre `Introduction`, les environnements virtuels pour la formation permettent aux apprenants de se former sur un ordinateur où ils le souhaitent et quand ils le souhaitent. Ils peuvent répéter ces formations en environnements virtuels aussi souvent qu'ils le souhaitent et manipuler les instruments sans aucun risque de dégâts matériels ou humains. L'utilisation d'environnements virtuels pour la formation ainsi que de STI est donc un axe d'exploration intéressant pour Stago. Il existe plusieurs STI issus de différents domaines de recherche, il convient donc de faire un bilan des points intéressants de ces STI ainsi que de leurs limites (section 1.3).

Le choix d'utiliser la réalité virtuelle apporte de plus une touche d'originalité et d'innovation par rapport aux autres entreprises. En plus de l'environnement virtuel, nous voulons permettre au formateur de décrire sa propre situation d'apprentissage. Pour cela il nous faudra donc coupler notre environnement virtuel à un scénario pédagogique (section 1.2). La figure 1.2 illustre l'ensemble des entités que nous souhaitons mettre en interaction.

1.2 Scénario Pédagogique

Nous nous intéressons à l'activité du formateur qui souhaite concevoir et mettre en place des sessions d'apprentissage dont les activités se déroulent en environnement virtuel. Dans ce cadre, le rôle du formateur consiste à préparer et exploiter les interactions entre l'apprenant et le dispositif d'apprentissage [Henri et al., 2007]. Pour cela, il écrit un scénario pédagogique dont l'exécution doit permettre la construction des connaissances visées chez les apprenants.

1.2.1 Définitions

Dans sa définition originale, Koper [2001] considère qu'un scénario pédagogique est composé de :

- *Objectifs pédagogiques* : la description des objectifs pédagogiques peut s'effectuer sur deux niveaux. Le premier niveau consiste à décrire les objectifs de réalisation. Il s'agit des tâches que l'apprenant doit être capable de réaliser, en respectant des consignes, pour considérer l'exercice comme réussi. Le deuxième niveau concerne les objectifs de compétences d'apprentissage visées. Il s'agit alors d'exprimer les connaissances ou compétences que l'apprenant doit avoir acquises en réalisant l'exercice.
- *Prérequis pédagogiques* : de la même manière que pour les objectifs pédagogiques ils peuvent être vus selon deux niveaux. Le premier niveau consiste à décrire les objectifs de réalisation qui doivent être validés avant de réaliser le scénario. Le second niveau concerne les compétences d'apprentissage qui doivent être acquises avant de réaliser le scénario.
- *Activités pédagogiques* : là encore ces activités peuvent être appréhendées selon deux niveaux. Le premier niveau consiste en l'organisation d'activités de niveaux inférieures.

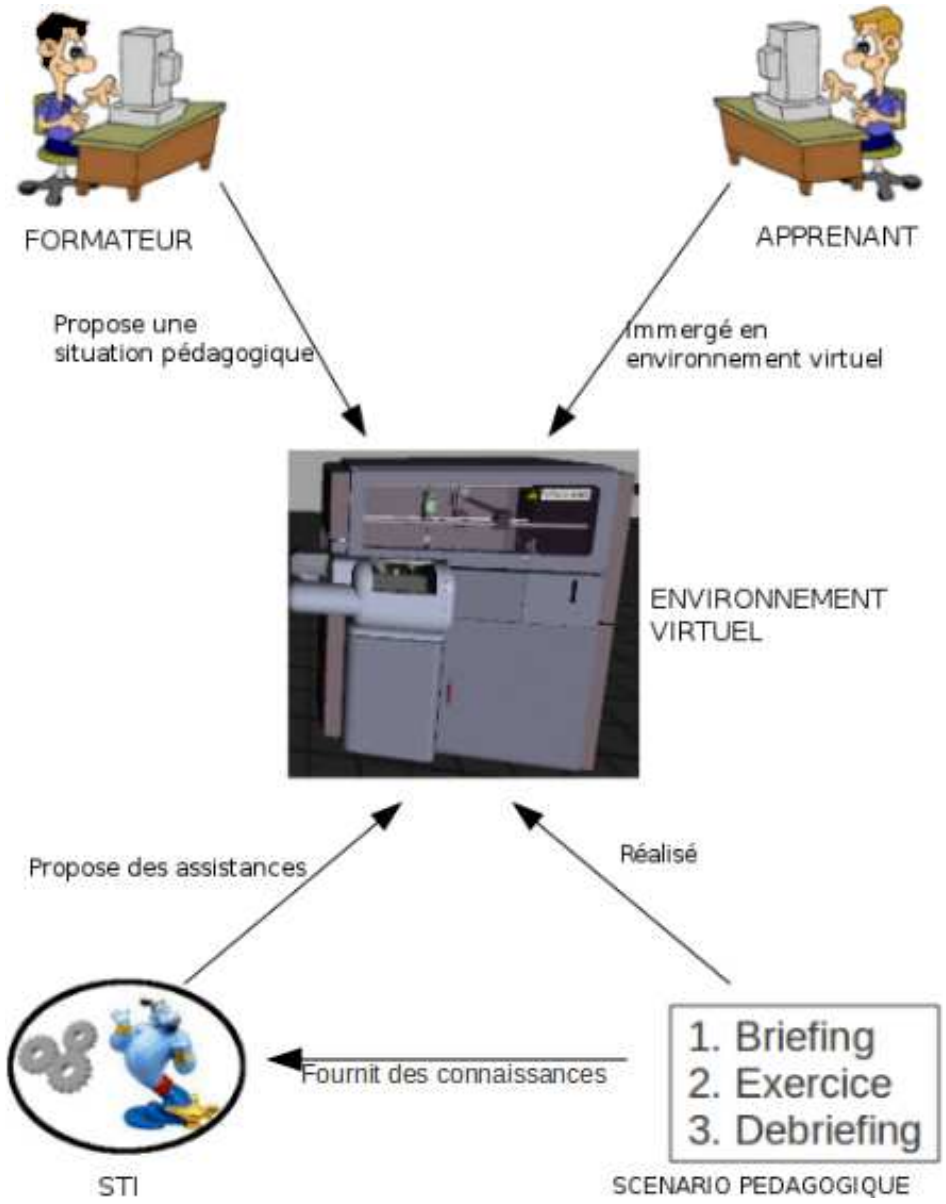

Figure 1.2 – Schéma général des interactions entre les entités de notre futur système

Il s'agit alors de décrire le cursus pédagogique. Le second niveau décrit plus concrètement une activité pédagogique dans le contexte de son environnement d'exécution (*e.g.* l'environnement virtuel). Il s'agit ici de décrire un exercice. L'activité pédagogique décrit l'enchaînement des actions des participants dans leur environnement (ensemble de ressources).
- *Organisation pédagogique* : il s'agit ici de décrire l'ensemble des rôles intervenant dans le scénario. Dans les outils existants et dans le modèle de Koper deux types de rôles sont définis : Apprenant et Tuteur. L'organisation définit le nombre d'intervenants de chaque rôle par exemple.
- *Environnement* : dans le cas d'un enseignement classique l'environnement désigne l'environnement réel dans lequel se déroule l'apprentissage (la salle de classe, le système réel à étudier), dans le cas des EIAH il s'agit de l'outil informatique utilisé, dans le cadre de notre thèse il s'agit de l'environnement virtuel. L'environnement désigne également l'ensemble des ressources pédagogiques utilisées.

Nous souhaitons utiliser la réalité virtuelle pour nos simulations, cela nous amène à identifier plusieurs critères nécessaires pour l'utilisation de scénarios pédagogiques. Tout d'abord il nous faut avoir un modèle basé sur les descriptions de Koper afin que notre STI (entre autre) puisse raisonner dessus. Il faut également que notre scénario pédagogique se présente de manière formelle et exécutable (pour la réalité virtuelle). Cela permet de tester le modèle, comme par exemple vérifier au cours de la simulation les *prérequis*. Nous souhaitons également que notre modèle de scénario pédagogique soit générique afin qu'il puisse être utilisé indépendamment de l'application. Finalement, nous devons obtenir des liens entre le scénario pédagogique et notre STI afin que ce dernier puisse interpréter et raisonner sur le scénario. Cela permettra d'adapter le raisonnement du STI à la stratégie pédagogique du formateur.

Nous allons donc tout d'abord présenter des modèles de scénarios pédagogiques issus de la littérature dans la section 1.2.2. À partir de cet état de l'art nous expliciterons notre choix de modèle de scénario pédagogique dans la section 1.2.3.

1.2.2 Limites de l'existant

Les modèles de scénarios pédagogiques peuvent être classés en deux catégories : les modèles de scénarios décontextualisés et ceux contextualisés (c'est-à-dire des modèles qui prennent en compte ou non le contexte d'exécution). Nous présenterons ici ces modèles ainsi que leurs avantages et leurs inconvénients en se basant sur les travaux de Marion [2010].

Les modèles de scénarios décontextualisés

Avant 2000, les scénarios pédagogiques se basaient essentiellement sur des approches documentalistes telle que le *Learning Object Metadata*[2] (LOM) qui est un schéma de description de ressources d'enseignement et d'apprentissage. LOM est surtout utilisé dans un contexte de production massive d'objets pédagogiques. Une autre approche est l'utilisation du *Sharable*

2. http ://www.lom-fr.fr

Content Object Reference Model[3] (SCORM) qui est une spécification de codage permettant de créer des objets pédagogiques structurés. Le modèle SCORM permet entre autre de construire des plate-formes *e-learning* et les contenus de formation. La norme SCORM garantit l'interopérabilité des contenus produits, c'est-à-dire qu'elle permet d'utiliser sur une autre plate-forme des composants d'enseignement développés sur une plate-forme ou un outil de génération de contenus.

A partir des années 2000, Koper [2001] a initié l'utilisation de langages de modélisation pédagogiques (*e.g.* EML). La principale caractéristique des EML est que contrairement aux approches documentalistes, les EML se concentrent sur la description des activités et non sur les ressources pédagogiques. EML se base sur un processus d'apprentissage appelé *unité d'apprentissage*, qui permet de décrire les activités que les participants doivent réaliser dans les environnements adaptés (*via* les rôles). La figure 1.3 présente une vue d'EML.

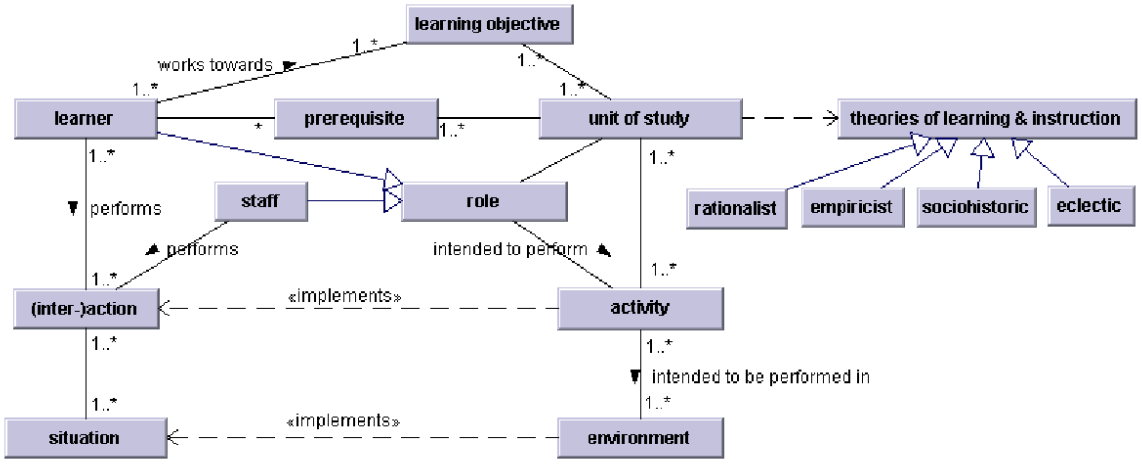

Figure 1.3 – Une vue du modèle EML, d'après Koper [2001]

Par la suite, en se basant sur EML, Koper et al. [2003] ont contribué à la norme IMS-Learning Design (IMS-LD). IMS-LD reprend la notion d'unité d'apprentissage comme élément de base de la description du processus d'apprentissage. En effet, dans IMS-LD, un scénario est considéré comme un enchaînement d'activités pédagogiques. Chacune de ces activités est décrite par un texte ou un ensemble de documents expliquant le but de l'activité, la tâche à réaliser, les consignes à respecter, etc. Afin de permettre à l'industrie de développer peu à peu l'infrastructure technique nécessaire, il existe 3 niveaux :

- Niveau A : contient le noyau de la conception pédagogique d'IMS (rôles, les activités élémentaires et les ressources) et leur coordination grâce aux éléments méthode, play, acte. Les activités d'apprentissage sont simplement ordonnées dans le temps, pour être exécutées par des apprenants, en utilisant les objets et/ou les services d'apprentissage.
- Niveau B : ajoute des conditions. Les conditions permettent de décider de l'évolution d'un scénario pédagogique à un moment donné. Par l'évaluation d'une expression on peut décider, en fonction de son résultat, quel parcours suivra le scénario.
- Niveau C : ajoute des notifications.

3. http ://www.scorm.com

Limites

Cependant, IMS-LD a ses limites, comme par exemple cette critique de Ferraris et al. [2005] sur le manque d'expressivité en ce qui concerne la description des interactions entre utilisateurs lors de tâches collaboratives. Martel et al. [2006] ont proposé leur propre modèle de scénario pédagogique : *Learning Desing Language* (LDL). Ce modèle constitue un enrichissement d'IMS-LD capable de prendre en compte le contexte social des activités.

Ceci retrace l'évolution des différents modèles de scénarios décontextualisés proposés dans la littérature. Si ces modèles peuvent produire des scénarios couvrant une large gamme d'applications, cela induit un manque de liens entre le scénario pédagogique et son environnement d'exécution [Allert, 2004]. Du fait de la description textuelle de ces modèles, le déroulement des activités pédagogiques est considéré du point de vue de la plate-forme exécutant le scénario comme une « boîte noire » dont on ne connaît que les entrées (ressources) et sorties (productions). Si ce type de description peut être suffisant pour la réalisation de tâches ne nécessitant pas le contrôle de l'activité des apprenants (*e.g.* remplir des questionnaires ou réaliser des calculs), cela ne convient pas pour décrire des activités pendant lesquelles les apprenants sont en forte interaction avec l'environnement [Guéraud, 2006], comme c'est le cas avec les EVAH. Pour compenser cela, de nouveaux modèles ont été proposés : les modèles de scénarios contextualisés.

Les modèles de scénarios contextualisés

Dans la littérature, nous pouvons trouver de nombreux modèles de scénarios pédagogiques contextualisés. Ceux-ci peuvent être classés en deux catégories :
- modèles orientés simulation : RIDES [Munro et al., 1997], SimQuest [Joolingen et de Jong, 2003], FORMID [Guéraud et al., 2004], SAM [Rosmalen, 1994],
- modèles orientés réalité virtuelle : FIACRE [Lourdeaux, 2001], GVT [Gerbaud et al., 2008], METISSE [El-Kechai et Desprès, 2007], Poseidon [Marion, 2010].

Dans un premier temps, nous allons détailler chacun de ces modèles afin de définir les points forts et les points faibles de chacun.

Modèles orientés simulation

RIDES est une application-auteur permettant de créer des simulations interactives pour l'apprentissage et est utilisée pour créer des simulations couvrant un large panel de sujets, incluant le fonctionnement des dispositifs médicaux, la nomenclature et la théorie de la mécanique orbitale ou encore le système circulatoire de l'homme. Un inconvénient de RIDES est que, si le scénario décrit bien la tâche à réaliser par l'apprenant dans l'environnement, il ne contient aucune information pédagogique concernant cette tâche (erreurs classiques, assistances préconisées, etc).

SimQuest est un système-auteur conçu pour créer des simulations embarquées dans un environnement d'apprentissage. SimQuest prend en charge le processus complet, de la

création du modèle de la simulation à la définition des interactions pédagogiques. Un des inconvénients de SimQuest porte sur le type de tâche que le modèle est capable de décrire : il restreint l'usage de SimQuest aux situations d'apprentissage par découverte (pas de contrôle de l'activité).

Le projet FORMID (FORMation Interactive à Distance) s'intéresse à la scénarisation pédagogique d'activités durant lesquelles les apprenants interagissent avec un Objet Pédagogique Interactif (OPI), tel qu'une simulation, un micromonde, etc. Le point fort de l'approche proposée par FORMID est de permettre la création de scénarios d'activités décrivant précisément la tâche à réaliser en interaction avec une simulation, sans que le modèle soit spécifique à un type particulier de simulations.

Le projet SAM (Simulation And Multimedia) a pour but de proposer un framework permettant la conception de simulations pédagogiques. SAM se présente comme une plate-forme permettant d'intégrer des simulations existantes ou développées pour l'occasion. Un des reproches que l'on peut formuler à l'encontre de SAM est que, à l'instar de FORMID la description de la tâche et des informations pédagogiques la concernant ne sont pas indépendantes.

`Modèles orientés réalité virtuelle`

Les travaux de thèse de Lourdeaux [2001] s'intéressent à la conception d'environnements virtuels pour la formation. Ces travaux ont abouti à la création d'un environnement virtuel pour la formation des conducteurs de TGV à l'intervention sur les voies ferrées à travers le projet FIACRE. La démarche de scénarisation proposée par Lourdeaux est intéressante car elle ne considère pas seulement la tâche à réaliser dans l'environnement mais permet également de décrire les spécificités pédagogiques de chaque tâche et la manière dont peuvent être traités les cas particuliers. Cependant, comme pour SAM et FORMID, la description de la tâche et des informations pédagogiques la concernant ne sont pas indépendantes.

GVT (Generic Virtual Training) est commercialisé par Nexter Training. L'objectif du projet GVT est de proposer un ensemble de logiciels permettant de couvrir le cycle de vie d'un EVAH, allant de sa conception à son exploitation par les apprenants et les formateurs. GVT se situe dans le cadre des EVAH destinés à l'apprentissage de procédures. Comme pour les modèles de FORMID ou FIACRE, la distinction n'est pas faite entre le scénario d'activité qui décrit les procédures à réaliser dans l'environnement, et le scénario pédagogique. De plus, à cause du contexte industriel dans lequel s'inscrit le projet, il ne peut être utilisé que pour l'apprentissage de procédures.

Le projet APLG (Atelier Pédagogique Logiciel Générique) [Lourdeaux et al., 2005] a pour objectif de proposer une bibliothèque logicielle pour le suivi de l'apprenant, la détection d'erreur et les feedbacks pédagogiques en EVAH. Dans ce cadre, les travaux de [El-Kechai et Després, 2007] ont conduit à la conception du langage de description de tâches METISSE (Modèle de dEscription de Tâches pour l'assIStance et de Suivi de l'apprEnant). Cette modélisation est intéressante pour plusieurs raisons :
– elle permet de décrire la tâche de manière indépendante de l'environnement virtuel ;
– elle permet de rendre compte des distances sémantiques des objets, et ainsi de classer les erreurs de l'apprenant (utilisation d'un objet similaire, d'un objet proche, etc) ;
– elle permet de décrire de manière générique les assistances pédagogiques sur des classes

d'objets, en termes de modification d'attributs (changement de couleur, d'échelle) ou de comportement (faire clignoter).

Les travaux de Marion se focalisent sur un modèle de scénario pédagogique nommé Poseidon, qui a pour objectif d'être directement réutilisable dans différents environnements. Pour cela il considère qu'un modèle de scénario pédagogique doit s'appuyer sur un langage de description d'environnement virtuel contenant tous les concepts nécessaires. Il propose donc un langage qui permet l'expression d'un ensemble de connaissances d'un domaine, qui permet également la description d'entités virtuelles et enfin qui permet de décrire l'activité humaine dans l'environnement. Poseidon couvre différents points tels que : les objectifs pédagogiques, les prérequis, les activités pédagogiques, les organisations pédagogiques et les environnements.

`Limites`

Si la plupart des simulations pédagogiques (et *a fortiori* des EVAH) décrivent les informations relatives à l'activité attendue des apprenants dans l'environnement, peu d'entre elles intègrent des informations pédagogiques relatives à cette activité (situations à repérer, assistances à fournir, etc.). Quelques simulations cependant explicitent ces informations [Lourdeaux, 2001; Gerbaud et al., 2008], mais le font généralement de manière spécifique à une tâche, un domaine ou une stratégie d'apprentissage. De ce fait, le modèle de description des activités pédagogiques ne peut être réutilisé pour une autre application. Poseidon quant à lui vient gommer cette limitation.

1.2.3 Choix du modèle de scénario

Plusieurs modèles permettent au concepteur de manipuler les connaissances entrant en jeu dans l'apprentissage *via* l'utilisation d'un modèle du domaine (*e.g.* Sam, SimQuest, Metisse, Poseidon).

Parmi ces modèles, nous avons choisi de nous baser sur Poseidon (modèle générique permettant la description de scénarios pédagogiques applicables à des simulations de réalité virtuelle). En section 1.2.1 nous avions indiqué que nous souhaitions un modèle de scénario pédagogique basé sur les notions décrites par Koper. Poseidon se base justement sur ces notions.

Une des avancées majeures de Poseidon est qu'il rend possible la description d'activités pédagogiques permettant le contrôle de l'activité d'apprenants en interaction au sein d'un environnement virtuel de manière générique dans le sens où il n'est pas spécifique à un domaine d'étude, à une stratégie pédagogique ou à une plate-forme d'exécution. Pour atteindre ce niveau de généricité, Poseidon s'appuie sur une représentation abstraite des environnements virtuels, et cette représentation est fournie par Mascaret. Nous pouvons donc considérer que Poseidon est un modèle de scénario pédagogique générique.

Le dernier critère important était de créer des liens entre notre scénario pédagogique et notre STI. En effet, pour décrire le scénario pédagogique, celui-ci devra être écrit par un formateur et constituera une base de connaissances pour le STI qui sera relative à un exercice particuler.

Lors de l'exécution du scénario, ces connaissances peuvent être ou non prises en compte par l'entité jouant le rôle du formateur (humain ou agent pédagogique intelligent). Ce critère n'étant pas présent dans Poseidon, nous souhaitons donc créer un lien entre notre scénario pédagogique et notre STI qui pourra utiliser les connaissances du scénario pédagogique.

1.3 Systèmes Tutoriels Intelligents

Dans le chapitre `Introduction`, nous avons énuméré les caractéristiques qui nous paraissaient essentielles pour un STI : la généricité, la modularité, l'individualisation, l'adaptativité et le scénario pédagogique. Il nous faut donc maintenant expliciter ces choix. L'état de l'art proposé ici nous permettra de faire le point sur l'existant au regard de ces caractéristiques. Nous tirerons alors un bilan sur les possibilités d'utilisation des STI existants.

1.3.1 Critères

Généricité

Tel que mentionné dans le chapitre précédent, la généricité est une caractéristique importante et nécessaire à notre STI. La méthodologie actuelle de l'ingénierie des STI n'est pas idéale. En effet, chaque application est développée indépendamment, l'expertise formateur n'est pas considérée comme une donnée et est codée de manière implicite dans les applications. Il existe également très peu de réutilisation des composants des tuteurs [Rodrigues et al., 2005]. Pour Imbert et al. [2007], une grosse partie du travail à effectuer pour développer un nouveau système de ce type est très souvent similaire à ce qui existe déjà. De plus, dans les Environnements Informatiques pour l'Apprentissage Humain (EIAH), certains composants sont difficilement réutilisables d'une simulation sur l'autre. C'est le cas avec HAL [Lourdeaux et al., 2001], dans lequel les assistances du modèle pédagogique doivent être reformulées à chaque exercice. Par conséquent il nous paraît essentiel de rendre notre STI le plus générique possible.

Modularité

La stratégie pédagogique peut varier sensiblement suivant la façon dont le formateur souhaite intervenir, quand intervenir et comment intervenir [Wenger, 1987; Lourdeaux, 2001]. Le formateur pourrait vouloir ajouter un élément perturbateur [Aimeur et al., 2000] où bien encore un élément compagnon [Pesty et Webber, 2004]. De la même manière, certains comportements du STI pourraient être remplacés par une intervention humaine. Afin d'obtenir cette liberté d'adapter aisément les stratégies pédagogiques en ajoutant ou remplaçant des comportements, il est nécessaire d'avoir une certaine modularité des comportements du STI. La modularité permet par la même occasion d'augmenter la généricité du STI.

Individualisation

Afin d'obtenir une meilleure efficacité, nous souhaitons que notre STI propose des assistances adaptées à chaque apprenant. Par conséquent, nous devrons travailler sur l'individualisation du STI. Les STI classiques sont composés d'un modèle de l'apprenant qui représente les connaissances de l'apprenant sur le problème [Leman et al., 1996; Py, 1998; Webber et al., 2002]. Ce modèle doit contenir une représentation du profil de l'apprenant, établi et mis à jour soit par un dispositif hors ligne (questionnaire par exemple), soit directement à partir des interactions que l'apprenant opère avec son environnement [Buche et Querrec, 2011]. La modélisation de l'apprenant est un problème réputé difficile [Bruillard, 1997]. Par conséquent, notre STI doit avoir un modèle de l'apprenant afin de permettre une individualisation des assistances.

Adaptativité

Du point de vue du formateur, les assistances proposées pourraient ne pas être idéales. Nous souhaitons donc que notre STI puisse modifier ses comportements en fonction de ce qu'il se passe dans la simulation. Le processus de raisonnement du STI doit donc s'auto-adapter dans le but de prendre en compte l'expérience passée. Par conséquent un de nos objectifs est que notre STI, grâce à l'expérience passée, puisse automatiquement suggérer les assistances appropriées en tenant compte à la fois du contexte de l'apprenant et celui de la simulation : le système deviendrait donc adaptatif [Buche et al., 2010].

Lien avec un scénario pédagogique

Un objectif majeur est de permettre au formateur de personnaliser chaque exercice. En effet, le formateur pourrait avoir la possibilité d'écrire un scénario pédagogique. Ce dernier serait une base de connaissances connectée à un exercice particulier. Grâce aux environnements virtuels de formation, nous sommes passés d'un schéma où le formateur expose un cours magistral (formation traditionnelle), à un schéma où la présentation des savoirs est laissée au dispositif alors que la tâche du formateur est de préparer et d'exploiter les interactions entre l'apprenant et le dispositif d'apprentissage [Henri et al., 2007]. Les différentes propositions de scénarios pédagogiques ont été présentées dans la section 1.2.

1.3.2 Limites de l'existant

Durant les dernières années, plusieurs recherches ont été effectuées dans le domaine des STI, et la plupart d'entre elles sont axées sur deux ou trois de ces caractéristiques (Table 1.1).
Plusieurs projets qui ont pour objectif d'invidualiser la simulation pour chaque apprenant. Cela peut être effectué par exemple à travers des agents émotion [Ailiya et al., 2010; Zakharov et al., 2008; Ortiz et al., 2009; Peng, 2006] ou en utilisant la classification de Hollnagel [El-Kechai, 2007].
Durant les cinq dernières années, le nombre de STI utilisant la technologie des systèmes

multi-agents se sont multipliés [Pesty et Webber, 2004; De Antonio et al., 2005; Li et al., 2010b,a], et de ce fait ont basé leur approche sur l'aspect modulaire.
D'autres travaux visaient à développer l'aspect générique des STI [Sanchez et Imbert, 2007; Shi et Lu, 2006].
Marion et al. [2009] et Sorensen et Ramachandran [2007] ont proposé la possibilité de créer et de modifier les scénarios, alors que d'autres se sont focalisés sur l'aspect adaptatif des STI [Dos Santos et Osorio, 2004; Fricoteaux et al., 2010].
Malheureusement, aucun de ces STI ne comporte les cinq caractéristiques définies précédemment.

	Généricité	Modularité	Individualisation	Scénario	Adaptativité
[Ailiya et al., 2010]	✗	✗	✓	✗	✗
[El-Kechai, 2007] (METISSE)	✗	✗	✓	✗	✗
[Li et al., 2010b; De Antonio et al., 2005]	✗	✓	✓	✗	✗
[Sanchez et Imbert, 2007] (MAEVIF)	✓	✓	✓	✗	✗
[Shi et Lu, 2006]	✓	✗	✓	✗	✗
[Sorensen et Ramachandran, 2007] (SimCore)	✓	✗	✗	✓	✗
[Marion et al., 2009] (Poseidon)	✗	✗	✗	✓	✗
[Dos Santos et Osorio, 2004]	✗	✗	✓	✗	✓
[Buche et Querrec, 2011] (Pegase)	✓	✓ / ✗	✓ / ✗	✗	✓

Table 1.1 – Comparaison des STI actuels

1.3.3 Choix du modèle de Système Tutoriel Intelligent

Dans notre équipe de recherche, nous avons développé un STI, appelé Pegase [Buche et Querrec, 2011], qui est basé sur les quatre modèles classiques (expert, apprenant, pédagogique et interface) [Woolf, 1992]. Pegase est générique dans le sens où il intègre une caractérisation des erreurs et une définition du modèle pédagogique utilisable indépendamment de l'exercice à réaliser. Enfin, Pegase est adaptable et cela est rendu possible par l'auto-modification du modèle pédagogique. Pegase est donc le STI le plus complet en se basant sur ces cinq caractéristiques.

Faiblesses de Pegase

Dans de la section 1.3.2, nous avons vu que Pegase présentait un manque : le lien avec un scénario pédagogique ; ainsi que des faiblesses sur deux autres caractéristiques : la modularité et l'individualisation.

Tout d'abord, la modularité nécessite d'être améliorée. En effet, dans le modèle de Pegase, le comportement du STI est « écrit en dur », il est donc très difficile de modifier ce comportement. Si on souhaite modifier un comportement il est donc nécessaire de le réorganiser complètement.

Dans Pegase, l'individualisation est relativement simpliste. En effet, le STI raisonne sur les actions de l'apprenant et certaines caractéristiques connues (*e.g.* le niveau de l'apprenant :

débutant, confirmé, expert...), mais ces connaissances sont relativement simples et cela mérite d'être approfondi.

Pour terminer, Pegase ne comporte aucun lien avec le scénario pédagogique. Il serait donc intéressant de travailler sur un modèle de scénario pédagogique afin de pouvoir le coupler à notre STI.

Perspectives d'amélioration

Afin d'améliorer la modularité de Pegase, nous proposons l'utilisation d'un système multi-agents (base de connaissances, communication). Comme indiqué précédemment, cela permettra de modifier ou ajouter facilement un comportement d'agent ainsi que les connaissances associées (*e.g.* agent perturbateur).

De la même manière afin d'améliorer l'individualisation, il serait intéressant d'incorporer des connaissances supplémentaires, telles que des informations sur les capacités cognitives de l'apprenant (*e.g.* ses capacités visuelles en environnement 3D).

Finalement, le fait de créer un lien entre le scénario pédagogique et notre STI permettra à ce dernier de raisonner sur les connaissances issues du scénario pédagogique (*e.g.* le STI aura connaissance d'un point critique de la tâche, identifié par le formateur).

Afin de pouvoir raisonner, notre STI nécessite d'avoir accès à des connaissances (*e.g.* celles du scénario pédagogique ou de l'apprenant). Généralement ces connaissances sont les informations contenues dans l'environnement virtuel. Dans la section 1.2 nous avons montré l'intérêt d'avoir un scénario pédagogique. Il serait donc intéressant que notre STI puisse également obtenir un accès aux connaissances du scénario pédagogique afin de raisonner dessus. Nous souhaitons donc que notre STI soit capable de raisonner sur différentes connaissances telles que l'environnement virtuel et le scénario pédagogique, voir de nouvelles connaissances ultérieurement. Il nous faut donc déterminer comment exprimer ces connaissances afin que, malgré leurs origines différentes, elles puissent être utilisées par le STI.

1.4 Représentation des connaissances

Afin de pouvoir raisonner correctement, un STI doit pouvoir accéder à diverses connaissances. La base de connaissances propose donc de regrouper ces connaissances spécifiques à un domaine spécialisé. La recherche sur les modèles de bases de connaissances d'agent est donc nécessaire, c'est pourquoi beaucoup d'études y sont consacrées.

1.4.1 Classement des modèles de bases de connaissances

Plusieurs modèles sont proposés dans la littérature, nous avons classé les différentes structures de bases de connaissances sous trois catégories distinctes. Tout d'abord nous avons regroupé

les structures utilisant les langages de programmation agent : APL (Agent Programming Language) dans la section 1.4.1.1. Nous avons également regroupé celles basées sur les communautés virtuelles de connaissances dans la section 1.4.1.2. Finalement, il nous paraissait difficile de classer les autres systèmes qui nous semblaient intéressants. En effet, il existe une pléthore de modèles différents avec chacun leurs qualités et défauts, c'est pourquoi cette catégorie, décrite dans la section 1.4.1.3, regroupe tous les modèles particuliers.

1.4.1.1 APL

Dans la littérature, il existe beaucoup de systèmes utilisant les langages de programmation agent. Par exemple en 2003, Dastani et al. [2003] ont proposé le langage 3-APL. Comme les autres APL, ce langage fournit des structures de données et des outils pour manipuler ces données. Il est adapté pour manipuler des agents cognitifs (qui contiennent des croyances, désirs, intentions et des règles de raisonnement), plus connus sous le nom d'agents BDI (Belief, Desire, Intention). Ce langage est d'après l'auteur, compatible avec les normes proposées par *FIPA*[4] (Foundation for Intelligent Physical Agents) telles que La norme *FIPA-ACL* (Agent Communication Language) pour la communication entre les agents. Une extension du langage 3-APL a été proposée en 2005 [Winkelhagen et al., 2005] : des opérateurs modaux des croyances ont été ajoutés, permettant de donner aux agents une signification pour raisonner explicitement sur leurs croyances et celles des autres agents. Ce nouveau langage de programmation fournit des structures de données telles que les croyances, les buts, la planification et les règles de raisonnement. Un nouvel APL a également été proposé en 2008 [Katasonov et Terziyan, 2008] : le langage S-APL. Ce langage permet l'échange de comportements via une plate-forme middleware, et est basé sur RDF (Resource Description Framework). RDF est un modèle de graphe destiné à décrire de façon formelle les ressources Web et leurs métadonnées, de façon à permettre le traitement automatique de telles descriptions.

Ce type de proposition met l'accent sur le langage agent pour manipuler les connaissances, mais cela n'apporte que peu de choses sur la façon de gérer les connaissances, d'un point de vue structurel. C'est pourquoi il existe dans la littérature d'autres types de recherches qui sont plus axés sur le modèle et non le langage, comme par exemple les communautés virtuelles de connaissances.

1.4.1.2 Communautés Virtuelles de Connaissances

Il existe d'autres travaux plus axés sur la structure des bases de connaissance, tels que les récentes recherches sur les communautés virtuelles de connaissances. En effet, Maret et Calmet [2009] ont proposé un modèle général qui étend l'abstraction des agents, de telle manière qu'ils deviennent des acteurs au sein de la communauté des connaissances. Plus précisément, les agents sont des membres de la communauté virtuelle de partage des connaissances. Les connaissances des agents sont représentées par une ontologie et décrites via le Web Ontology Language (*OWL*). *OWL* est un langage de représentation des connaissances construit sur le modèle de données de RDF. Il fournit les moyens pour définir des ontologies structurées. Le

4. http ://www.fipa.org

système proposé par Maret est basé sur deux API : JENA et Protégé-OWL. Ce système de communautés virtuelles de connaissances permet de partager de meilleures connaissances car la taille des communautés est plus petite que celle des bases de connaissances habituelles. Notons que l'ontologie est l'ensemble structuré des termes et concepts représentant le sens d'un champ d'informations. Par la suite, d'autres travaux récents ont contribué à développer cet axe de recherche. Par exemple Wang et Yu [2005] ont proposé des travaux sur l'échange des connaissances de communautés virtuelles dynamiques, basés sur l'ontologie.

Les communautés virtuelles de connaissances nous paraissent très intéressantes et connaissent une recrudescence du nombre de recherches dans les dernières années, mais il existe également d'autres propositions bien différentes.

1.4.1.3 Autres propositions

La recherche dans le domaine des bases de connaissances étant très vaste, il existe beaucoup de modèles très intéressants, mais très variés.

`OKBC`
En 2008 Chaudhri et Fikes [1998] ont proposé une OKBC (Open Knowledge Base Connectivity) : c'est une API pour les systèmes de représentations des connaissances, ayant pour but de faciliter la réutilisabilité des outils des bases de connaissances. Dans le protocole OKBC, les spécifications des systèmes de représentations des connaissances ont trois composants :
– un modèle des connaissances,
– une collection d'opérations pour accéder aux systèmes de représentation des connaissances : KRS (Knowledge Representation Systems),
– une collection de comportements.
L'objectif de OKBC est de servir d'interface à de multiples KRS différentes.

`Système AIM`
En 2000, Raphael et Deloach [2000] ont proposé leur système nommé : AIM (Agent knowledge Interchange Mechanism). Ce système est proche de AgentTool (système adapté à la méthodologie de l'ingénierie des systèmes multi-agents) et est utilisé pour stocker, retrouver et filtrer des connaissances du domaine qui sont persistantes, réutilisables et fiables. Il y a une représentation des connaissances et un langage d'échange appelé MAML (Multiagent Markup Language). Un système de stockage des connaissances appelé RAMS (the Agent oriented Random-Access Meta-Structure) est proposé. RAMS permet d'organiser les connaissances dans des librairies.

`Systèmes basés sur les croyances`
Nguyen [2004] a proposé le Modal Logic Programming System MProlog. Ce système est basé sur des logiques modales permettant de manipuler des croyances. Braubach et al. [2005] ont ensuite proposé Jadex, qui est un système d'agents BDI. Il permet l'intégration d'un agent middleware avec un moteur de raisonnement pour combiner leurs avantages. Il est basé sur Jade et contient une base de croyance.
Portillo-Rodriguez et al. [2007] ont proposé une architecture multi-agents à trois niveaux afin de prendre en compte la réputation et la fiabilité des agents au sein des communautés ou les connaissances sont échangées. Les trois niveaux sont : `Reactive, Deliberative, Social`.

Ce système, proche des travaux de Barber et Kim [2004], est basé sur les concepts de croyance (agents qui évaluent les informations reçues) et de réputation.

`SAIBA`
Récemment, le projet SAIBA [Vilhjalmsson, 2007] a été présenté, et son but est d'unifier un framework multimodal de génération de comportements pour les agents conversationnels animés. Il utilise le BML (Behavior Markup Language), et toutes ses évolutions y sont présentées. Il utilise également le langage FML qui fait le lien entre 2 niveaux dans SAIBA : de *Intent planning* vers *Behavior planning*, alors que BML fait le lien entre *Behavior planning* et *Behavior Realization*.
Une autre forme d'approche a été proposée par De Sevin et al. [2009] en 2009 : le projet *GRETA* qui est une plate-forme d'agent conversationnel expressif et interactif. Leur agent est une implémentation complète de l'architecture standard SAIBA pour les modules de planification et de génération du comportement, ainsi qu'une implémentation partielle du module de planification des intentions (basé sur les deux langages de balises : BML et FML).

`Systèmes divers`
En 2006 Schiemann [2006] a proposé une solution basée sur la combinaison de *OWL-DL* et des messages *FIPA-ACL*. Les théorèmes pour raisonner sur l'application sont stockés dans leur base de connaissances.

Par la suite, Sajja [2008] a proposé un framework pour les découvertes, l'utilisation et la gestion des connaissances, pour donner un accès à la base de données du domaine, utilisant une approche système multi-agents. Il suggère également une amélioration de KQML. KQML est un langage de communication et un protocole de haut niveau pour l'échange de l'information, orienté messages et indépendant de la syntaxe du contenu et de l'ontologie applicable. Toutes les bases de données suivent une modélisation de topologie déterminée, telle que la RDBMS (Relational Database Management Systems).

Bilan

Ces différents travaux issus de la littérature nous ont permis d'avoir une vue globale des recherches faites sur les bases de connaissances. Il nous faut maintenant trouver le modèle (ou des propositions issues des modèles) qui s'adapteront le mieux à notre modèle de STI.

1.4.2 Choix du modèle de base de connaissances

Nous pouvons voir que différents systèmes et modèles de bases de connaissances existent dans la littérature. Les travaux en système multi-agents portent essentiellement sur les comportements plutôt que sur la structure des connaissances. Cependant, il apparait que l'utilisation d'ontologies et plus précisément le langage de description d'ontologie *OWL*, sont très fréquentes.

Intérêt de *OWL*

OWL permet, grâce à sa sémantique formelle basée sur une fondation logique, de définir des associations plus complexes des ressources ainsi que les propriétés de leurs classes respectives. *OWL* est une amélioration de RDF (il est plus expressif et apporte une meilleure intégration, une évolution, un partage et une inférence plus facile des ontologies) qui était déjà un standard bien répandu. Dans les modèles cités précédemment, nous pouvons voir que *OWL* est souvent couplé avec le protocole de communication agent *FIPA-ACL*. Rappelons que la norme *FIPA-ACL* est un standard de communication agent très utilisé. Il est donc très probable que nous utilisions également un langage d'ontologie pour décrire les connaissances.

Dans le chapitre `Introduction`, nous avons cité Mascaret qui est un méta-modèle d'environnement et d'agent permettant de décrire de manière formelle un environnement virtuel et les agents évoluant dans cet environnement. L'utilisation d'ontologies pour informer (donner de la semantique) aux entités de l'environnement et donc servir de connaissances aux agents semble communément acceptée. C'est justement l'objectif de Mascaret de fournir cette sémantique aux entités. L'utilisation de Mascaret est largement compatible avec les langages d'ontologie de type *OWL*. Cependant Mascaret propose des avantages dans le cadre de la réalité virtuelle et nous proposons donc de l'utiliser dans nos travaux.

Choix de MASCARET

Mascaret ayant été développé au laboratoire, nous souhaitions nous baser sur ce méta-modèle pour décrire notre environnement. Ainsi, les modèles exprimés en Mascaret pourront être utilisés comme une base de connaissances. Plus précisément, une base de connaissances (`Mascaret::KnowledgeBase`) référencera un modèle de l'environnement (Figure 1.4), comme par exemple le modèle de scénario pédagogique. En effet, en faisant ce postulat, toutes les connaissances sur le scénario pédagogique (rôles, pédagogie) pourront appartenir à la base de connaissances d'un agent, lequel pourra exploiter ces connaissances au cours de l'exécution de son comportement.

Afin de mieux comprendre l'intérêt de se baser sur le méta-modèle Mascaret pour décrire nos connaissances (environnements, rôles, agents) ainsi que notre STI qui exploitera ces connaissances, nous devons dans un premier temps détailler Mascaret.

1.5 MASCARET

1.5.1 Introduction

L'objectif de Mascaret est de fournir un haut niveau d'abstraction dans la conception d'applications de réalité virtuelle. Il en résulte que les modèles de conception d'une application spécifique de réalité virtuelle deviennent des données pour ce méta-modèle. Par conséquent, Mascaret fournit un langage qui permet à un expert du domaine de définir aussi bien

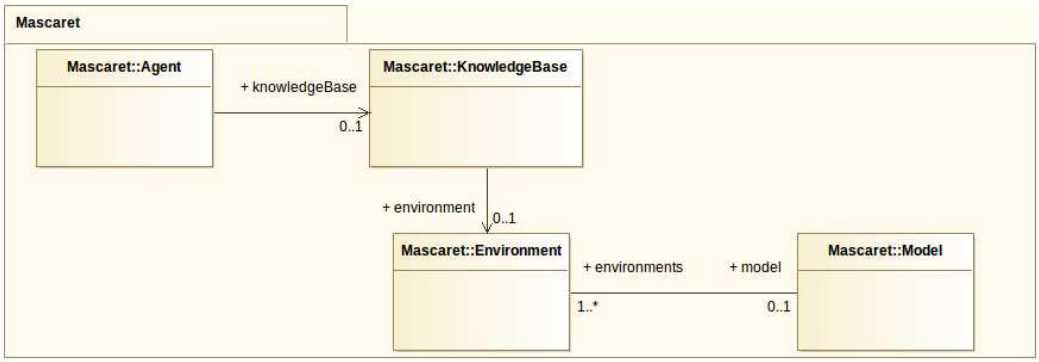

FIGURE 1.4 – Diagramme de classe des bases de connaissances d'un agent MASCARET

l'environnement que les activités qui sont exécutées dans cet environnement. MASCARET fournit une sémantique opérationnelle à chaque concept du langage, ce qui permet de créer automatiquement la simulation dans une application de réalité virtuelle et d'être vue comme une base de connaissances des agents qui exécutent les activités dans l'environnement.

MASCARET est donc un méta-modèle pour décrire les environnements virtuels et les agents qui évoluent dans cet environnement. Ce méta-modèle se base sur UML pour décrire la structure de l'environnement (entités, positions,...), les entités et les comportements d'agents. UML a déjà été utilisé par des modèles d'agents pour décrire les activités des agents [Bauer et al., 2001], mais la contribution majeure de MASCARET est le lien fort entre la conception de l'environnement et la conception des activités des agents. Un modèle en MASCARET peut ensuite être exporté en XMI (format normalisé par l'OMG [5]) puis interprété afin de générer les concepts dans la simulation (les entités, les rôles, les agents,...).

1.5.2 Le méta-modèle

MASCARET est un méta-modèle permettant de décrire un environnement virtuel, non pas en termes d'espaces géométriques, mais en fournissant la sémantique nécessaire aux agents artificiels ou aux humains pour construire leur propre représentation de l'environnement et pour agir ensemble afin d'atteindre leurs objectifs. Le méta-modèle MASCARET est basé sur UML et permet la construction de modèles du domaine à partir des environnements virtuels et des environnements virtuels *concrets* correspondant. Cependant, le méta-modèle UML ne nous permet pas de définir les concepts spécifiques de la réalité virtuelle. MASCARET propose une extension d'UML afin de représenter ces concepts.

Les agents ont besoin de savoir quels objets constituent l'environnement virtuel, comment y accéder, leurs propriétés, leurs comportements ainsi que la manière d'interagir avec eux. Trois types de connaissances peuvent être exprimés en utilisant MASCARET :

– Les concepts du domaine. Cela implique la description sémantique des concepts relatifs au domaine d'activité concerné.

5. http ://www.omg.org/spec/UML/

– La possibilité de structuration et d'interaction avec l'environnement. Ces concepts ressemblent à ceux suggérés dans les *smart objects* [Kallmann et Thalmann, 1998] qui réifient les propriétés requises pour les interactions.
– Les comportements des entités. Dans le cadre d'un environnement virtuel, les réactions de l'environnement à des actions de l'utilisateur doivent être simulées. Le comportement des entités et son exécution constituent également un élément de connaissance de l'agent.

Toutes les applications conçues par Mascaret suivent le processus illustré dans la figure 1.5.

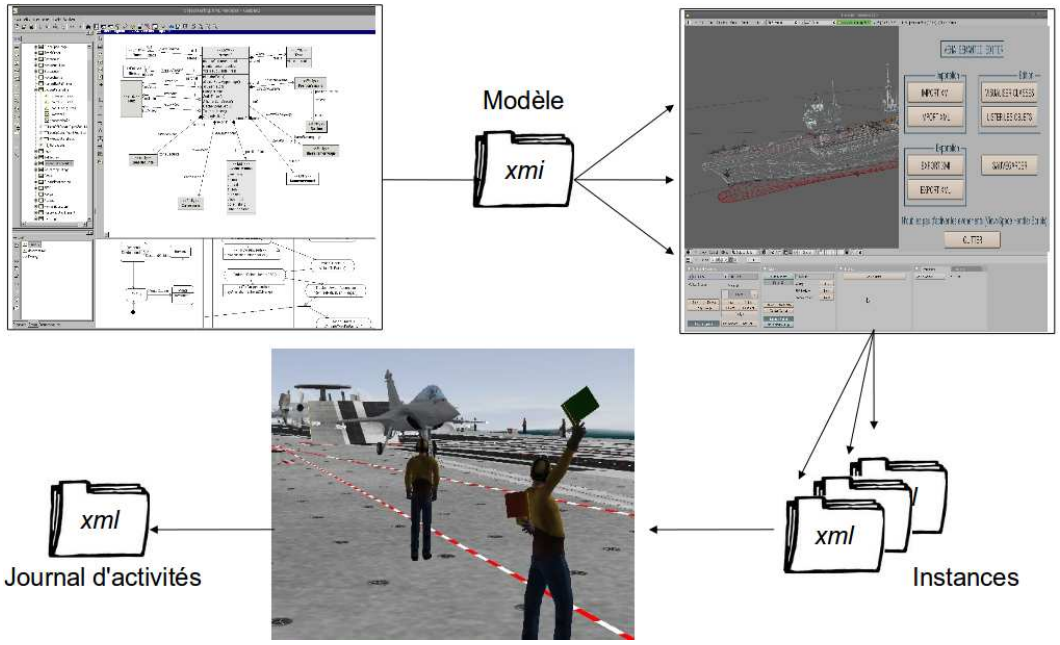

Figure 1.5 – Processus pour développer des applications utilisant Mascaret

Tout d'abord, l'expert du domaine définit le modèle de l'environnement virtuel (modèle M1) sous la forme de diagrammes Uml–Mascaret, exportés au format Xmi. Il doit décrire les modèles de classes et les modèles de comportements (machines à états et activités) et les activités humaines utilisant la collaboration Uml et les diagrammes d'activités. Cette étape est effectuée en utilisant un modeleur Uml.

Deuxièmement, les concepteurs 3D doivent construire des objets géométriques. Cela signifie la construction de formes et la définition de géométries (points informés, des surfaces et des volumes d'interaction) à l'aide d'un modeleur 3D classique. Un plugin Mascaret est associé au modeleur 3D afin de se référer au modèle Uml (fichier Xmi) et d'ajouter de la sémantique aux objets géométriques qui sont ensuite définis comme des instances du modèle du domaine (modèle M0). Plusieurs environnements virtuels peuvent être construits sur la base du même modèle M1 (figure 1.6).

Troisièmement, l'informaticien doit coder les éventuels *opaque behaviors* pour des comportements spécifiques non-introspectables.

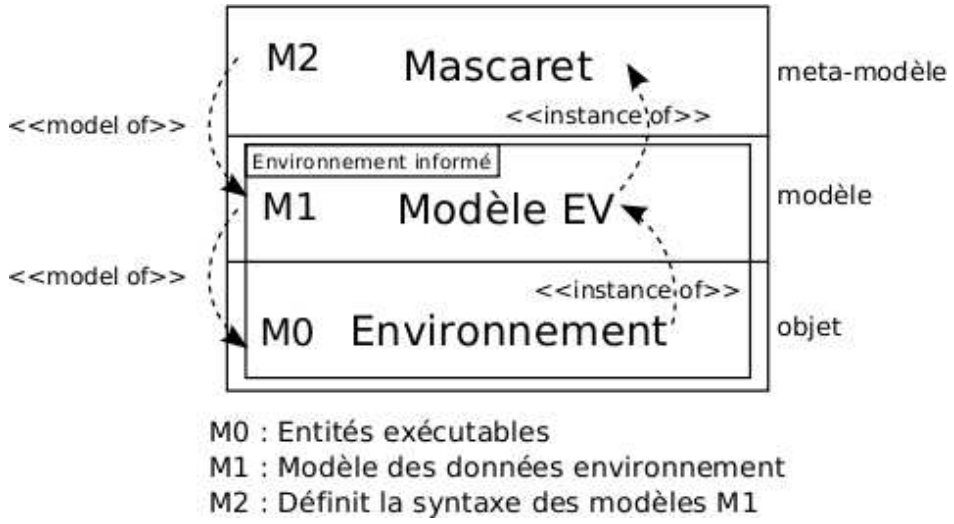

FIGURE 1.6 – Les trois niveaux de modélisation de MASCARET

Finalement, l'utilisateur doit démarrer la plate-forme de simulation : chargement des modèles du domaine (M1) et des environnements spécifiques (M0), activation des dispositifs d'interaction et d'immersion.

1.5.3 Le méta-modèle agent

MASCARET utilise les systèmes multi-agents pour simuler les activités humaines qui sont fortement contextualisées par l'environnement. Il est donc nécessaire d'utiliser le même langage pour décrire les activités que celui utilisé pour décrire l'environnement, de telle façon que ces activités simulées puissent manipuler l'environnement virtuel.

Plusieurs modèles d'agents ou méta-modèles d'agents ont été proposés dans la littérature. Ces modèles peuvent être basés sur le modèle UML [Bauer et al., 2001] ou peuvent proposer un modèle de comportement d'agent qui interprète automatiquement les connaissances exprimées en UML [Huget et Odell, 2004; Torres DaSilva et al., 2004; Ehrler et Cranefield, 2004]. Cependant, ces modèles ne couvrent qu'une partie des types de comportements qui nous intéressent, et surtout ne prennent pas en compte la modélisation de l'environnement qui gère ces comportements.

De plus, *FIPA* propose des modèles qui prétendent être un standard pour la modélisation agent. Le modèle MASCARET est donc basé sur cette norme. Par conséquent, MASCARET ne propose pas un modèle d'agent totalement novateur, mais une opérationnalisation des concepts qui existent dans d'autres modèles, pour des applications de réalité virtuelle. Les concepts impliqués sont : l'agent, ses actions ou ses comportements, ses moyens de communication et les organisations.

Agents et comportements

Le modèle d'agent que Mascaret propose se base sur le standard *FIPA* et son implémentation s'inspire de *Jade*. Un agent exécute ses comportements et peut communiquer avec d'autres agents par le biais de messages. Le comportement appelle la méthode `action()` tant qu'une condition n'est pas remplie. Pour aider à la conception d'un comportement, *Jade* fournit `OneShotBehavior` qui est exécutée une fois et `CyclicBehavior` qui boucle indéfiniment. L'agent effectue ensuite un ensemble d'activités qui sont organisées en séquence.

L'exécution du comportement (appel de la méthode `action()`) est gérée par l'ordonnanceur proposé par Mascaret. L'utilisateur fournit alors de nouveaux comportements en dérivant `OneShotBehavior` ou `CyclicBehavior` afin de surcharger la méthode `action()`. L'exécution du comportement `BehaviorExecution` est défini dans le méta-modèle et peut donc également être une connaissance explicite (démarrer, résultat ...) qui peut être utilisée par les agents.

Communication

Les agents utilisent des messages pour communiquer entre eux. Nous utilisons par conséquent le modèle proposé par *FIPA-ACL*. Un message est représenté par une performative. Le modèle *ACL* propose 23 performatives. Les messages sont exprimés dans un langage et couvrent une ontologie. Plusieurs langages existent à cette fin, mais nous utilisons celui proposé par *FIPA* : *FIPA-SL*. Chaque agent a un comportement de communication automatique. Ce comportement de communication est un `CyclicBehavior` qui réagit à chaque nouveau message. Le but de ce comportement est d'analyser automatiquement le contenu du message en fonction des performatives. Dans le langage *FIPA-SL*, Mascaret gère tout ce qui se rapporte à la réalisation d'une action. Ainsi, il est possible pour un agent de demander l'exécution d'une action à un autre agent. Toutes ces fonctions sont possibles dans le cadre d'un comportement d'agent automatique et générique, grâce au fait que toutes les informations sont explicites dans la simulation.

Organisation

Les notions de collaboration ou d'interaction au sein de la structure organisationnelle entre les participants sont importantes. En effet, ces notions gèrent la simulation de l'activité humaine avec des systèmes multi-agents collaboratifs. L'organisation peut être une description *a priori* ou une inférence *a posteriori*. Elle peut être définie selon des règles strictes de comportements ou issues d'agents. Dans le cadre de la simulation des activités humaines dans un environnement virtuel, le modélisateur décrit explicitement la structure de l'organisation.

Plusieurs modèles organisationnels existent [Montealegre Vazquez et Lopez y Lopez, 2007; Parunak et Odell, 2002; Omicini et Ricci, 2004; Ferber et Gutknecht, 2000], mais dans chacun d'eux la notion de groupe, d'organisation ou de collaboration est reliée à la notion de rôle. En général, l'organisation vise à structurer les rôles. Un rôle peut inclure la description des objectifs de l'agent jouant le rôle ou une liste des actions réalisées par l'agent. Dans [Hubner et al., 2010] le rôle décrit les droits et les devoirs d'un agent.

Ces concepts ne sont pas assez approfondis et ne sont pas adaptés à l'utilisation de scénarios pédagogiques, c'est pourquoi nous proposons notre modèle qui sera présenté en section 2.

1.5.4 Bilan

Nous pouvons noter plusieurs caractéristiques importantes qui nous ont guidées dans notre choix d'utiliser Mascaret :
– c'est un méta-modèle, ce qui nous permet d'obtenir de la généricité,
– il propose des concepts d'agents, de rôles, d'environnements,
– il utilise son propre langage d'ontologie pour décrire les connaissances.

Notre choix d'utiliser Mascaret nous permettra de décrire notre STI, nos environnements et par conséquent les représentations des connaissances (*e.g.* le scénario pédagogique sera un environnement Mascaret et sera considéré comme une connaissance pour notre STI).

1.6 Synthèse

1.6.1 Formation par la réalité virtuelle en entreprise du diagnostic *in vitro*

L'entreprise Stago qui est le financeur de ces travaux est spécialisée dans le domaine du diagnostic *in vitro*. Dans un premier temps il nous a donc fallu se renseigner sur les formations proposées par différents leaders du diagnostic *in vitro*, afin de pouvoir se positionner par rapport à la concurrence. D'après nos recherches les formations pratiquées sont essentiellement des formations classiques sur le site de l'entreprise, des formations chez le client, ou du *e-learning*.

Toutes les entreprises proposent de la formation sur le site de leur entreprise, car c'est le moins coûteux : tout le matériel est déjà sur place, les formateurs également. Dès qu'une entreprise devient plus importante elle propose également une formation dans le laboratoire du client, mais cela a un coût plus conséquent. Plus récemment, les entreprises de taille conséquente commencent à proposer des formations de type *e-learning*.

Il semblerait donc que les technologies de réalité virtuelle pour la formation ne soient pas utilisées dans ces entreprises. Le choix d'utiliser la réalité virtuelle apporte donc une touche d'originalité et d'innovation par rapport aux concurrents de Stago, mais surtout cela permet de réaliser plus efficacement des formations aux instruments Stago. En effet, la réalité virtuelle permet :
– une plus grande immersion pour l'apprenant,
– une facilité d'utilisation à moindre coût,
– la possibilité de reproduire des cas complexes,
– de faire des erreurs sans risque pour l'apprenant et le matériel,

– de se former à la manipulation d'un système.

1.6.2 Scénario Pédagogique : POSEIDON

En plus de l'environnement virtuel, nous souhaitons que le formateur puisse décrire sa propre situation d'apprentissage. Pour cela il nous faudra donc coupler notre environnement virtuel à un scénario pédagogique dont l'exécution doit permettre la construction des connaissances visées chez les apprenants. Dans la littérature, nous pouvons classer les modèles de scénarios pédagogiques en deux catégories : les modèles décontextualisés et les modèles contextualisés.

Pour les scénarios décontextualisés, les scénarios pédagogiques se basaient essentiellement sur des approches documentalistes telles que LOM ou SCORM. Par la suite il y a eu la proposition EML qui se concentre sur la description des activités et non sur les ressources pédagogiques ou encore la norme IMS-LD qui a repris la notion d'unité d'apprentissage. Le gros inconvénient de ces modèles est le manque de liens entre le scénario pédagogique et son environnement d'exécution : le déroulement des activités pédagogiques est l'équivalent d'une « boîte noire ».

Afin de compenser ce problème, les modèles de scénarios pédagogiques contextualisés ont fait leur apparition. Il y a les modèles orientés simulation (RIDES, SimQuest, FORMID, SAM) et les modèles orientés réalité virtuelle (FIACRE, GVT, METISSE). Très peu de simulations pédagogiques intègrent des informations pédagogiques relatives à l'activité attendue des apprenants dans l'environnement.

Parmi ces modèles, nous avons choisi de nous baser sur Poseidon qui est un modèle générique permettant la description de scénarios pédagogiques applicables à des simulations de réalité virtuelle. Poseidon est réutilisable dans des environnements différents et n'est pas spécifique à un domaine particulier, à un type de tâche ou à une stratégie pédagogique. C'est un modèle de scénarios pédagogiques contextualisés.

1.6.3 Systèmes Tutoriels Intelligents : PEGASE

Nous souhaitons coupler notre modèle de scénarios pédagogiques à un STI qui pourra utiliser les connaissances du scénario pédagogique et pourquoi pas réaliser les activités du formateur dans la simulation. Dans un premier temps nous avons déterminé les caractéristiques importantes pour notre STI Chrysaor : la généricité, la modularité, l'individualisation, l'adaptativité et le lien avec le scénario pédagogique.

Nous avons donc réalisé une liste non exhaustive de plusieurs STI existants en détaillant leurs implications dans chacune des caractéristiques. A partir de cette liste, nous avons déterminé que le STI Pegase était le plus complet. En effet Pegase est un STI générique et adaptatif. Il est également partiellement modulaire. Il nous faudra donc améliorer la modularité de Pegase en passant d'un comportement global à des comportements spécifiques facilement modifiables. L'individualisation étant relativement simpliste, cette caractéristique mérite également d'être améliorée afin que notre STI propose des assistances mieux adaptées

à chaque apprenant. Pour terminer, Pegase ne comporte aucun lien avec le scénario pédagogique. Il serait donc intéressant de travailler sur un modèle de scénario pédagogique afin de pouvoir le coupler à notre STI.

1.6.4 Représentation des connaissances : MASCARET

Afin de pouvoir raisonner, notre STI nécessite d'avoir accès à des connaissances du scénario pédagogique, de l'apprenant et aux connaissances métier. Il nous faut donc déterminer comment exprimer ces connaissances afin que, malgré leurs origines différentes, elles puissent être utilisées par le STI.

Nous pouvons voir que différents systèmes et modèles de bases de connaissances existent dans la littérature. En effet, certains modèles se focalisent sur les langages de programmation agent pour les bases de connaissance, d'autre proposent des communautés virtuelles de connaissances, et finalement beaucoup proposent des modèles divers et variés. Malgré leurs différences, il apparait que l'ontologie *OWL* est utilisée très fréquemment, cela indépendamment de la structure des bases de connaissances. *OWL* permet, grâce à sa sémantique formelle basée sur une fondation logique, de définir des associations plus complexes des ressources ainsi que les propriétés de leurs classes respectives.

Le méta-modèle Mascaret, développé au CERV, est largement compatible avec les langages d'ontologie de type *OWL*, c'est pourquoi nous souhaitons nous baser sur ce méta-modèle pour décrire notre environnement. Ainsi, les modèles exprimés en Uml pourront être utilisés comme une base de connaissances, comme par exemple le modèle de scénario pédagogique.

L'objectif de Mascaret est de fournir un haut niveau d'abstraction dans la conception d'applications de réalité virtuelle. Mascaret est donc un méta-modèle pour décrire les environnements virtuels et les agents qui évoluent dans cet environnement. Ce méta-modèle se base sur Uml pour décrire la structure de l'environnement (entités, positions...), les entités et les comportements d'agents.

Afin de bien situer le contexte, nous avons donc détaillé les concepts fondamentaux de Mascaret. Notamment au sujet des agents, des comportements, de la communication (utilisation du modèle proposé par *FIPA-ACL*) et de l'organisation.

1.6.5 Travail à réaliser : CHRYSAOR

Cet état de l'art nous a permis d'identifier les points forts et les faiblesses des différents modèles de STI, de scénarios pédagogiques et de la représentation des connaissances. Nous avons choisi de baser notre modèle de scénario pédagogique sur Poseidon, notre modèle de STI sur Pegase et notre représentation des connaissances en s'appuyant sur Mascaret.

En se basant sur nos critères et à partir des identifications de points faibles de chacun de ces modèles, nous pouvons déterminer nos objectifs. L'essentiel de nos travaux est dédié à notre STI. Pour cela, nous améliorerons la généricité et la modularité de Pegase en nous basant

sur les systèmes multi-agents. En effet, notre STI sera composé d'agents aux comportements propres. Nous devrons par la même occasion travailler sur la communication entre ces agents ainsi que leur organisation.

Nous réaliserons également un couplage entre Poseidon et notre STI. Pour cela nous proposerons un scénario pédagogique comme étant une instance d'un environnement. En effet, jusqu'à présent Pegase raisonnait en se basant, entre autre, sur les connaissances d'un environnement représentant le modèle du domaine. De la même manière, notre STI pourra raisonner sur les connaissances du scénario pédagogique si celui-ci est exprimé sous forme d'une instance d'un environnement, donc totalement exprimé en Mascaret, ce qui n'est pas le cas de Poseidon.

Pour représenter nos connaissances qui pourront être utilisées par notre STI pour raisonner, nous nous baserons sur le méta-modèle Mascaret. En effet, Mascaret se base sur un langage d'ontologie, proche de *OWL*, pour la génération de ses modèles. Notons que notre scénario pédagogique sera une instance d'un environnement Mascaret, de la même manière que l'environnement représentant le modèle du domaine.

Notre proposition de STI est appelée Chrysaor et est présentée dans le chapitre 2.

Chapitre 2

Proposition : CHRYSAOR

Dans la mythologie grecque, Chrysaor est le fils de Poséidon et le frère de Pégase.

Dans le chapitre 1, nous avons identifié les limites de PEGASE :
- le manque de liens avec un *scénario pédagogique* (ce qui permettrait au STI d'exploiter les connaissances issues du formateur),
- la *modularité* à perfectionner pour permettre au formateur de mettre en œuvre facilement de nouvelles stratégies pédagogiques,
- l'*individualisation* à améliorer afin de proposer des aides adaptées à chaque apprenant.

Dans ce chapitre, nous présentons notre contribution qui vise à répondre aux deux premiers points (modularité et lien avec le scénario pédagogique). Les deux premiers points s'appuient sur le même modèle, au contraire de l'individualisation qui nécessiterait un nouveau modèle qui dépasse la proposition que nous faisons ici. Pour ce faire, nous allons tout d'abord nous positionner vis à vis de ces caractéristiques dans la section 2.1, ce qui nous permettra d'identifier les concepts importants de notre proposition. A partir de ces discussions, dans la section 2.2, nous présenterons de manière formelle notre proposition ayant pour but d'améliorer la modularité de notre STI. Puis en section 2.3 nous montrerons la manière dont nous sommes parvenus à coupler notre STI à un scénario pédagogique. Pour terminer nous ferons une présentation globale de l'évolution de CHRYSAOR grâce à ces deux améliorations dans la section 2.4.

2.1 Objectifs

Dans cette section nous proposons de nous positionner vis à vis de la modularité (section 2.1.1) et du scénario pédagogique (section 2.1.2). Cela nous permettra de justifier nos propositions

ultérieures.

2.1.1 Modularité du Système Tutoriel Intelligent

La modularité a été identifiée comme une caractéristique essentielle pour notre STI, permettant de fournir au formateur une plus grande souplesse dans la construction de sa formation. Dans le cas de Pegase ceci se traduirait par une modification de sa structure interne. En effet, modifier Pegase revient à revoir l'ensemble de son programme informatique. La souplesse que nous souhaitons pour Chrysaor peut se traduire par la capacité de modifier les comportements pédagogiques du STI, de pouvoir faire intervenir un ou plusieurs humains dans une tâche à réaliser (qui remplaceront un ensemble de ces comportements), mais également de pouvoir y faire intervenir un ou plusieurs agents automatiques (qui réaliseront ces comportements).

Comportement et rôle

En fournissant au formateur la capacité de modifier les comportements, le STI devient plus modulaire. Par exemple, nous pouvons définir un comportement d'aide permettant de fortement guider l'apprenant en lui indiquant les actions à réaliser, puis souhaiter dans la suite du cursus un comportement qui va simplement suivre les erreurs réalisées. Dans l'état actuel de Pegase, il nous faudrait repenser entièrement la structure de notre STI. Alors que pour Chrysaor, l'objectif est justement de rendre aisée la possibilité de modifier ces comportements. Dans la section 2.2 nous allons donc montrer comment nous permettrons cette gestion des comportements.

Actuellement, dans Pegase chaque composant (*teacher, learner, error, compare, pedagogical, interface, expert*) a son propre comportement, et l'ensemble est exécuté de manière autonome et séquentielle. Le séquencement et les composants sont figés dans Pegase. Cependant le formateur peut vouloir adapter sa stratégie pédagogique en modifiant un de ces composants. Par exemple, l'expertise du composant *teacher* ne serait plus apportée par des comportements préalablements identifiés, mais par un humain. Pour cela, il faut permettre au formateur d'affecter ces composants à un ou plusieurs humains. Dans certains cas ces comportements peuvent être trop complexes pour être réalisés en temps réel par un humain. C'est pourquoi il faudrait pouvoir associer des comportements à des agents qui les exécuteront automatiquement. Cela permettra également de pallier l'absence d'un humain. Dans le cadre d'une application de formation, ces agents automatiques joueront un rôle de tuteur ou un rôle d'apprenant (*e.g.* compagnon, compétiteur), en fonction des choix du formateur. Un rapprochement peut être identifié entre ces objectifs et les notions de IMS-LD, dans lequel il existe également deux types de rôles : formateur (*Staff*) et apprenant (*Learner*). Par exemple, dans le cadre d'une procédure de formation comprenant trois rôles, il y a :

- un rôle de type apprenant étant le rôle principal qui devra réaliser la procédure dans son intégralité, qui sera joué par un humain,
- un rôle de type apprenant étant le rôle de compagnon de l'apprenant principal, qui sera joué par un agent automatique,
- un rôle de type tuteur qui sera joué par un humain.

Plusieurs comportements d'apprenants pourront être décrits, et suivant les objectifs d'apprentissage, l'un ou l'autre de ces comportements sera préféré. Par exemple il pourrait y avoir un comportement perturbateur (induisant en erreur l'apprenant) ou un comportement d'assistant (pour aider l'apprenant dans la réalisation de la procédure). C'est cette notion de rôle que nous allons introduire dans la section suivante.

Base de connaissances

Afin de réaliser de manière générique les comportements, ces agents automatiques doivent pouvoir raisonner sur différentes connaissances. Habituellement dans les STI existants, le raisonnement s'effectue sur les connaissances métier (*e.g.* les actions de la tâche à réaliser, les ressources, etc.) qui sont implémentées « en dur » dans l'application spécifique. Le fait d'avoir accès aux connaissances métier permet de déterminer si une action réalisée est correcte ou non. C'est le cas actuellement dans Pegase et l'aspect générique de ces connaissances métier est rendu possible grâce à Mascaret. Dans le cadre d'un scénario pédagogique, le formateur y incorpore sa stratégie pédagogique, il semble donc nécessaire que les connaissances issues de ce scénario (*e.g.* objectif pédagogique, ressources, rôle) puissent également être exploitables par les agents automatiques pour influencer la réalisation de leurs comportements. Notre proposition sur les bases de connaissances sera présentée dans la section 2.2.

Nous considérons que d'une certaine façon, ces connaissances sont des connaissances métier spécifiques : celles de la pédagogie. Il y a une certaine analogie entre les connaissances métier et pédagogiques : ces connaissances peuvent donc être représentées de la même manière. Pour cela nous allons identifier dans la section suivante, les concepts importants liés aux scénarios pédagogiques.

2.1.2 Réification du scénario pédagogique

Dans le chapitre 1 nous avions détaillé les 5 points d'un scénario pédagogique définis par Koper [2001] : *objectifs, prérequis, activités, organisations, environnements*. D'après cette définition, nous estimons que nous pouvons utiliser Uml (et l'extension proposée dans Mascaret) pour définir un scénario pédagogique. En effet un scénario est un enchaînement de tâches réalisées par des participants. Ceci peut se traduire par une activité, des actions et des rôles. Le scénario possède des pré/post-conditions. Ceci peut se traduire par les pré/post-conditions que tout comportement possède dans Mascaret. Comme pour le scénario pédagogique, ces pre/post conditions peuvent être exprimées en Uml selon deux niveaux. Le premier représentant une pre/post-condition de réalisation (par une contrainte sur une valeur de propriété du modèle par exemple) et le second représentant un niveau plus abstrait portant sur des comportements humains (*e.g.* être capable de, savoir, etc.). Le scénario se déroule au sein d'une organisation et d'un environnement constitué de rôles et de ressources. L'ensemble de ces concepts peuvent s'exprimer en utilisant Mascaret. Un scénario pédagogique serait alors considéré comme un modèle du domaine particulier traitant de pédagogie (ou de didactique plus précisément). Notre objectif est que les scénarios pédagogiques soient des connaissances sur lesquelles pourra raisonner Chrysaor, contrairement à Pegase.

Classiquement, en EIAH, les auteurs distinguent deux niveaux de scénario pédagogique [Ferraris et al., 2005]. Le scénario abstrait (modèle) et le scénario concret (exécution). Le scénario abstrait décrit la structure du scénario sans savoir comment celui-ci sera effectivement « joué ». Classiquement, ce scénario abstrait est décrit sous forme d'un diagramme d'activité. Le scénario concret affecte les ressources et les rôles aux participants réels, il peut alors être exécuté. Ceci correspond à la distinction que nous réalisons également dans Mascaret entre classe et instance, entre structure organisationnelle et entité organisationnelle et entre description de comportement et exécution de comportement. Il s'agit alors de la distinction entre un modèle de domaine de niveau M1 et son instanciation de niveau M0 (figure 1.6). La différence entre un modèle de scénario pédagogique et un modèle d'environnement virtuel c'est que le modèle de scénario pédagogique ne peut exister seul ; il est construit autour d'un modèle d'environnement virtuel.

2.1.3 Bilan

À partir de ce positionnement, il ressort que pour répondre aux besoins des formateurs nous devons baser nos travaux sur la modularité et les scénarios pédagogiques, dont les concepts importants sont le `Rôle` qui fait partie d'une `Organisation`, le `Comportement`, la `Base de Connaissances` et l'`Activité`. C'est pourquoi nous allons par la suite proposer en Uml un modèle formel afin de définir ces différents concepts. Pour cela dans la section 2.2 nous étendrons Mascaret et présenterons ces modifications visant à améliorer la modularité du STI. Dans la section 2.3 et 2.4 nous mettrons en œuvre ces concepts afin de montrer qu'ils peuvent servir de langage de scénario pédagogique et décrire le comportement du STI. Des exemples viendront illustrer ces utilisations.

2.2 Extension de MASCARET

Les concepts sur lesquels nous allons appuyer nos travaux de recherche sont les rôles et leurs organisations (section 2.2.1). Dans la section 2.2.2 nous présenterons notre modèle d'agents, avec les comportements, les base de connaissances et la communication entre ces agents. La représentation de ces concepts permet de réifier les connaissances du scénario pédagogique et par conséquent permet leur manipulation par le STI.

2.2.1 Rôle et Organisation

2.2.1.1 Définitions

Dans la section 2.1 nous avons identifié plusieurs concepts importants, dont le `Rôle` et l'`Organisation`. Classiquement, une organisation est composée de plusieurs rôles et de ressources [Ferber et Gutknecht, 1998; Hannoun et al., 2000].

Ressource

Une ressource est jouée par une instance de l'environnement. Dans une organisation métier une ressource est décrite comme une entité. Dans l'exemple d'une procédure de formation à un instrument de diagnostic biomédical, cela reviendrait à considérer un flacon de produit comme étant une ressource. Dans le cas d'un scénario pédagogique il s'agit classiquement d'une ressource pédagogique. A titre d'exemple, cette ressource pédagogique peut être un fichier vidéo montrant la réalisation d'une procédure ou un document Powerpoint représentant un cours magistral.

Rôle

A partir de la littérature basée sur les études en systèmes multi-agents ou en analyse de l'activité humaine et sociale [Ferber et Gutknecht, 1998; Hannoun et al., 2000; Zambonelli et al., 2003; Montealegre Vazquez et Lopez y Lopez, 2007], nous avons identifié les différentes notions autour du concept de rôle :
- un rôle regroupe un ensemble de comportements, de services ou d'actions,
- un rôle peut avoir des objectifs à atteindre (*e.g.* amener telle entité dans tel état),
- un rôle peut être composé d'un ensemble de contraintes à vérifier (*e.g.* que l'environnement ne soit jamais dans une situation définie ou bien qu'une entité soit toujours dans un état particulier),
- un rôle peut être responsable de ressources,
- un rôle peut être considéré comme un représentant dans l'organisation,
- un rôle peut avoir une cardinalité. Un même rôle peut être joué par un ou plusieurs agents au même moment.

Organisation

De la même manière que pour le rôle, une organisation peut être définie ainsi :
- une organisation est composée de plusieurs rôles (*e.g.* dans une association il y a un rôle président, secrétaire, trésorier, etc.),
- une organisation peut être hiérarchisée, c'est-à-dire qu'elle est composée d'organisations de niveau inférieur, d'où la notion de rôle représentant (*e.g.* une société à sa propre organisation : directeur général, employé, etc. ; et cette société peut faire partie d'un groupe commercial, et dans ce cas les rôles de directeur général pourront être les rôles représentant de chaque organisation),
- une organisation peut jouer un rôle dans une organisation de niveau supérieur (*e.g.* une équipe de football est une organisation composée de plusieurs rôles, et cette équipe peut avoir un rôle dans un tournoi qui est donc une organisation de niveau supérieur),
- une organisation peut avoir un objectif ou des contraintes à respecter (*e.g.* à nouveau dans une équipe de football un objectif sera de marquer des buts, et une contrainte serait de ne pas toucher le ballon avec la main),
- une organisation peut être responsable de ressources (*e.g.* une équipe d'intervention de pompiers est responsable de leur camion),

– une organisation propose des services et des activités qui peuvent être planifiés (*e.g.* dans une équipe de sapeurs-pompiers, il y aura une activité pour éteindre un feu de voiture).

Activité

A nouveau, nous pouvons définir une activité suivant plusieurs critères :
– une activité permet d'organiser des actions : elles seront agencées d'une manière prédéfinie (*e.g.* d'abord prendre tel objet, puis ensuite le retourner, etc.),
– une activité est organisée autour de plusieurs rôles (*e.g.* dans une procédure d'appel d'offre, il y a le rôle du client qui lance l'appel, et les rôles de fournisseurs qui font les propositions),
– une activité peut être paramétrée (*e.g.* dans le cas d'une procédure de mise à l'heure il y a deux paramètres : l'heure et la minute. Ce ne sont pas forcément des ressources),
– une activité possède un but (*e.g.* dans une activité de montage de meuble, le but est justement de réussir à monter le meuble),
– une activité peut avoir des contraintes à respecter (*e.g.* un rôle ne peut pas réaliser l'action de ramasser un objet si un autre rôle n'a pas redéposé l'objet précédemment),
– une activité possède des ressources (*e.g.* dans l'activité de montage de meuble il y a la ressource tournevis).

2.2.1.2 Support d'exemples

Afin de proposer des exemples qui exploitent nos modèles, nous illustrerons nos propos sur une application de réalité virtuelle d'un programmateur de prise de courant électrique (figure 2.1). Concrètement, un utilisateur a la possibilité de régler l'heure sur ce programmateur, ainsi que de planifier des plages horaires pendant lesquelles le courant ne passera pas.

2.2.1.3 Modèle

Nous allons nous appuyer sur des concepts Uml pour étendre le modèle Mascaret. En effet, Uml propose des notions intéressantes telles que Interface, Collaboration, Activity. Nous allons formaliser certaines notions d'Uml dans notre modèle, et cela est possible car Mascaret est une extension d'Uml.

Rôle

L'idée principale que nous retenons de la définition d'un rôle, c'est qu'il décrit l'ensemble des comportements, de services ou d'actions que doit réaliser un agent jouant ce rôle. Ceci s'approche du concept d'*interface* dans le méta-modèle d'Uml. Dans ce méta-modèle, une interface définit l'ensemble des opérations que les classes implémentant l'interface doivent réaliser. L'interface définit la liste des opérations à réaliser sans spécifier la façon dont ces opérations sont réalisées, ceci étant à la charge de la classe implémentant l'interface.

Figure 2.1 – Exemple d'application de réalité virtuelle : programmateur de prise

En d'autres termes, l'interface définit le « quoi » mais pas le « comment » sous forme d'un contrat que doivent remplir les instances des classes. Ceci correspond à la définition principale d'un rôle. La description d'un rôle est donc une sorte d'interface (héritage entre `Mascaret::RoleClass` et `UML::Interface`) (figure 2.2).

La description d'une ressource est une sorte d'interface, tout comme le rôle (héritage entre `Mascaret::Ressource` et `UML::Interface`). Certaines ressources peuvent être externes (*e.g.* un document PowerPoint), c'est pourquoi nous décrivons également la classe `Mascaret::ExternRessource` qui hérite de `Mascaret::Ressource` (figure 2.2) et qui permet de référencer une URL. Ce concept correspond aux ressources dans IMS-LD ou SCORM.

Un rôle peut également être défini comme un ensemble de contraintes. Il peut s'agir de contraintes à atteindre (rendre vraies), ce qui correspond à un objectif, ou de contraintes à surveiller (doivent rester vraies ou fausses). Dans Mascaret, la notion de `UML::Constraint` est déjà utilisée pour représenter les pré/post-conditions sur les comportements par exemple ; dans Uml, une interface en tant que `UML::Namespace` peut porter des contraintes. Nous utilisons donc ceci pour faire porter des contraintes aux rôles (figure 2.2).

Pour compléter cette définition, un rôle peut être considéré comme un représentant dans une organisation. Pour représenter ce concept, absent du méta-modèle Uml, nous ajoutons l'attribut `isLeader` de type Boolean à la classe `Mascaret::Role`. Ceci se traduit par une `TaggedValue` que pourra porter une interface stéréotypée en rôle lors de la définition d'un modèle métier dans un modeleur Uml.

Nous considérons également qu'un rôle doit pouvoir être responsable de ressources. Dans le méta-modèle Uml, une `UML::Interface` (dont hérite `Mascaret::RoleClass` et `Mascaret::Ressource`) ainsi que `UML::Property` héritent de `UML::Type`. Par conséquent,

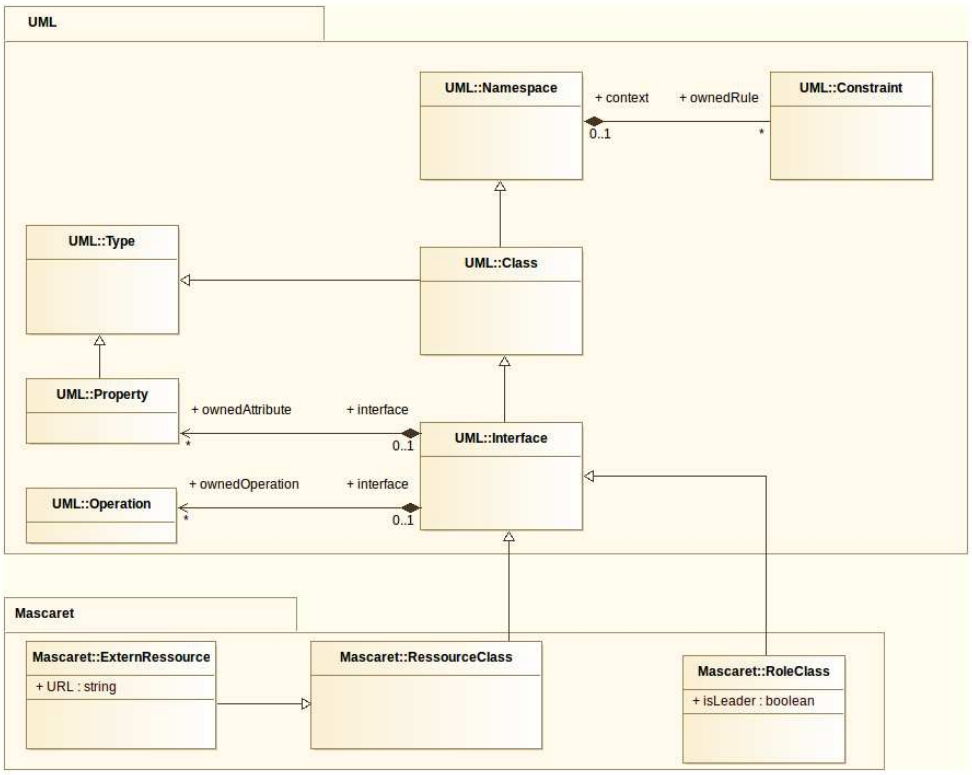

Figure 2.2 – Modèle de `RoleClass` et de `RessourceClass`

un attribut d'une interface a un type qui peut être celui d'une autre interface : un rôle peut posséder un attribut de type ressource.

La notion de cardinalité pour un rôle n'a de sens que dans le cadre d'une organisation, nous la détaillerons donc dans le paragraphe qui portera sur les collaborations.

En nous basant sur le programmateur de prise, nous proposons un exemple extrait de la définition d'un modèle métier (figure 2.3) qui décrit deux `UML::Interface` qui sont « Teacher » et « Learner ». Chacune de ces interfaces est stéréotypée en « Rôle ». Le rôle « Teacher » possède la `TaggedValue` nommée `isLeader` avec une valeur « true », l'agent jouant le rôle leader sera donc le représentant de l'organisation pédagogique. Ce rôle possède également deux opérations : `Expliquer()` et `Montrer()`, représentant des actions pédagogiques. Le rôle « Learner » possède deux opérations : `Appuyer()` et `Relacher()`, représentant des actions métier de manipulation du programmateur. Les agents implémentant les rôles « Teacher » et « Learner » devront être capable de réaliser leurs opérations respectives. Nous avons également décrit deux interfaces supplémentaires dans notre exemple qui sont les ressources : « Notice » (de stéréotype `ExternRessource`) et « Programmateur » (de stéréotype `Ressource`). « Notice » est une ressource dont est responsable le rôle « Teacher » et qui référence un document externe dont l'URL est donné par une `TaggedValue`, et qui lors de l'exécution d'un scénario pédagogique pourra être affecté à telle ou telle version de la notice d'utilisation du programmateur. La ressource « Programmateur » sera affectée à l'entité 3D représentant le programmateur dans un scénario particulier.

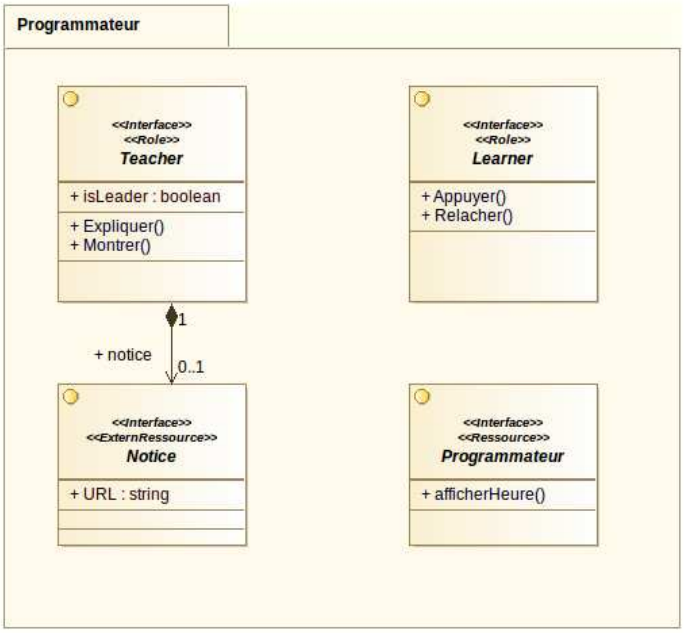

Figure 2.3 – Exemple de rôle dans l'application de programmateur de prise

Implémentation des rôles

Sous Mascaret un rôle hérite donc d'une interface Uml. Une ou plusieurs implémentations peuvent être associées à cette interface. Cette notion est représentée dans Uml par la méta-classe `UML::InterfaceRealization` (figure 2.4) qui réifie le lien entre la `UML::Class` et une `UML::Interface`. Ce lien signifie que la `UML::Class` est conforme au contrat spécifié par l'interface.
Dans Mascaret, les `AgentClass` (classes qui décrivent les types d'agents métier) sont des sortes de classes (héritent de `UML::Class`). Ainsi, un agent, instance d'une `Mascaret::AgentClass` ayant déclaré un rôle comme étant réalisé (`InterfaceRealization`), pourra jouer ce rôle dans une organisation.

En nous basant à nouveau sur le programmateur de prise, nous proposons un exemple (figure 2.5) qui décrit deux types d'agents (`UML::Class` stéréotypés `Agent`) : « Tuteur » et « Utilisateur », qui implémentent chacun un rôle (`Interface` stéréotypée en `Rôle`) : « Teacher » et « Learner ». Cela signifie que ces classes doivent fournir une implémentation pour les opérations définies dans le rôle. Un type d'objet : « Programmateur » est quand à lui stéréotypé en `Entity`, et il implémente une ressource (`Classe` stéréotypée `Ressource`) nommée « Programmateur ».

Organisation

Le critère principal que nous retenons de notre définition d'une organisation est qu'elle est composée de plusieurs rôles, ce qui s'approche fortement du concept de `collaboration` dans le méta-modèle Uml. Une `collaboration` décrit une structure composée d'éléments qui

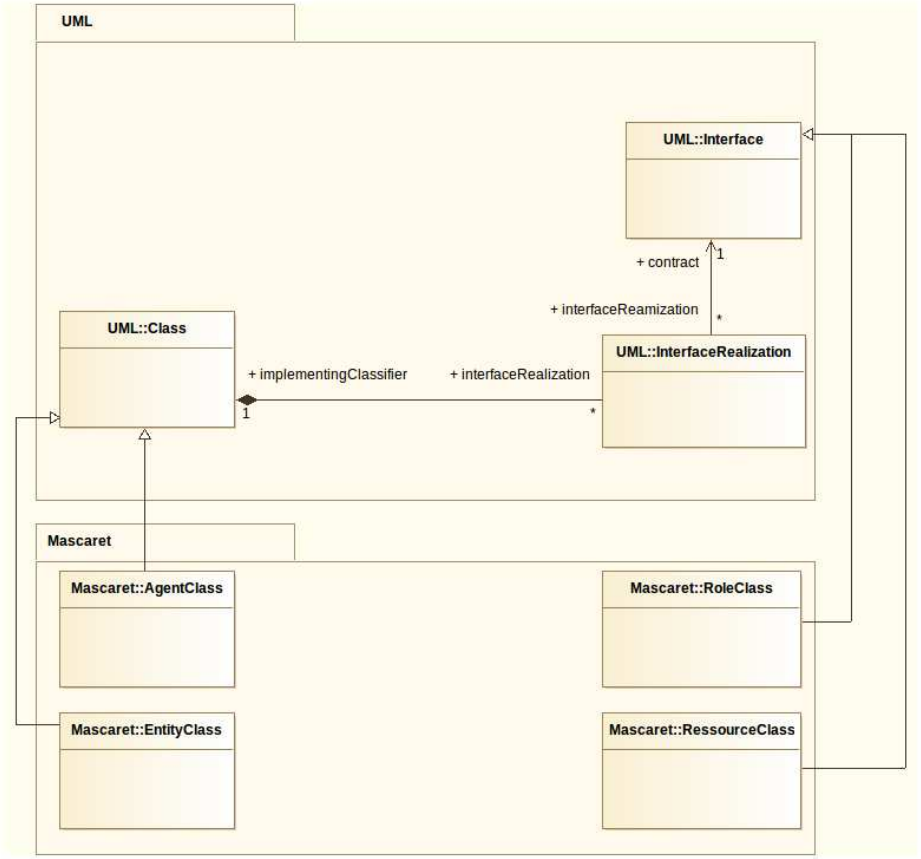

Figure 2.4 – Modèle d'implémentation de RoleClass par des AgentClass

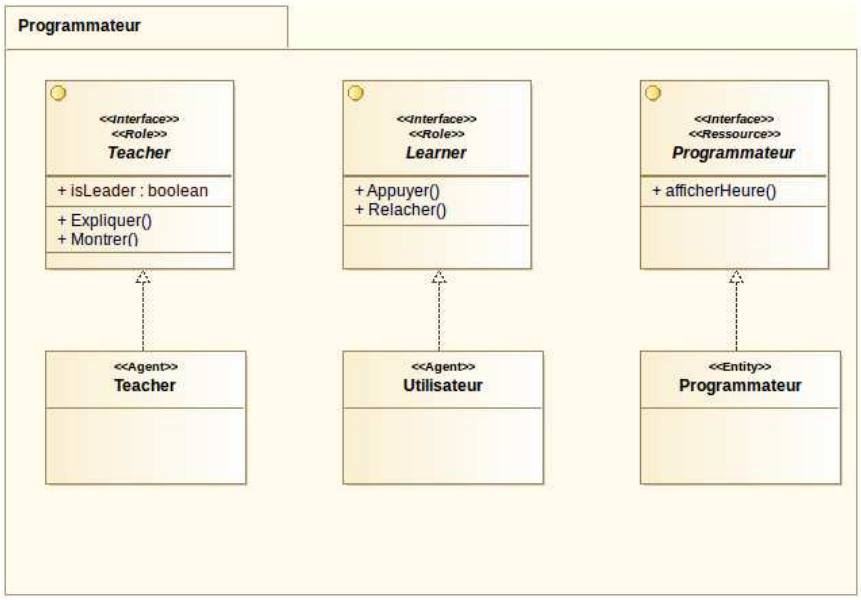

Figure 2.5 – Exemple d'implémentation de RoleClass dans l'application de programmateur de prise

collaborent entre eux, chacun réalisant une fonction spécifique, permettant d'accomplir en collaboration certaines fonctionnalités prévues. Ces éléments sont des `CollaborationRole` issus de la classe `UML::ConnectableElement`. Le but premier d'une collaboration est d'expliquer la manière dont un système fonctionne et par conséquent il intègre généralement des informations qui sont jugées pertinentes. Les collaborations sont donc des interactions entre objets, dont le but est de réaliser un objectif du système (c'est-à-dire aussi de répondre à un besoin d'un utilisateur). Ceci correspond à la définition principale d'une organisation. Nous considérons donc qu'une organisation est une sorte de collaboration. Ainsi, nous faisons hériter l'organisation de Mascaret : `Mascaret::OrganisationalStructure` de `UML::Collaboration` (figure 2.6).

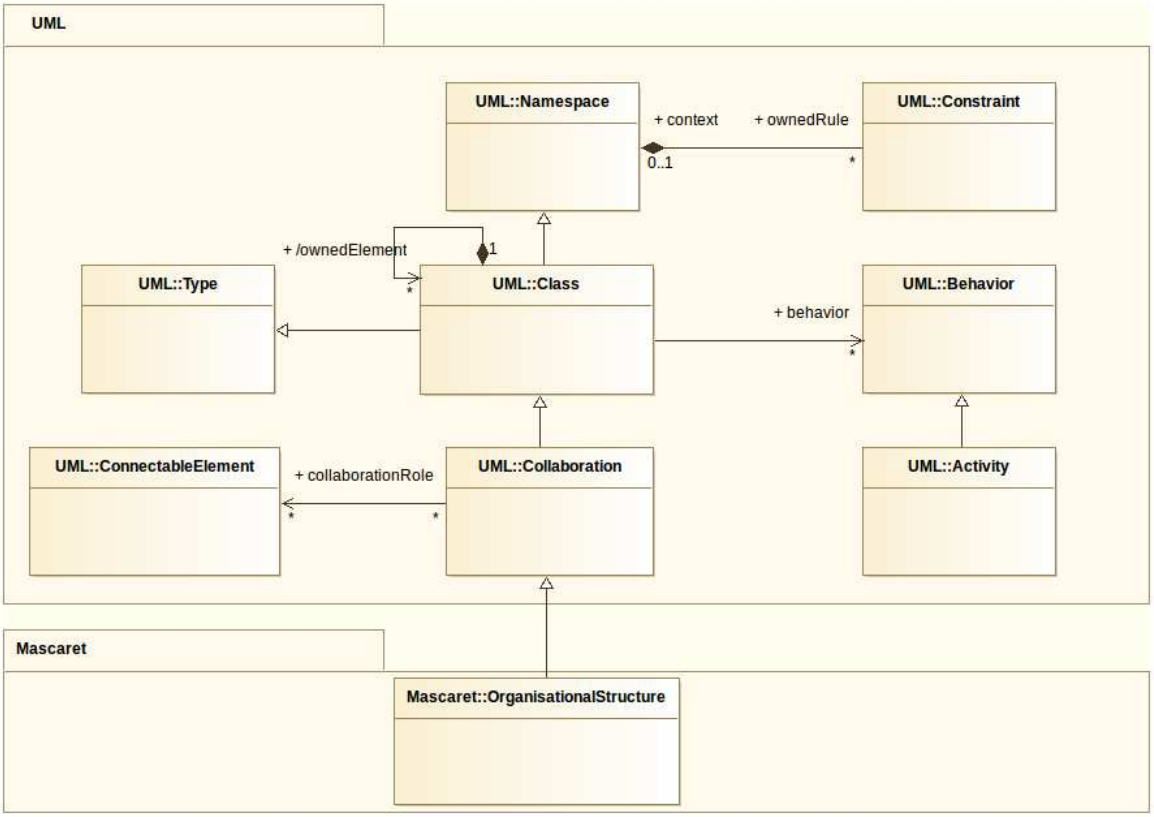

Figure 2.6 – Modèle de Structure Organisationnelle

Nous avons défini une organisation comme pouvant être responsable de ressource. Dans Uml, la classe `UML::Property` qui hérite de `UML::ConnectableElement` possède un type (figure 2.7). Une `UML::Interface` hérite de `UML::Type`, par conséquent une `UML::Property` peut être du type des classes Mascaret que nous avons définies comme interfaces, c'est-à-dire des `UML::RoleClass` ou `UML::RessourceClass` (figure 2.7). Ceci implique qu'une collaboration peut être composée de rôles et de ressources.

Une organisation peut également être définie comme un ensemble de contraintes. Ce que nous avions décrit pour la notion de rôles peut également s'appliquer ici : il peut s'agir de contraintes à atteindre, ce qui correspond à un objectif, ou de contraintes à surveiller. Dans Uml, une collaboration peut également porter des contraintes car c'est une `UML::Namespace`.

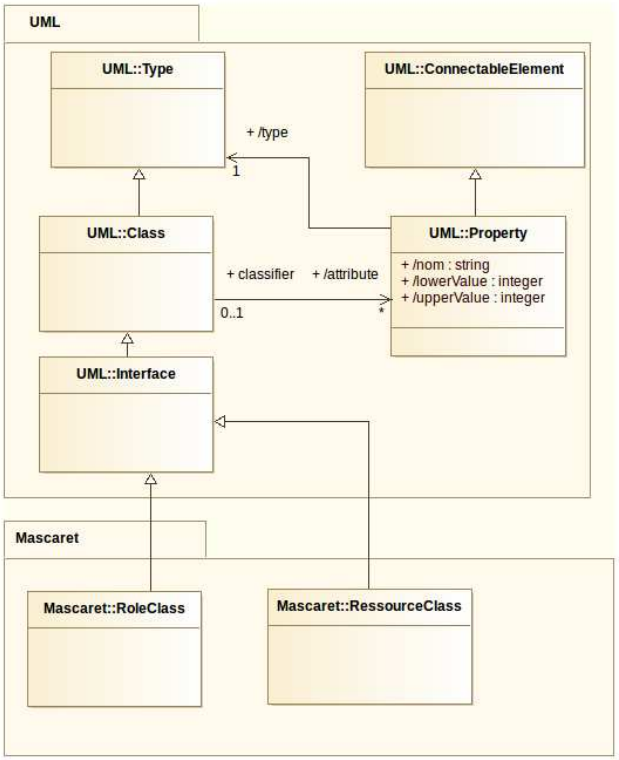

Figure 2.7 – Modèle de Structure Organisationnelle : les rôles et les ressources

Nous utilisons donc ce concept pour faire porter des contraintes aux organisations (figure 2.6).

Nous avons également défini qu'il peut exister une hiérarchie au sein des organisations. Dans Uml, une `UML::Class` peut contenir plusieurs `UML::Class` par le biais d'un `ownedElement`. Une `UML::Collaboration` héritant de `UML::Class` peut donc posséder elle même plusieurs collaborations. Ceci correspond exactement à la notion de hiérarchie au sein des organisations (figure 2.6).

Une organisation doit également pouvoir jouer un rôle dans une organisation de niveau supérieur. D'après la figure 2.8, une classe peut réaliser une interface. Nous avons vu précédemment qu'une interface pouvait être un rôle Mascaret. Sachant qu'une `Mascaret::OrganisationalStructure` hérite de `UML::Collaboration` et donc par conséquent de `UML::Class`, alors une organisation Mascaret peut jouer un rôle dans une autre organisation. Cette idée est un point très important dans notre modèle, car Chrysaor (en tant que système de tuteur intelligent), étant lui même une organisation composée de plusieurs rôles, pourra jouer le rôle « Teacher » décrit dans une procédure de formation (figure 2.9).

Afin de compléter nos définitions d'une organisation, celle-ci doit posséder des services et des activités pouvant être planifiés. Dans Uml, une collaboration possède des `UML::Behavior` (figure 2.6). Par conséquent, une organisation possédera ses propres comportements qui peuvent être des activités permettant de représenter la notion de scénario (voir section 2.3).

Nous nous devons également de compléter notre définition d'un rôle. En effet, nous avions

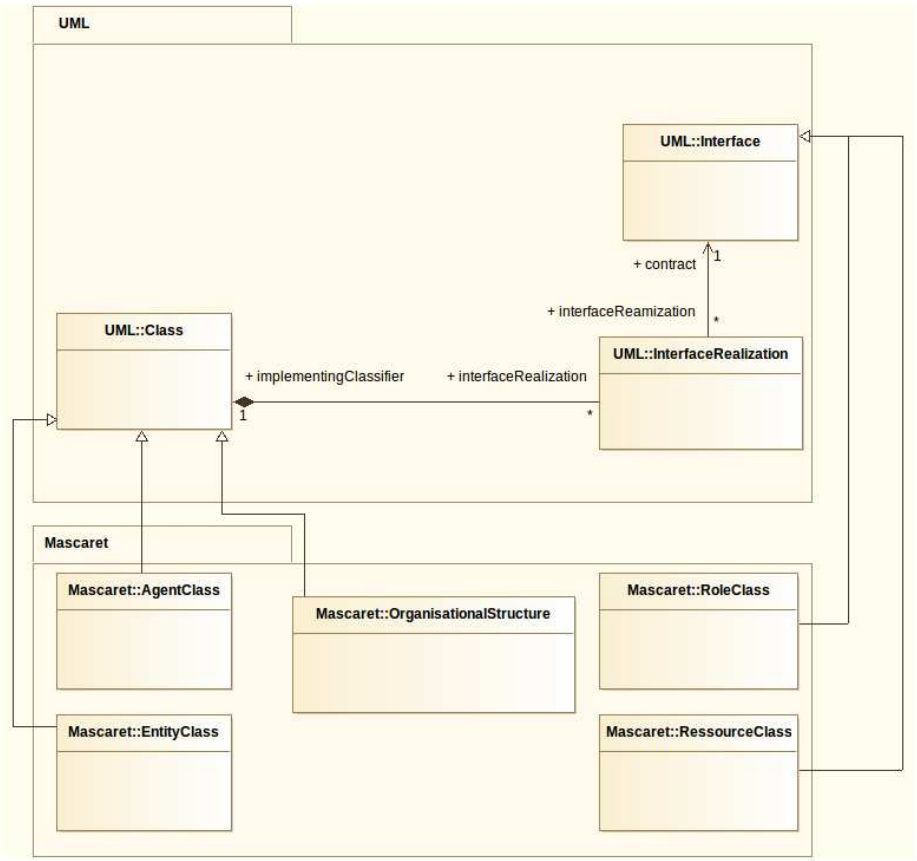

Figure 2.8 – Modèle d'implémentation de RoleClass par des structures organisationnelles Mascaret

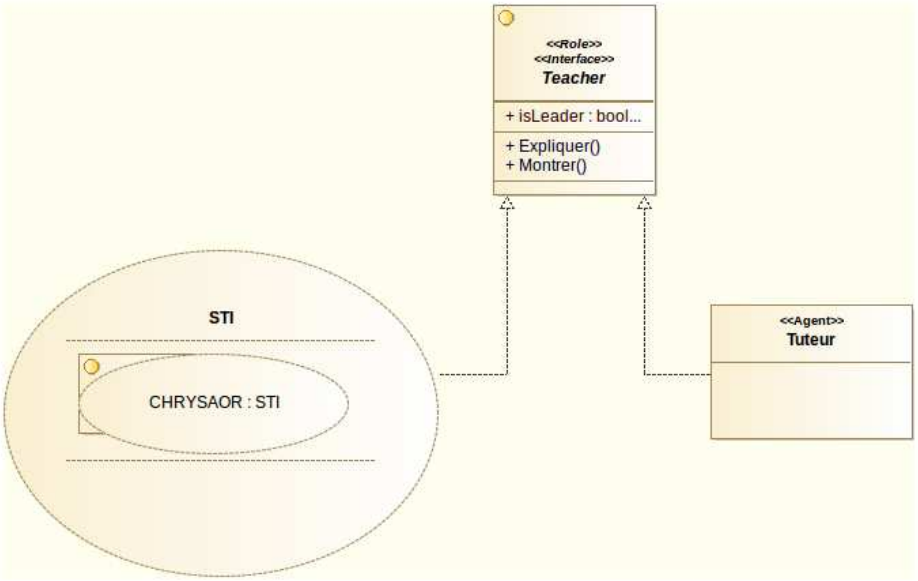

Figure 2.9 – Exemple d'organisation jouant un rôle dans l'application de programmateur de prise

abordé la notion de cardinalité au sein d'un rôle. La figure 2.6 montre que `UML::Property` possède un attribut de multiplicité. Dans le cas d'un rôle ceci est interprété dans Mascaret comme le nombre d'agents pouvant jouer ce rôle, cela implique donc que un rôle peut être joué par plusieurs agents.

Affectation des rôles

Une organisation Mascaret hérite d'une collaboration Uml, une implémentation de cette organisation devra donc être associée à cette collaboration.
Tout comme `UML::InterfaceRealization` avec `UML::Interface`, Uml définit également des `UML::CollaborationUse` qui permettent d'appliquer un modèle (décrit par une collaboration) à une situation spécifique impliquant des instances jouant un rôle dans la collaboration spécifiée. Une `UML::CollaborationUse` est définie par un ensemble de dépendances (`UML::Dependency`) qui relient une instance à un rôle (figure 2.10).

Une `UML::Dependency` possède deux éléments : un `client` et un `supplier`, qui peuvent être par héritage un `Mascaret::RoleClass` ou un `Mascaret::Agent`. Cela implique donc qu'un rôle est joué par un agent, et cet agent a la possibilité de jouer ce rôle car il est d'une `Mascaret::AgentClass` qui implémente le `Mascaret::RoleClass` du rôle. L'implémentation d'une organisation sous Mascaret sera de type `Mascaret::OrganisationalEntity`. Nous faisons donc hériter `Mascaret::OrganisationalEntity` de `UML::CollaborationUse` (figure 2.10). Dans Mascaret, l'entité organisationnelle (`Mascaret::OrganisationalEntity`) contient des `Mascaret::RoleAssignments` permettant d'associer un `Mascaret::Role` à un `Mascaret::AID` (identifiant d'un agent). Au moment de l'exécution, cela permet de lier un agent qui s'exécute dans la simulation à un rôle décrit dans la structure organisationnelle (figure 2.11). De la même manière, l'entité organisationnelle contient des `Mascaret::RessourceAssignments` permettant d'associer une `Mascaret::Ressource` à une `Mascaret::Entity`. Les `Mascaret::RoleAssignments` et les `Mascaret::Ressource-Assignments` doivent donc hériter de `UML::Dependency`.

Nous ne proposons ici aucun exemple d'utilisation d'une `CollaborationUse` dans un modeleur Uml, car un tel modeleur ne se prête pas à la modélisation d'instances 3D.

Activité

Une organisation contient un ensemble d'activités (définition en début de section et sur la figure 2.6) et l'idée principale que nous retenons de la définition d'une activité est qu'elle permet d'organiser des actions qui seront donc agencées d'une manière prédéfinie. Ceci s'approche fortement du concept d'*activity* dans le méta-modèle Uml. En Uml, un `diagramme d'activité` offre une vue sur le comportement d'un système en décrivant l'enchaînement des actions dans un processus. Les diagrammes d'activités sont similaires aux organigrammes car ils montrent les flux entre les actions dans une activité. Cependant les diagrammes d'activités peuvent également montrer des flux parallèles ou concurrents et des flux alternatifs. Le diagramme d'activités est donc pertinent pour décrire une procédure au sein d'une collaboration. Une `UML::Activity` est composée de `UML::ActivityEdge` et de `UML:ActivityNode` à partir duquel hérite `UML::Action`. La description d'une activité sous

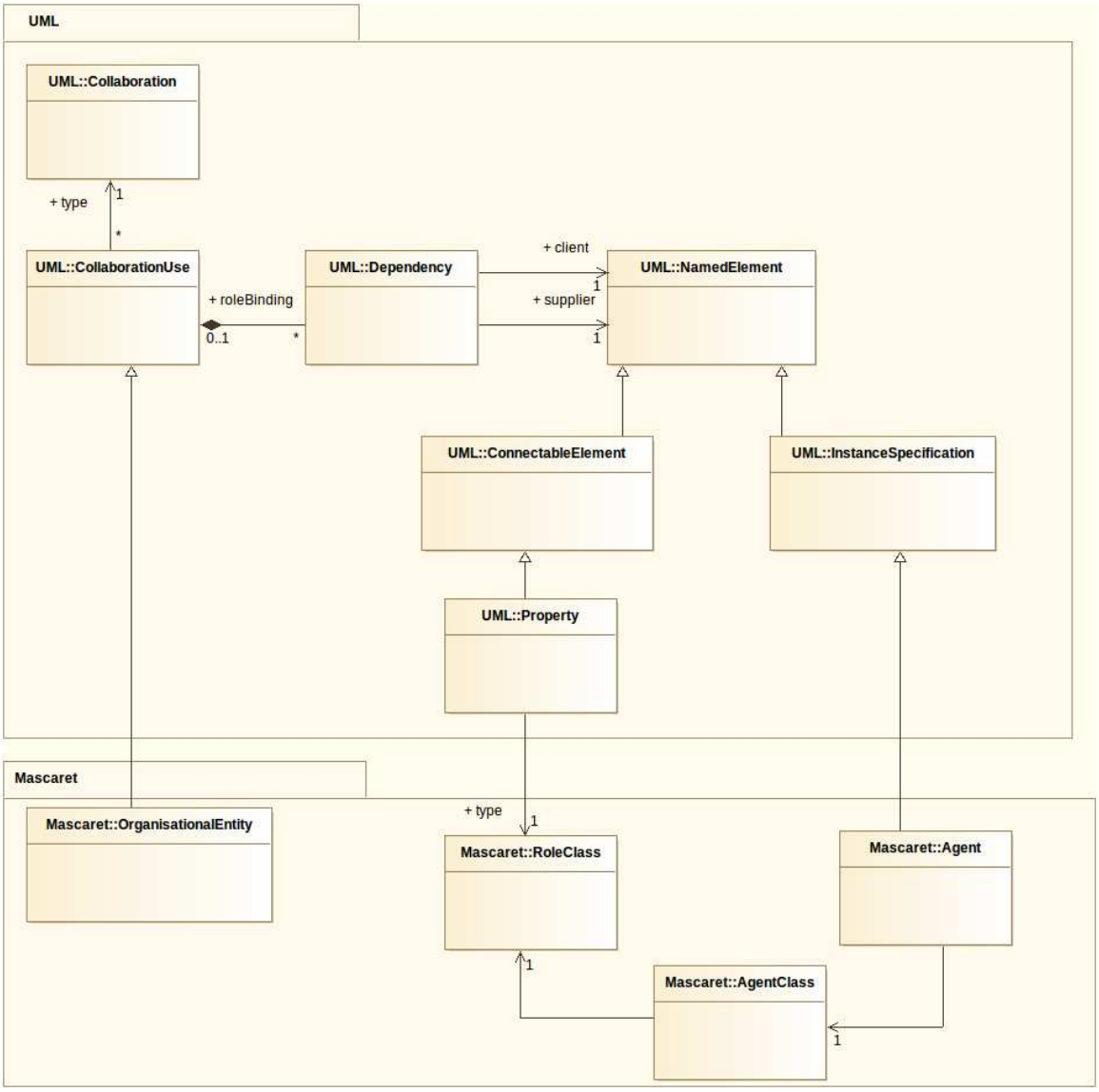

Figure 2.10 – Modèle d'entité organisationnelle

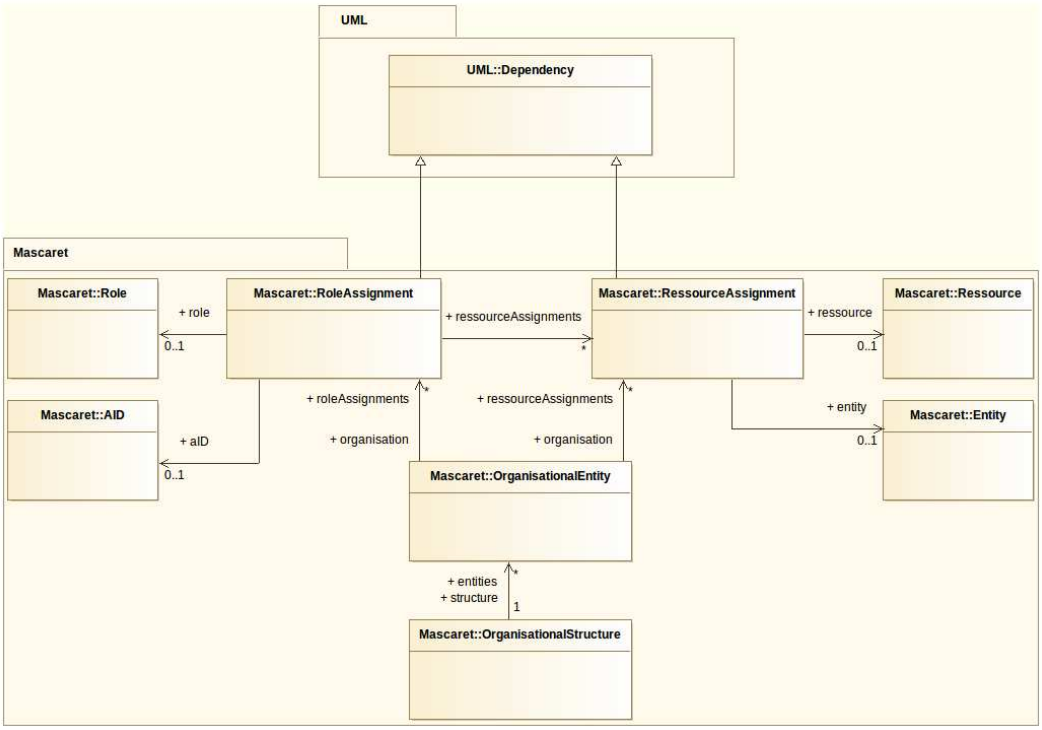

Figure 2.11 – Entité organisationnelle de Mascaret : assignation des rôles et des ressources

Mascaret (`Mascaret::Activity`) se fait donc par un héritage de `UML::Activity` (figure 2.12).

Une activité doit également être organisée autour de plusieurs rôles. Justement, Uml propose une notion de partition au sein de ses activités. En effet, une `UML::Activity` possède des `UML::ActivityPartition` représentant un `UML::Element`, ce qui peut correspondre au concept de rôle.

De la même manière que pour les rôles et les organisations, une activité possède un but et des contraintes à respecter. En Uml, les activités héritent de `UML::Behavior` possédant des pré- ou post-conditions qui sont des `UML::Constraint`. Ceci permet à nos `Mascaret::Activity` de pouvoir gérer ces notions de contraintes et de buts.

Une activité possède également la capacité d'être paramétrable. Or en Uml, un `UML::Behavior` est composé de `UML::ParameterSet` qui est lui même composé de plusieurs `UML::Parameter` (figure 2.13). Ceci permet donc de paramétrer les `Mascaret::Activity` à notre convenance (*e.g.* fixer l'heure du programmateur à 14h12).

Le dernier point que nous avons identifié est qu'une activité peut posséder des ressources. Une `UML::Activity` possède des `UML::ObjectNode` qui héritent de `UML::ActivityNode` (figure 2.14). Ces `UML::ObjectNode` sont d'un type, et donc potentiellement sont une `Mascaret::RessourceClass`. Ces concepts Uml permettent ainsi aux activités Mascaret de posséder des ressources.

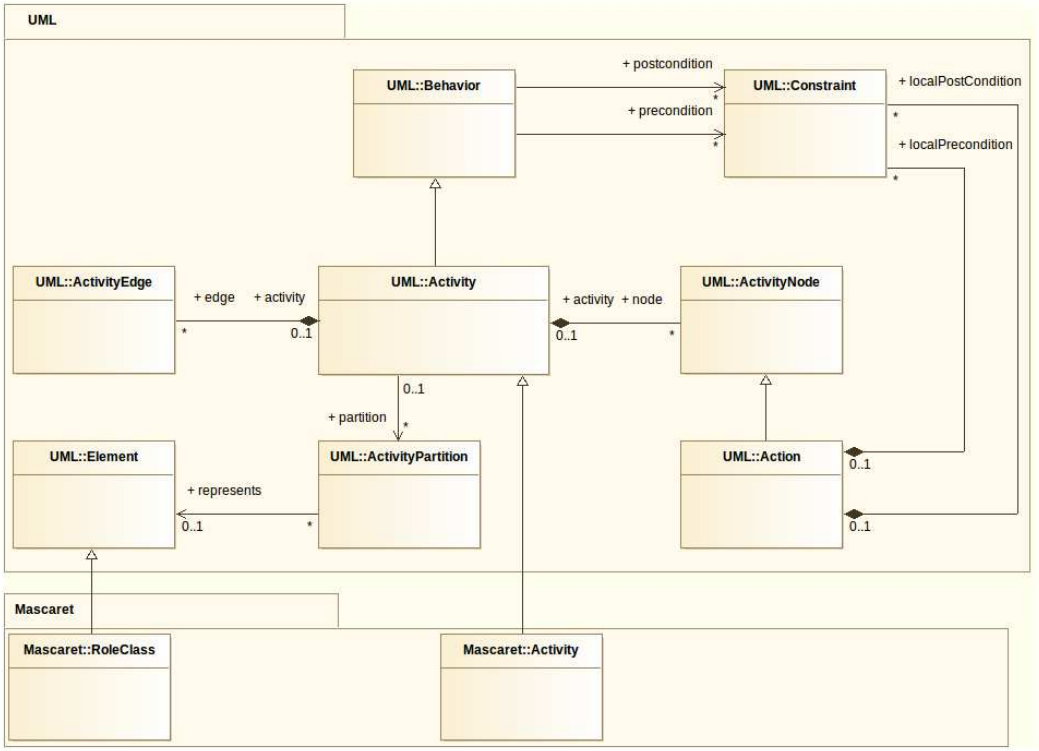

Figure 2.12 – Modèle d'activité de Mascaret

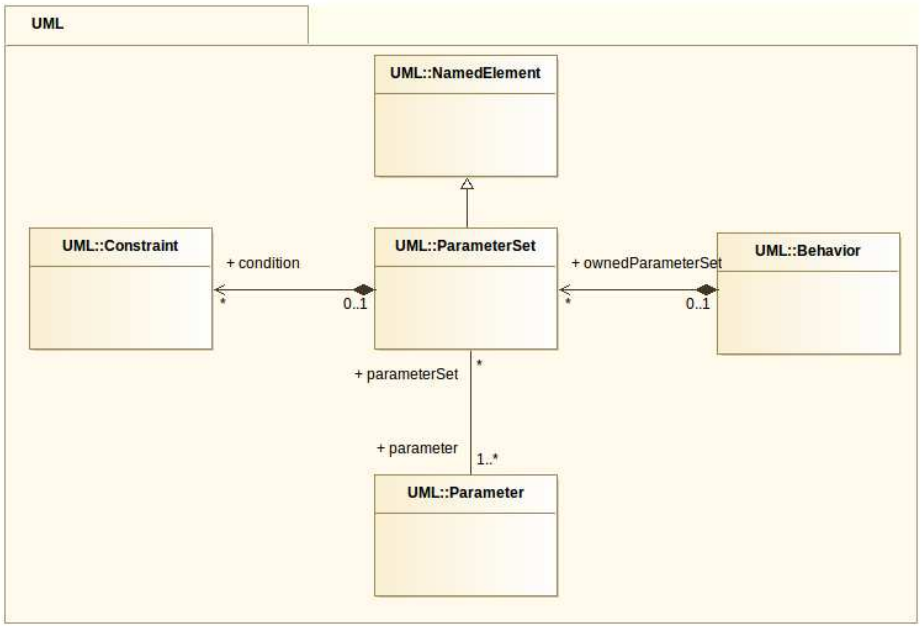

Figure 2.13 – Modèle de paramètre d'activité

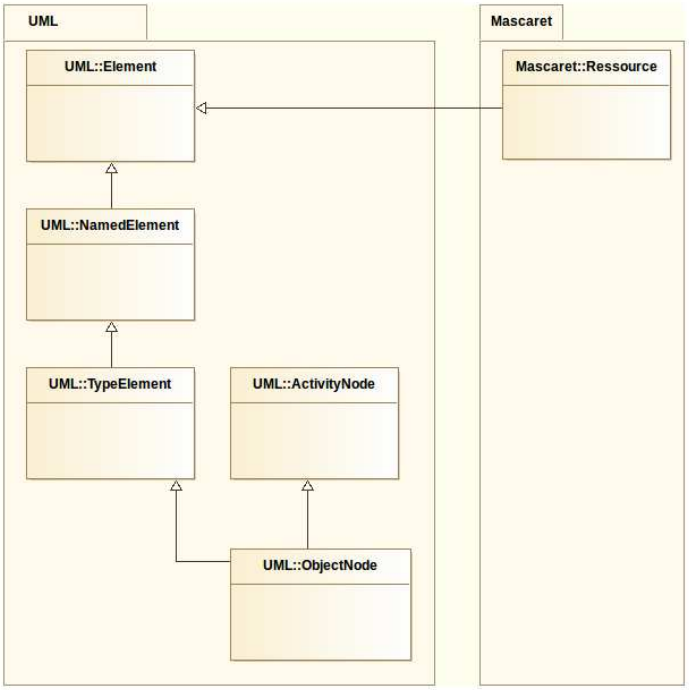

Figure 2.14 – Modèle de ressource au sein d'une activité dans Mascaret

Ce concept d'activité est très important pour notre modèle car nous considérons qu'un scénario pédagogique est une activité. En effet, toutes les définitions que nous avons énumérées pour une activité sont également applicables dans le cadre d'un scénario pédagogique. Par conséquent nous décidons de décrire notre scénario pédagogique par une activité, et dans la section 2.3 nous expliquerons plus précisément la notion de scénario pédagogique en Mascaret.

2.2.2 Agents

Mascaret définit le concept d'agent. La notion d'agent est importante car ce sont les agents qui vont jouer les rôles définis dans les organisations. C'est cette organisation des agents et la façon dont on peut les utiliser qui participent à définir la modularité.

Afin de réaliser des activités dans l'environnement, les agents doivent posséder des comportements. Dans notre cas, les agents sont soit les agents métier du domaine d'apprentissage, soit des agents pédagogiques. Dans le cas des agents pédagogiques, ils doivent agir en fonction de leurs connaissances sur la pédagogie (rôles, actions, scénario). Cela nous permet de définir un agent comme possédant des connaissances et ayant des comportements.

2.2.2.1 Base de connaissances

Afin de pouvoir raisonner sur la simulation, les agents doivent accéder à des connaissances (`Mascaret::KnowledgeBase`) portant sur l'environnement et son modèle. Au lieu d'imaginer un nouveau langage ou un nouveau formalisme pour décrire ces connaissances, nous adoptons le même formalisme utilisé pour décrire l'environnement virtuel. Ainsi la base de connaissances d'un agent référence un environnement (un `Mascaret::Environment`). De plus, un environnement est une instance d'un modèle d'environnement (`Mascaret::Model` figure 2.15) ; les modèles d'environnement peuvent être des modèles métier mais également des modèles pédagogiques (scénarios). Par conséquent, les agents peuvent posséder des connaissances métier et des connaissances sur le scénario pédagogique.

Une instance de l'environnement peut donc être une connaissance d'un agent, par exemple l'entité `BoutonHeure` dans le cadre du programmateur de prise. L'agent peut alors avoir dans sa base de connaissances la position du bouton et également l'état actuel du bouton (*e.g.* « Relaché »). Le modèle de cette instance peut également être une connaissance. En effet, l'agent possédera les informations sur les propriétés de la classe `Bouton` : cela lui permettra d'avoir connaissance du fonctionnement d'un bouton. Ainsi l'agent a dans sa base de connaissances l'information (issue de la machine à états du bouton) que si le bouton reçoit l'évènement « Press », alors il passera dans l'état « Appuyé ».

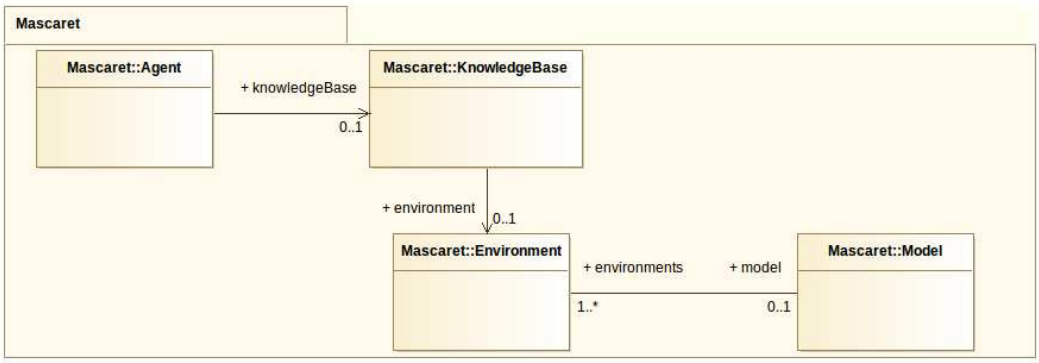

Figure 2.15 – Diagramme de classe des bases de connaissances d'un agent Mascaret

Sur ce sujet deux questions importantes restent à répondre : que sait l'agent à sa création, et comment (par quels mécanismes) sa connaissance évolue ? Classiquement, pour des raisons de crédibilité, les agents ne pourraient avoir accès qu'à une sous-partie du modèle. Certains agents peuvent être de type omniscient, c'est-à-dire qu'ils ont des connaissances sur tous les environnements. Cela signifie que l'instance de l'environnement référencé par la base de connaissances de l'agent est l'instance de l'environnement réellement simulé. Ce type de connaissance peut être par exemple celui des agents composant un STI : ces agents n'ont pas besoin d'être crédibles et ont justement besoin d'avoir le plus d'informations possibles pour s'adapter au mieux à l'apprenant.

Pour mettre à jour ces connaissances, nous proposons que les agents puissent obtenir ces informations de deux manières distinctes, par :
– Perception visuelle (*e.g.* dans une scène virtuelle, un agent « chat » connaîtrait la position d'une « souris » en la voyant). Dans le cas d'un agent omniscient, cette connaissance est

directe.
– Communication avec un agent possédant cette connaissance (*e.g.* un « chat » demande à un autre « chat » s'il connait la position de la « souris »).

Toutes ces connaissances ont pour but d'être exploitées et traitées grâce à des comportements d'agents. Dans le cadre d'une utilisation pour la définition de scénarios pour l'apprentissage de procédures, nous avons défini deux comportements génériques : un comportement de communication et un comportement de suivi de procédure. Mascaret propose déjà un mécanisme de description de comportements issus de la plate-forme d'agent *Jade* [Rimassa, 2003]. Nous utilisons ce mécanisme pour implémenter nos deux comportements.

2.2.2.2 Communication et comportements

Moyens de communication

Par notre volonté de permettre au formateur d'adapter sa stratégie pédagogique, nous avons choisi d'utiliser la technologie des systèmes multi-agents, cela afin d'obtenir une plus grande modularité de notre STI. Nos agents ont une nécessité de communiquer entre eux, c'est pourquoi nous devons leur fournir des outils de communication. Pour cela, nous souhaitons utiliser le modèle proposé par *FIPA* pour décrire une communication entre agents (figure 2.16). Les destinataires ou expéditeurs des messages sont identifiés par leur `AID` (identifiant d'agent dans Mascaret, issu de *FIPA*). Chaque message maintient un identifiant de conversation afin de faciliter le suivi du dialogue par les agents. Les messages sont exprimés dans un language et portent sur une ontologie.

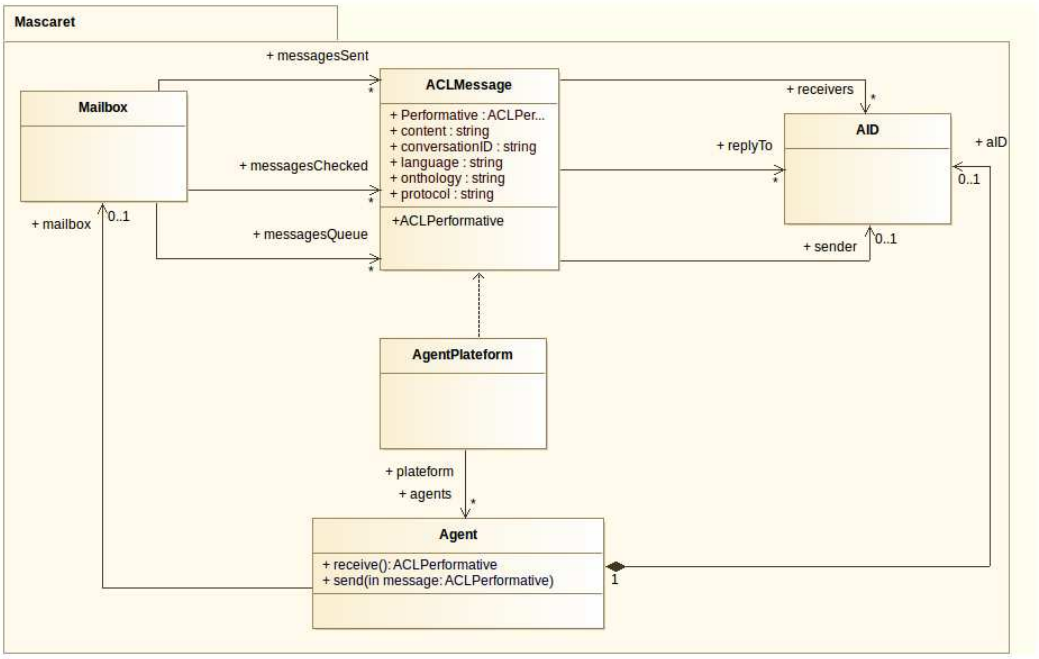

Figure 2.16 – Messages entre agents

Le standard *FIPA* s'impose car celui-ci propose justement des modèles qui prétendent être un standard pour la modélisation agent. C'est également le protocole reconnu et le plus utilisé par la communauté. Dans ce protocole, la structure d'un message est défini par un langage de communication pour les agents.

FIPA est une organisation destinée à établir des standards afin de favoriser l'interopérabilité des applications, des services et des équipements informatiques. Les messages *FIPA-ACL* sont des messages standardisés utilisant le langage ACL. *FIPA-ACL* précise les participants (*e.g.* destinataires), des descriptions du contenu (*e.g.* encodage), des contrôle de conversation (*e.g.* protocoles) et le type d'acte de langage (*e.g.* informer). Le contenu proprement dit n'est pas précisé.

Comportement de communication

Chaque agent de la plate-forme a automatiquement un comportement de communication. Ce comportement de communication est un `cyclicBehavior`, (figure 2.17) il vérifie cycliquement si l'agent a reçu un nouveau message. L'objectif de ce comportement est d'analyser automatiquement le contenu du message. Par exemple, un agent doit pouvoir demander à un autre

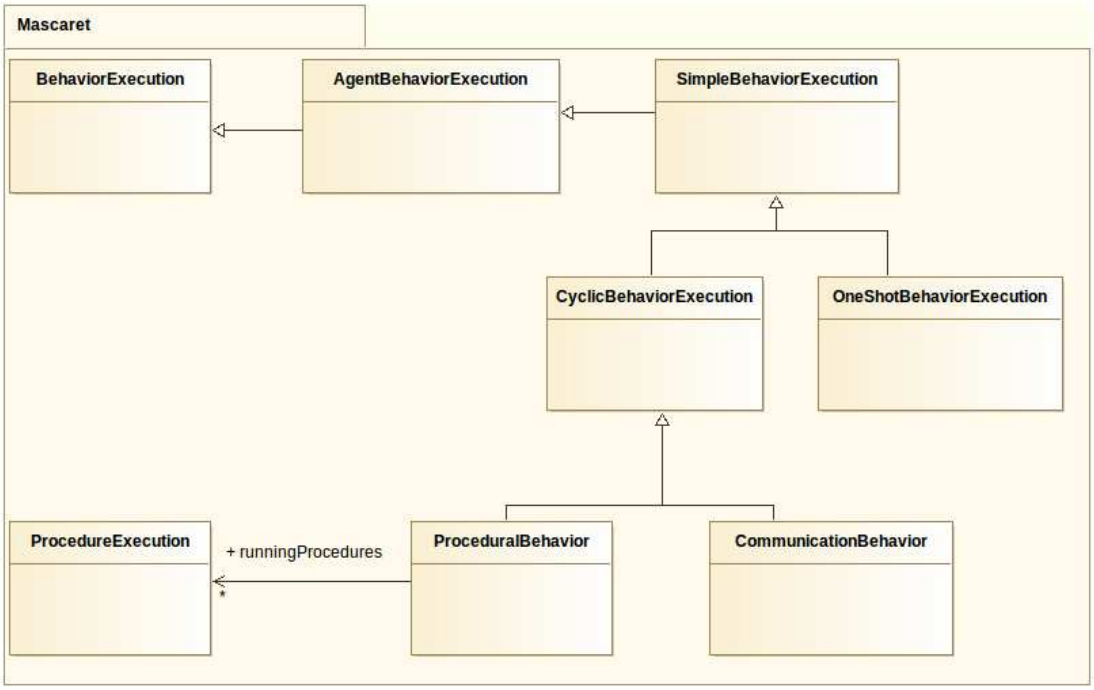

Figure 2.17 – Comportement de communication et comportement procédural

agent des informations sur des connaissances pour lesquelles il n'a pas accès (*e.g.* l'agent *error* aura des connaissances sur les types d'erreurs). Ce comportement de communication permet justement de traiter automatiquement une demande de valeur issue d'un autre agent.

Contenu des messages

Les messages entre agents peuvent être de types différents. Un message peut avoir pour but de demander de réaliser des actions (*e.g.* demander à l'agent pédagogique de réaliser une action pédagogique) ou de permettre de manipuler les bases de connaissances (*e.g.* demander ou faire modifier une valeur d'attribut).

Par exemple, un agent « Teacher » peut demander à l'agent « Learner » de réaliser l'action `appuyerBouton` sur l'entité `boutonHeure` (figure 2.18). Pour cela il envoie un message avec une performative de type `REQUEST` et un contenu de message écrit en *FIPA-SL* : `(action learner(appuyerBouton :boutonHeure)))`.

le mot *action* informe que le contenu porte sur une action, *learner* correspond à l'agent qui doit réaliser l'action, ensuite entre parenthèse on décrit l'action qui est donc *appuyerBouton* suivi du nom de l'entité sur laquelle porte l'action : *boutonHeure*.

Dans le cas où l'action est réalisable par l'agent et que celui-ci connait l'entité sur laquelle porte l'action, alors l'agent « Learner » pourra la réaliser et répondra à l'agent « Teacher » un message de performative `AGREE`. Si l'action n'est pas réalisable par l'agent (action non existante dans sa liste d'opérations, entité inconnue ou inexistante), alors l'agent « Learner » répondra par une performative de type `NOT UNDERSTOOD`. Dans le dernier exemple, l'agent « Teacher » envoie un message `INFORM` contenant l'action qu'il vient de réaliser à l'agent « Learner ». Ce message est traité par un comportement procédural. En effet ce comportement procédural connaîtra les actions faites par les autres agents de l'organisation et pourra ainsi faire évoluer ses connaissances sur la procédure.

Dans le cadre de messages permettant de manipuler les bases de connaissances, nous présentons un exemple de messages que des agents peuvent envoyer (figure 2.19). Nous avons ici des performatives de type `QUERY REF` qui permettent de demander une référence sur un attribut particulier d'une entité. Par exemple le message `QUERY REF` contenant `programmateur.heure` envoyé à l'agent « Learner » a pour but de demander au « Learner » la valeur de l'attribut heure du programmateur. Par conséquent, si cette valeur est connue, l'agent « Learner » répondra un message avec une performative de type `INFORM` contenant la valeur de l'attribut (en l'occurence `programmateur.heure = 14`). Si l'attribut est inexistant ou que l'agent n'a pas de connaissances dessus, alors il répondra par un message `NOT UNDERSTOOD`. L'agent « Teacher » peut également indiquer à l'agent « Learner » la modification de la valeur de l'attribut. Pour cela, l'agent « Teacher » envoie un message de performative `INFORM` contenant la valeur de l'attribut : `programmateur.heure = 14`.

Dans la pratique, il arrive que le STI et l'application d'environnement virtuel ne s'exécutent par sur la même machine. Nous avons donc identifié les besoins de notre STI afin de déterminer les contenus des messages entre le STI et l'application. Chrysaor peut ainsi communiquer afin de demander à l'application d'environnement virtuel :
- le modèle (Environnement et Organisation) et des informations sur les entités (valeurs des attributs, états),
- l'abonnement aux changements de l'environnement (*e.g.* à chaque modification d'une valeur d'un attribut),
- des modifications dans l'environnement (*e.g.* mettre en rouge telle entité).

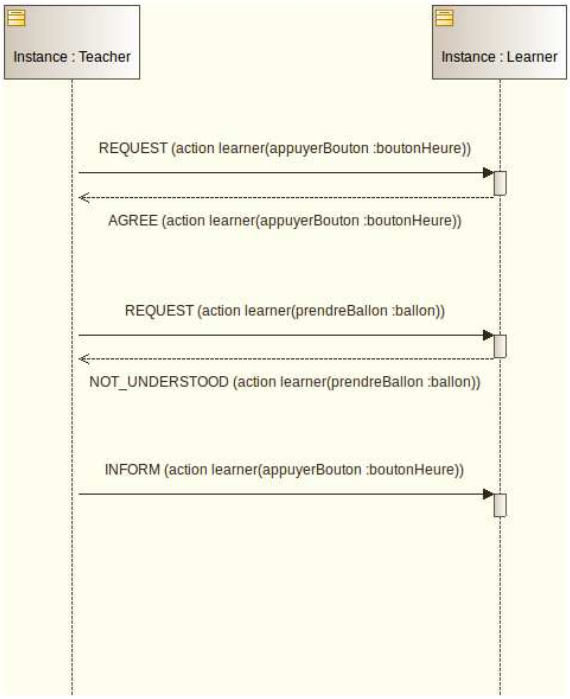

Figure 2.18 – Exemple de diagramme de séquence des demandes de réalisation d'actions par messages entre agents

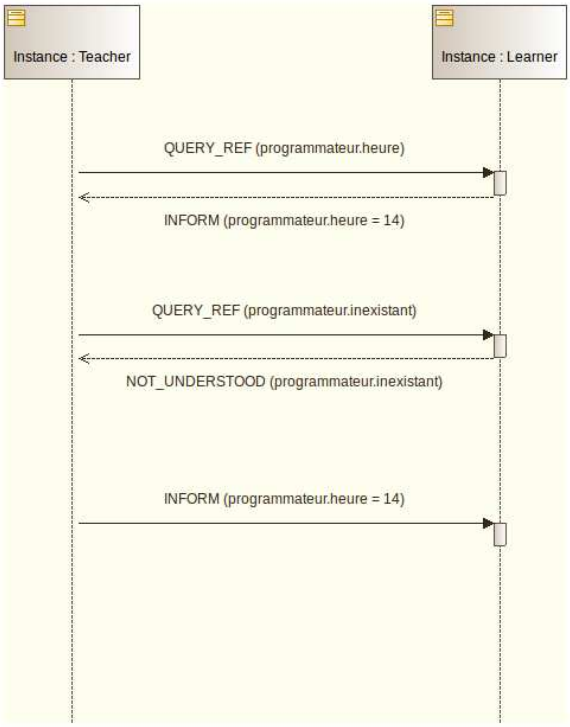

Figure 2.19 – Exemple de diagramme de séquence de manipulation de bases de connaissances par messages entre agents

De la même manière, l'application d'environnement virtuel communiquera avec Chrysaor en lui envoyant des messages prédéfinis ayant pour contenu :

– le modèle d'environnement
– le modèle d'organisation,
– des informations sur des entités
– un service de mises à jour des états/valeurs des entités,
– un service permettant d'annoncer les changements des états d'exécution des procédures.

Ces différentes syntaxes de contenus de messages entre nos agents nous permettent de combler tous les besoins identifiés. Le contenu des messages est décrit en utilisant le format XML. Ce type de communication permet aux différents agents (même ceux issus d'une machine différente de l'environnement) de pouvoir manipuler les connaissances.

Comportement procédural

Chaque agent participant à une organisation possède également un comportement de suivi de procédure. Ce comportement procédural est lui aussi un `cyclicBehavior` (figure 2.17). La procédure étant collaborative, il existe donc des agencements d'actions qui font intervenir plusieurs rôles, par exemple l'agent jouant le rôle R1 réalise l'action A1 après que l'agent jouant le rôle R2 ait réalisé l'action A2. Les agents connaissent la procédure et nous avons fournit une sémantique opérationnelle pour chacun des concepts de la procédure (fork, condition, etc.) : les agents savent donc comment les exécuter. Pour faire évoluer la procédure, un agent a besoin de savoir ce que les autres participants sont en train de faire (début d'action, action en cours, fin d'action, résultat d'action). Dans la réalité, ceci peut se faire par observation et reconnaissance d'action. Nous n'avons pas implémenté ce type de fonctionnalités. Les participants peuvent être distribués sur plusieurs machines et être joués par des agents autonomes ou par des utilisateurs humains. Il n'est donc pas possible d'avoir un mécanisme centralisé vers lequel chaque participant aurait un *pointeur*. Nous avons donc pris le parti que chaque participant possède sa propre connaissance de l'évolution de la procédure et que dès qu'un agent débute ou termine une action il envoie le message approprié à chaque participant de l'organisation. Ces messages sont alors automatiquement traités par le comportement de communication présenté plus haut.

2.2.3 Conclusion

Nous avons identifié nos besoins et cela nous a permis de définir nos concepts importants : rôle, organisation, activité, agent, base de connaissances, comportement et communication. Nous pouvons ainsi décrire notre modèle et déterminer les concepts autour des agents. Tout cela nous permet de coupler ces concepts aux notions de scénarios pédagogiques. En effet, dans la section suivante, nous allons présenter la notion de scénario pédagogique comme étant une connaissance décrite comme un environnement.

2.3 Scénario pédagogique

Dans la section précédente, nous avons identifié les concepts importants nous permettant de décrire notre modèle d'organisation : rôles, activités, comportements, base de connaissances. Afin d'améliorer le raisonnement pédagogique de Chrysaor, nous devons lui permettre de raisonner sur les connaissances apportées par le formateur, *via* le scénario pédagogique. Pegase quant à lui ne raisonne pas sur les connaissances issues du scénario pédagogique. Nous proposons donc que notre scénario pédagogique soit une connaissance au même titre que l'environnement de simulation. Pour cela, la structure du modèle de scénario pédagogique devra être décrite comme un modèle métier, celui de la pédagogie : le scénario pédagogique sera donc défini par un `Mascaret::Model` et un `Mascaret::Environment`. Notre scénario pédagogique sera ainsi une connaissance exploitable par Chrysaor.

2.3.1 Modèle de scénario pédagogique

Comme indiqué dans la section 2.1, le modèle que nous proposons permet de définir les deux niveaux de scénario pédagogique existants en EIAH : le scénario abstrait (modèle) et le scénario concret (exécution). Le scénario abstrait décrit la structure du scénario sans préjuger de la façon dont celui-ci sera exécuté. Le scénario concret affecte les ressources et les rôles aux participants réels, il peut alors être exécuté. Nous pouvons faire le rapprochement entre les deux niveaux de scénarios pédagogiques et les niveaux M1 et M0 de Mascaret. Nous considérons qu'un scénario pédagogique est un modèle exprimé en Mascaret (modèle de niveau M1, instance du méta-modèle Mascaret). L'instanciation de ce scénario est de niveau M0 (instance du modèle de niveau M1).

Afin de créer ses exercices d'apprentissage, le formateur décrit des scénarios pédagogiques. Dans Chrysaor, un scénario pédagogique est centré autour du concept d'organisation. Le formateur fournit à l'organisation des objectifs et des prérequis pédagogiques. De plus il définit les rôles, les ressources, ainsi que les activités (qui peuvent être hiérarchisées) au sein de l'organisation. Comme nous l'avions indiqué en section 2.2, un scénario pédagogique est décrit par une activité. En effet, un scénario pédagogique consiste en un enchaînement d'actions, chacune de ces actions étant réalisées par un rôle, certaines de ces actions possèdent des pré- ou post-conditions et portent sur des ressources. Toutes ces activités, qui sont des scénarios pédagogiques, sont organisées au sein d'une structure organisationnelle dans Mascaret. Par conséquent, nous considérons qu'un scénario pédagogique est une activité Mascaret décrite dans un environnement. Sachant qu'un environnement est une connaissance d'agent, alors le scénario pédagogique est également une connaissance accessible aux agents de la simulation. Ces connaissances pourront donc être exploitées par Chrysaor. Notons que les concepts qui permettent de décrire une activité serviront par conséquent à décrire les scénarios pédagogiques :

- `Mascaret::ReceiveEvent` pour les observables sur un temps ou une condition,
- `Mascaret::ReceiveSignal` et `Mascaret::SendSignal` pour communiquer entre les rôles,
- `Mascaret::CallAction` et `Mascaret::CallBehavior` pour appeler des actions pédagogiques ou d'autres activités pédagogiques,
- nœuds de contrôle (`Mascaret::Fork, Mascaret::Join, Mascaret::Condition`) pour

agencer la réalisation des activités.

Le scénario ainsi décrit peut ensuite être exécuté, ce qui correspond à la notion de scénario concret. L'exécution d'un scénario consiste en l'affectation des rôles et des ressources. Le scénario pédagogique pourra être exécuté soit par des agents autonomes, soit par des humains. Nous fournissons deux types de comportements aux agents : un comportement simple (comme par exemple un comportement procédural qui joue le rôle du formateur) et un comportement plus complexe (comme par exemple un STI).

IMS-LD est considéré comme la référence en tant que langage de modélisation pédagogique, il est donc nécessaire de pouvoir situer notre méta-modèle vis à vis de ce standard. Nous montrons dans cette section que le modèle de scénario pédagogique couvre l'ensemble des concepts d'IMS-LD et répond à certaines des critiques qui sont adressées à IMS-LD.

Un scénario pédagogique décrit en IMS-LD est structuré autour du concept de `Method`. C'est la méthode qui possède les informations sur les prérequis et les objectifs pédagogiques. Dans notre modèle, le concept structurant est l'organisation. L'organisation possède des contraintes qui peuvent servir à définir les prérequis et les objectifs pédagogiques. Dans notre cas, ces prérequis sont formalisés en Uml et portent sur des propriétés du modèle métier support de l'apprentissage, ou du modèle pédagogique. Ces conditions sont donc évaluables en cours de simulation et peuvent également servir de base de connaissances pour les agents pédagogiques. En IMS-LD ces conditions ne sont pas formalisées et servent donc uniquement d'informations pour les utilisateurs.

Le point le plus connu dans IMS-LD est sa structuration en trois niveaux hiérarchiques `Method`, `Play` et `Act` (figure 2.20). La méthode définit le cursus, la pièce définit les chapitres et les actes définissent les exercices. Dans Mascaret, ce niveau de hiérarchie est obtenu par le fait qu'une activité peut appeler d'autres activités. La différence est que d'une part il n'y a pas de limitations ou d'obligations dans le niveau de hiérarchie et que d'autre part les opérateurs de contrôle qui organisent l'enchaînement des activités sont plus riches.

Dans IMS-LD, l'acte est le seul niveau de hiérarchie à définir des activités réelles pour les participants. Un acte, par le biais des `Role-Part`, permet d'assigner une `Activity` à un `Rôle`. Le standard IMS-LD définit deux types de rôles qui sont « Learner » et « Staff ». Dans IMS-LD les rôles ont des objectifs obtenus en réalisant des activités d'apprentissage. Ces activités sont réalisées dans un environnement composé d'objets d'apprentissage et de services. Les concepts de rôle et d'activité existent également dans le modèle que nous proposons et partagent la même sémantique. Cependant des différences majeures existent entre notre modèle et celui de IMS-LD. Tout d'abord dans Mascaret, les activités peuvent être collaboratives, cela signifie que plusieurs rôles peuvent participer à une même activité. Mais surtout l'activité dans Chrysaor ne se limite pas à une « boîte noire » s'exécutant dans un environnement et produisant des données en sortie. Dans Chrysaor l'activité est formellement décrite par l'enchaînement des actions à réaliser par les rôles et l'effet de ces actions sur l'environnement est également décrit.

Nous avons donc montré ici qu'il est possible, à partir d'un scénario définit en Chrysaor, de générer un scénario en IMS-LD mais que l'inverse ne l'est pas.

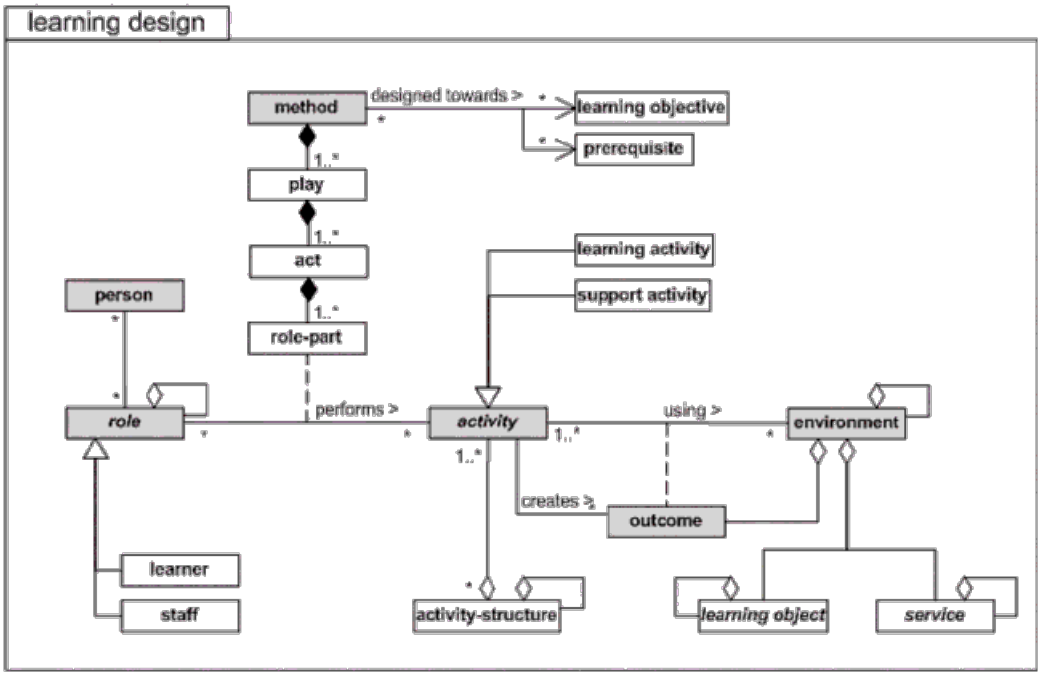

Figure 2.20 – Modèle conceptuel du niveau A de IMS-LD

2.3.2 Flot de conception

Afin de réaliser une application complète de formation par la réalité virtuelle (application, tuteur, scénario), plusieurs personnes de domaines différents doivent intervenir. La figure 2.21 résume l'ensemble des intervenants qui seront décrits dans cette section.

Pédagogue spécialisé en réalité virtuelle

Dans un premier temps, un pédagogue aidé d'un expert en réalité virtuelle définit le modèle du STI (niveau M1). En effet, ses qualités de pédagogue lui permettent de définir les meilleures stratégies pédagogiques pour le STI, et les connaissances de l'expert en réalité virtuelle lui permettent de mieux cibler le potentiel de la réalité virtuelle pour la formation. Le pédagogue devra définir les actions de bases, c'est-à-dire les actions pédagogiques telles qu'une assistance, ainsi que des procédures plus évoluées qui décrivent le comportement du tuteur. Ces actions pédagogiques seront implémentées par un informaticien. Le pédagogue définit ainsi les différents rôles et les actions pédagogiques. Il définit également les organisations, les procédures et les actions haut niveau. Il peut ainsi décrire des actions qui peuvent être une action simple tel qu'une assistance (*e.g. mettre en rouge*), ou bien une action plus complexe et décrite par une activité (*e.g.* l'action *identifier le type d'erreur*), et dans ce cas le pédagogue peut segmenter ses rôles. Le pédagogue peut également décrire une organisation plus complexe avec une procédure comprenant plusieurs rôles (error, teacher, learner,...) réalisant des actions. Au moment de l'exécution, nous devrons affecter un rôle à un agent. Tout cela est générique et s'appuie sur le méta-modèle métier et pédagogique. Par

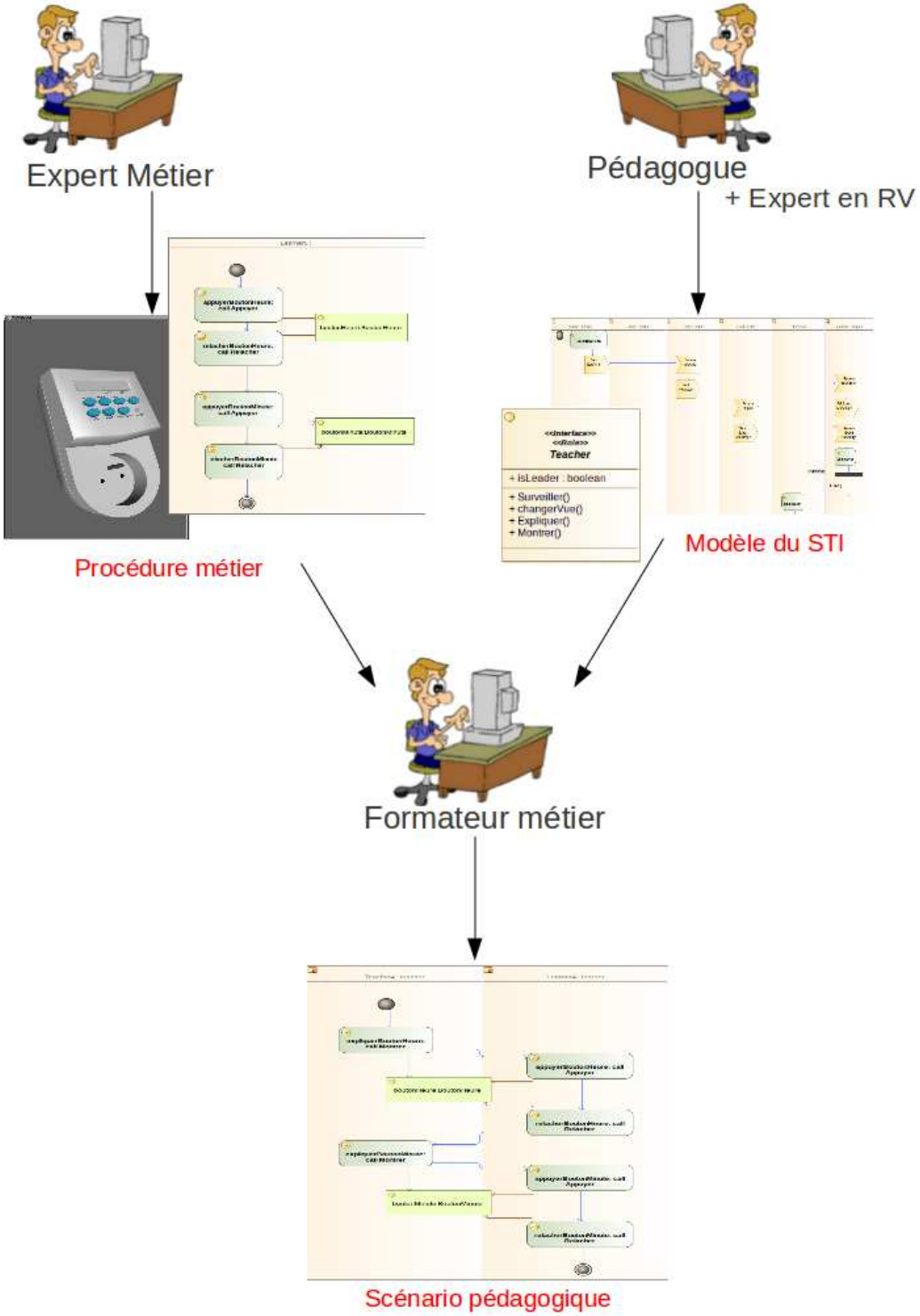

Figure 2.21 – Flot de conception décrivant les différents intervenants à l'élaboration du STI et de l'application

conséquent, toutes les connaissances seront des données ou des paramètres pour les actions et les procédures.

Actuellement dans Chrysaor, nous utilisons les assistances décrites par Lourdeaux [2001] dans le tableau 2.1.

Type d'assistance	Exemple d'assistance
Enrichissement	Ajout de symboles visuels, sonores ou de films d'animation.
Dégradation	Détérioration du réalisme (flous à l'arrière-plan et/ou sur les côtés, réduction d'objets, iconisation, etc.).
Réhaussement	Exagération de la réalité (représentation d'objets à plus grande échelle, objets surréalistes, plus lumineux, plus brillants, etc.).
Simplification	Allégement de la scène virtuelle (objets simplifiés, systèmes kinesthésiques simplifiés, représentation en fil de fer, etc.), représentation schématique de certains appareils.
Restriction	Limitation de certains déplacements ou manipulations (limitations du périmètre dans lequel l'utilisateur peut se déplacer, etc.).
Animation	Séquence animée (positionnement automatique, clé qui tourne automatiquement une fois mise en place, etc.).
Décentrage	Changement du point de vue habituellement attaché à l'œil du formé immergé (vue de derrière, au-dessus, etc.).
Modification	Modification d'aspect, de texture (changement de couleurs, clignotement d'objets, etc.).
Modélisation	Représentation de concepts abstraits, de phénomènes physiques invisibles à l'œil nu, de type de pannes, etc.
Visualisation	Visualisation de mécanismes cachés (intérieur d'un moteur, engrenages, etc.).

Table 2.1 – Liste d'assistances pédagogiques, d'après Lourdeaux [2001]

En se basant sur ces concepts, Marion [2010] a proposé des primitives qui sont fournies par Mascaret. Ces primitives peuvent être de deux types : soit portant sur l'environnement virtuel, soit portant sur le scénario pédagogique. Marion a identifié quatre primitives portant sur l'environnement virtuel :

- Manipulation d'une entité : cette primitive permet de créer, modifier ou supprimer une entité.
- Manipulation d'une entité organisationnelle : cette primitive permet de créer, supprimer ou modifier l'affectation des rôles d'une entité organisationnelle.
- Manipulation d'une procédure métier : cette primitive permet de démarrer ou stopper l'exécution d'une procédure métier par une équipe au sein de l'environnement virtuel.
- Manipulation des capacités de l'apprenant dans l'environnement : cette primitive peut être utilisée pour modifier les capacités d'action ou de perception de l'apprenant.

Dans le cadre des primitives portant sur le scénario pédagogique, Marion en a identifiées quatre également :

– Décharger un utilisateur d'une action : cette primitive permet au formateur d'alléger l'activité d'un apprenant en faisant en sorte qu'une action particulière de l'activité métier ne soit plus réalisée par un utilisateur mais par un agent.
– Montrer la réalisation d'une action : cette primitive donne la possibilité au formateur de mettre l'accent sur une action particulière dans l'environnement en montrant à l'apprenant sa réalisation par un agent.
– Envoyer un message à un acteur du scénario pédagogique : cette primitive permet au formateur d'envoyer des messages à un acteur du scénario, qu'il soit humain (apprenant ou formateur) ou virtuel (un agent, dans le but de modifier son comportement).
– Mettre une ressource pédagogique à disposition, de manière ponctuelle ou permanente (documents de cours, vidéos d'illustration, etc.).

Dans le cadre de l'application du programmateur de prise, nous montrons quelques assistances. Par exemple une action qui a pour but de montrer un objet peut appeler l'opération `Montrer` du rôle correspondant, et cette opération est implémentée par la réalisation de mettre en rouge l'objet considéré (figure 2.22), ou bien de mettre tous les objets en transparence, sauf l'objet concerné (figure 2.23). De la même manière, la réalisation de l'opération `Expliquer` peut se faire par exemple par un affichage texte de type OSD (*On-screen display*) (figure 2.24), ou directement par une synthèse vocale selon la plate-forme utilisée. Notons que ce type de modélisation permet d'avoir des scénarios pédagogiques indépendants de la plate-forme d'exécution. En effet, seule la description de l'opération `Montrer` (ou `Expliquer`) est contenue dans le modèle, car son implémentation dépend des choix du pédagogue et de la plate-forme d'exécution.

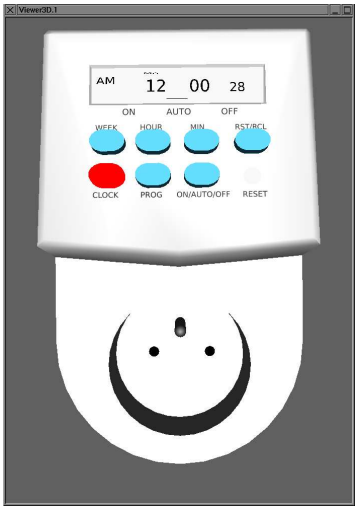

Figure 2.22 – Exemple de mise en rouge d'une entité dans l'environnement du programmateur de prise

Le pédagogue peut également définir des opérations qui permettent d'expliquer une action (contrairement à l'exemple précédent où il était question d'expliquer un objet). Son implémentation peut se faire de façon générique dans Mascaret car toutes les connaissances sont accessibles : connaissances sur l'action, sur l'action suivante, etc. Par exemple, la réalisation de `ExpliquerAction` correspond à indiquer qu'il faut appuyer sur le bouton et le

Figure 2.23 – Exemple de mise en transparence de l'environnement sauf d'une entité particulière dans le programmateur de prise

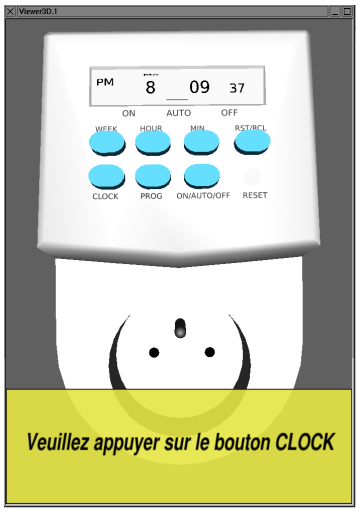

Figure 2.24 – Exemple de texte explicatif en OSD dans le programmateur de prise

relâcher (connaissances sur les actions suivantes), et expliquer que ceci changera l'heure du programmateur (connaissances sur la machine à état du programmateur).

Expert du domaine

Un autre intervenant doit participer à l'élaboration de l'application : c'est l'expert du domaine d'application qui permettra de décrire le modèle métier. Les connaissances de l'expert seront donc centrées sur le domaine sur lequel porte l'application. Dans le cadre du programmateur de prise, ce sera les connaissances des composants et du fonctionnement du programmateur.

Formateur métier

Pour terminer, un formateur métier participera également à fournir son expertise. En effet, celui-ci importera le modèle du STI décrit par le pédagogue, ainsi que le modèle métier décrit par l'expert du domaine afin de réaliser des scénarios pédagogiques. Par exemple, il pourrait souhaiter réaliser un scénario avec deux rôles : un rôle qui doit réaliser la procédure métier, et un rôle qui réalisera des actions définies dans le modèle du STI, et tout cela en étroite interaction. Cela correspond exactement aux exemples que nous fournissons dans la section 2.3.3.

2.3.3 Exemples

Dans le cadre de la formation à la manipulation d'un système, nous avons identifié deux grandes catégories de scénarios pédagogiques. Tout d'abord les scénarios qui permettent une acquisition des connaissances sur le système (*e.g.* structure du système, identification des sous-systèmes, but de chaque sous-système), ainsi que les scénarios visant un apprentissage de procédures de maintenance ou d'utilisation du système. C'est le formateur métier qui a la charge de rédiger ces scénarios en s'aidant du modèle métier et du modèle pédagogique. Nous présentons dans cette section des exemples classiques en nous basant à nouveau sur le modèle du programmateur de prise. Dans cet exemple, la procédure pour effectuer un changement d'heure s'effectue comme ceci : il faut appuyer sur le bouton horloge, le maintenir enfoncé, puis appuyer et relâcher plusieurs fois sur le bouton heure pour obtenir l'heure souhaitée. Ensuite il suffit de relâcher le bouton horloge.

2.3.3.1 Scénarios pour l'acquisition de connaissances

Comme premier exemple, nous proposons un scénario pédagogique qui permet d'expliquer et de montrer la composition du système (figure 2.25). Le formateur ne décrit qu'un seul rôle, celui du « Teacher » qui va montrer les différents éléments du système. Cette procédure est composée de plusieurs actions : `changerVueHorloge`, `montrerBoutonHorloge`, `expliquerBoutonHorloge`, etc., qui appellent toutes une opération définie dans le rôle `Teacher`. En effet, ces actions seront réalisées par l'agent jouant le rôle « Teacher », qui possédera la réalisation des opérations `Montrer(), Expliquer(), ChangerVue()`. Cet

exemple présente les deux ressources (utilisées dans la procédure) : `boutonHorloge` et `boutonheure`.

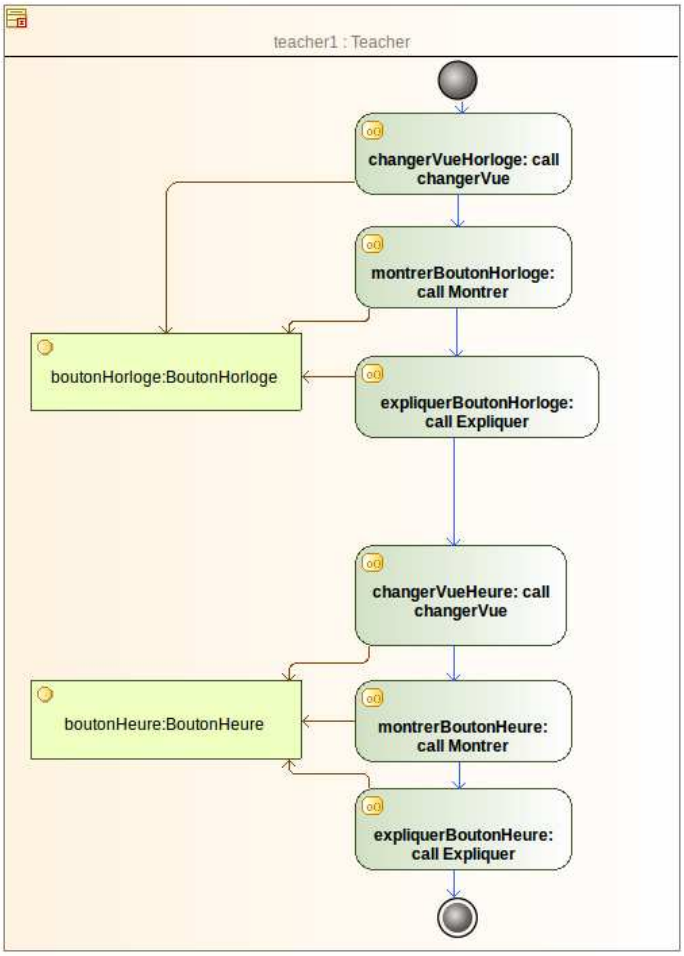

Figure 2.25 – Exemple de diagramme d'activité représentant un scénario pédagogique de présentation de deux objets de l'environnement

Nous proposons un autre exemple de scénario pédagogique qui permet d'expliquer les fonctionnements et les buts des composants du système (figure 2.26). À nouveau, nous avons plusieurs actions qui appellent les opérations définies dans un rôle. Dans cet exemple, l'agent « Teacher » présente tout d'abord le programmateur, puis l'écran (montrer puis expliquer). Ensuite l'agent montre le bouton horloge et interagit avec ce bouton et le bouton heure (appuyer et relâcher), cela afin de montrer à l'apprenant le fonctionnement de ces boutons. Ensuite l'agent « Teacher » réalise l'action `ChangerVue` vers l'écran et explique la conséquence de l'appui du bouton horloge et heure (en l'occurence augmentation de 1 de la valeur de l'heure).

Dans ces deux exemples, nous avons un rôle « Teacher » qui possède plusieurs opérations (`ChangerVue()`, `Expliquer()`, `Montrer()`, `Appuyer()`, `Relacher()`) et qui peut agir sur les différentes ressources de l'environnement (figure 2.27).

De la même manière que pour les rôles, nous décrivons des interfaces que nous stéréotypons

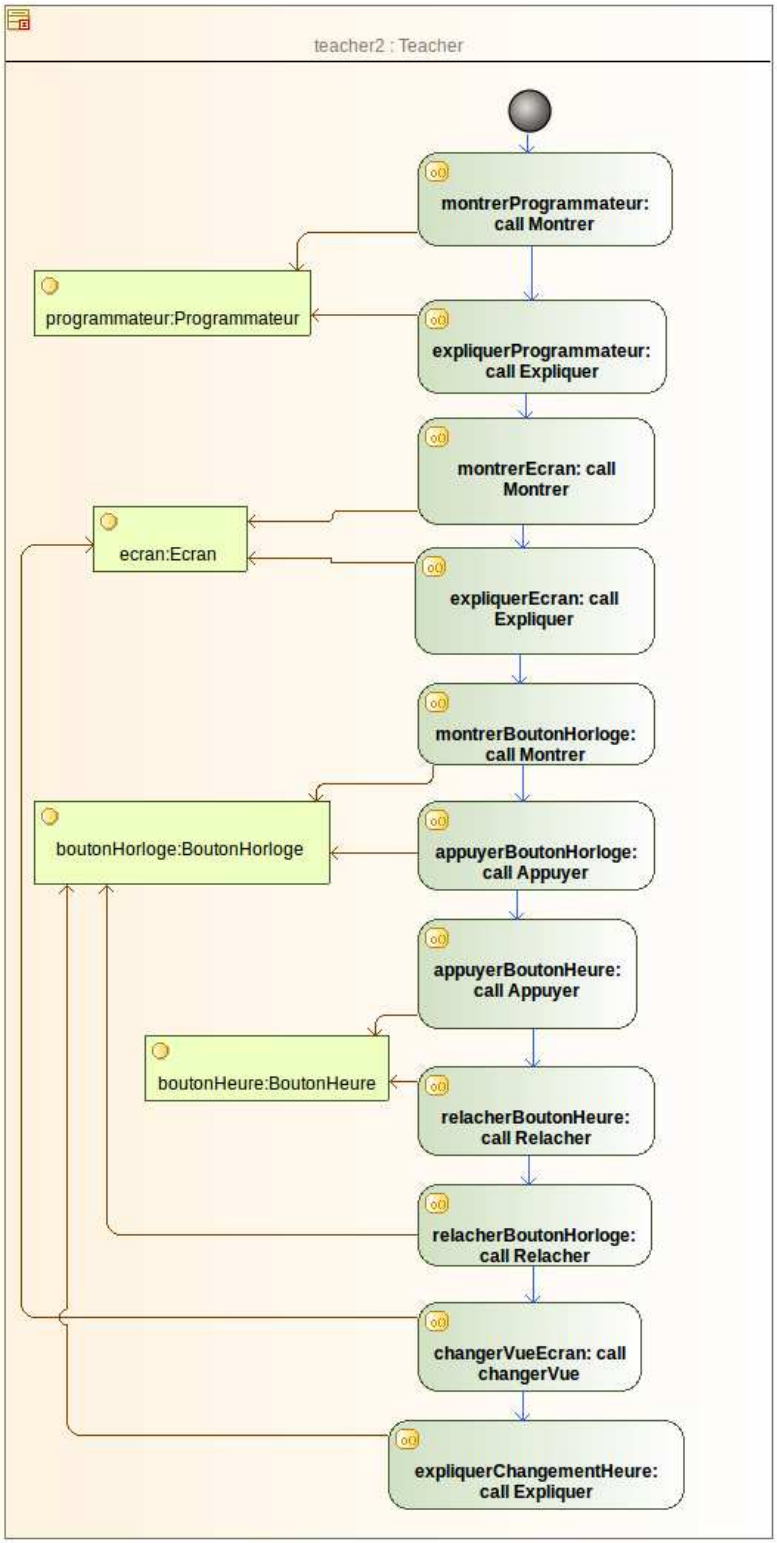

Figure 2.26 – Exemple de diagramme d'activité représentant un scénario pédagogique de présentation des objets de l'environnement et du fonctionnement du système

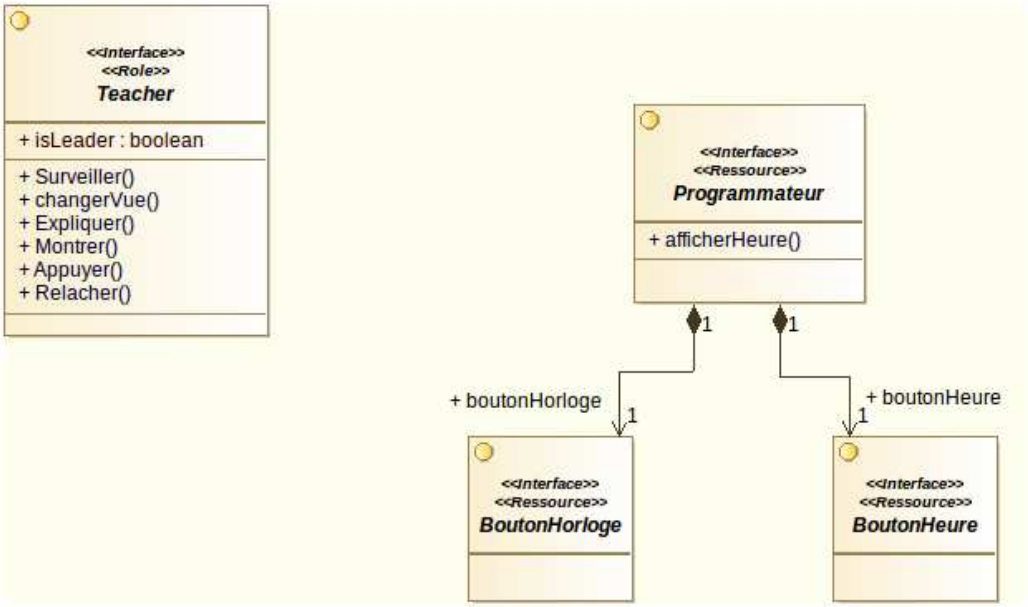

Figure 2.27 – Exemple de diagramme de classe représentant les rôles et les ressources

ressources. Les objets du scénario pédagogique sont donc représentés par ces ressources. Dans cet exemple, les ressources pédagogiques utilisées sont des ressources de l'environnement : `programmateur, boutonHeure, boutonHorloge, ecran`.

Les comportements décrits peuvent être généralisés dans des comportements haut niveau du tuteur. Par exemple une action `expliquerSysteme` qui effectue : `Expliquer, Montrer, ChangerVue` pour tous les éléments du système. Ce type de comportements haut niveau est également générique, car les actions qui le composent sont chacune réalisables sur les différents éléments de l'environnement. Ces acteurs sont également décrits dans le rôle.

2.3.3.2 Scénarios pour l'apprentissage de procédures

Le deuxième type de scénario que nous voulons décrire est un scénario pédagogique permettant l'apprentissage d'une procédure de maintenance ou d'utilisation, avec ses buts et ses actions. Le premier exemple concerne toujours le rôle « Teacher », mais celui-ci doit expliquer une procédure et la réaliser. Pour cela, l'agent « Teacher » réalisera l'action `ExpliquerAction` avant de réaliser chaque action (`appuyerBoutonHorloge`, etc.) de la procédure (figure 2.28). Ce scénario pédagogique permet à l'apprenant de comprendre ce qu'il fait et de voir ce qu'il faut vraiment faire. Dans ce cas, le formateur importe la procédure métier telle qu'elle est définie dans le modèle métier et il la modifie en insérant les actions pédagogiques souhaitées avant chaque action.

Un autre exemple permet de montrer un type de scénario plus avancé : nous avons un guidage effectué par le « Teacher » avant que l'apprenant réalise des actions. Nous aurons cette fois ci deux rôles : « Teacher » et « Learner ». Le « Teacher » progresse dans la procédure et ses actions lui permettent de guider l'apprenant en lui expliquant ce qu'il doit faire (figure 2.29).

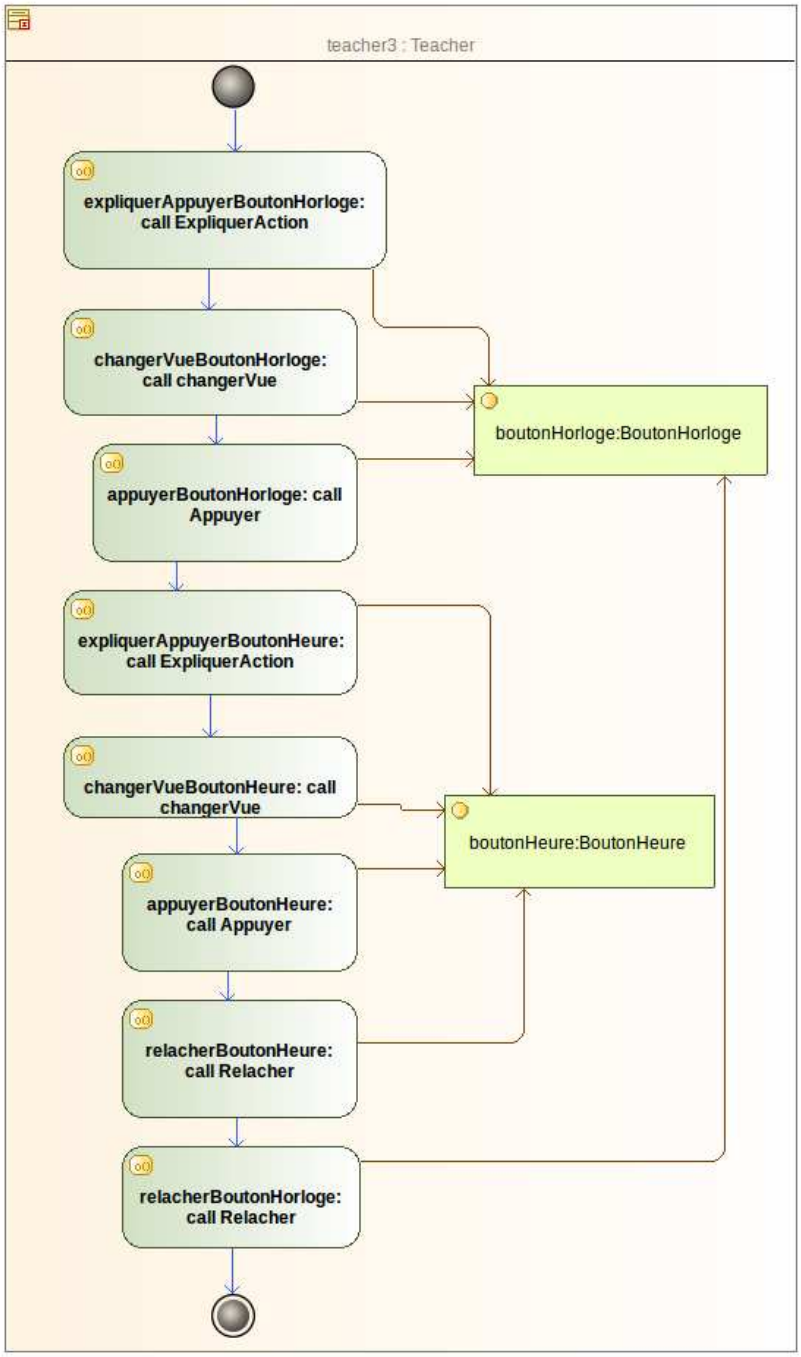

Figure 2.28 – Exemple de diagramme d'activité représentant un scénario pédagogique de présentation des objets de l'environnement et de leur utilisation

Le scénario pédagogique permet de mettre en évidence l'objet de l'action que doit réaliser l'apprenant. Pour cela, l'agent « Teacher » réalise d'abord l'action `expliquerBoutonHorloge`, puis l'agent « Learner » réalise l'action `appuyerBoutonHorloge`, puis la même chose pour le bouton Heure, et pour finir l'apprenant relache les deux boutons. Dans cet exemple, les ressources pédagogiques utilisées sont des ressources de l'environnement : `boutonHeure`, `boutonHorloge`, ainsi que la ressource externe `noticeProgrammateur`.

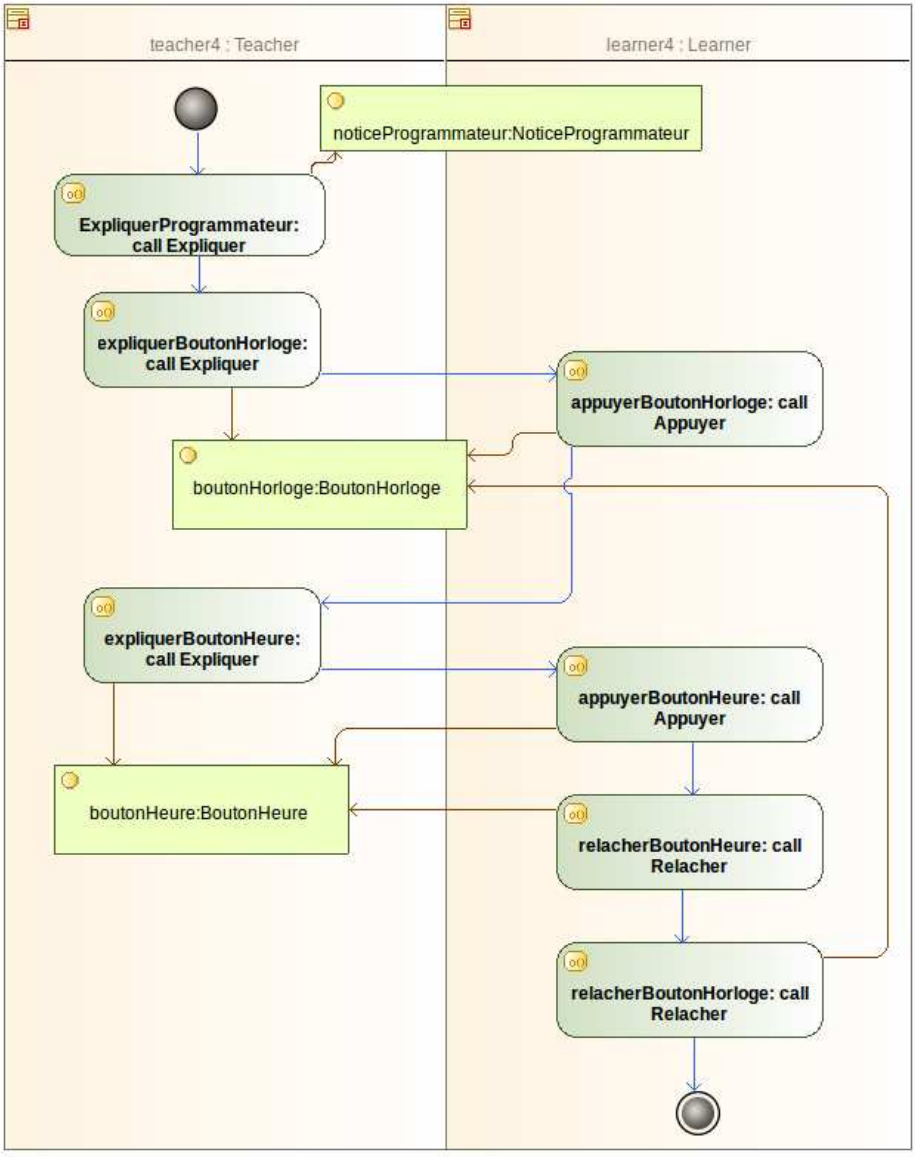

Figure 2.29 – Exemple de diagramme d'activité représentant un scénario pédagogique d'explication des objets de l'environnement avec une procédure métier

Un autre exemple permet de mieux évaluer les différentes possibilités de scénarios pédagogiques. Pour cela nous proposons d'utiliser la notion d'`Observables` décrits par Marion [2010]. Dans cet exemple (figure 2.30) nous avons un observable de type `TimeEvent` qui se déclenchera suivant une condition (*e.g.* temps d'exécution supérieur à 30 secondes).

Concrètement, si avant d'appuyer sur le bouton heure le temps est supérieur à 30 secondes, alors le `TimeEvent` se déclenchera, et l'action `afficherTimeOut` décrivant l'opération `Expliquer()` sera réalisée par l'agent « Teacher ». Ceci illustre que grâce à Chrysaor, dans lequel nous considérons un scénario pédagogique comme une activité, nous pouvons utiliser tous les concepts décrivants une activité pour décrire un scénario pédagogique (`TimeEvent, sendSignal, callBehavior`).

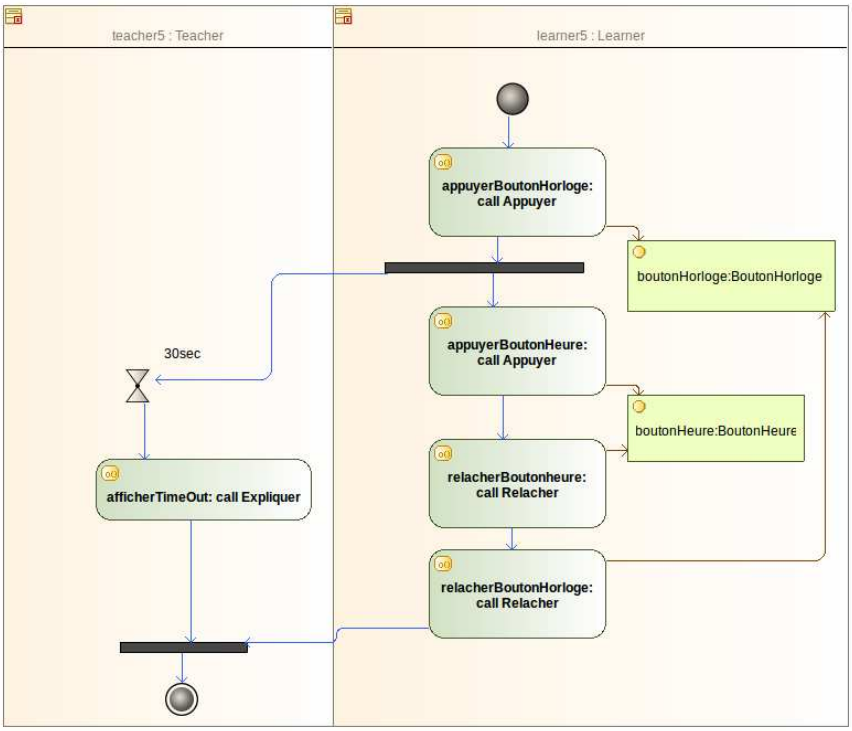

Figure 2.30 – Exemple de diagramme d'activité représentant un scénario pédagogique avec un observable sur l'environnement dans une procédure métier

Selon le même principe que précédemment il est possible de décrire des comportements plus haut niveau et génériques. Par exemple une action qui permet à l'agent « Teacher » de surveiller l'apprenant. Pour cela, le « Teacher » réalise des actions en parallèle de celles de l'apprenant (figure 2.31). Le rôle « Teacher » décrit une action `surveiller` qui sera réalisée par un agent. Cet agent sera chargé de surveiller les actions de l'apprenant et de proposer des aides en cas de mauvaise action. Selon le même principe, nous pouvons fournir un comportement `expliquerProcedure` qui permettra d'expliquer l'ensemble de la procédure, y compris son but. Ceci est faisable car la procédure est une connaissance de l'environnement. Par conséquent, le comportement pourra raisonner sur l'ensemble des actions et leurs enchaînements.

Dans ces exemples, les actions des scénarios pédagogiques sont représentées par une opération du rôle. Cependant, ces actions peuvent également être décrites par un diagramme d'activité. En effet, dans notre exemple 2.31 nous avions l'action `surveiller`. Celle-ci peut correspondre à un comportement complexe de STI qui sera lui même décrit par un diagramme d'activité. Dans la section suivante nous allons donc présenter des exemples de STI pouvant jouer des rôles dans des scénarios pédagogiques.

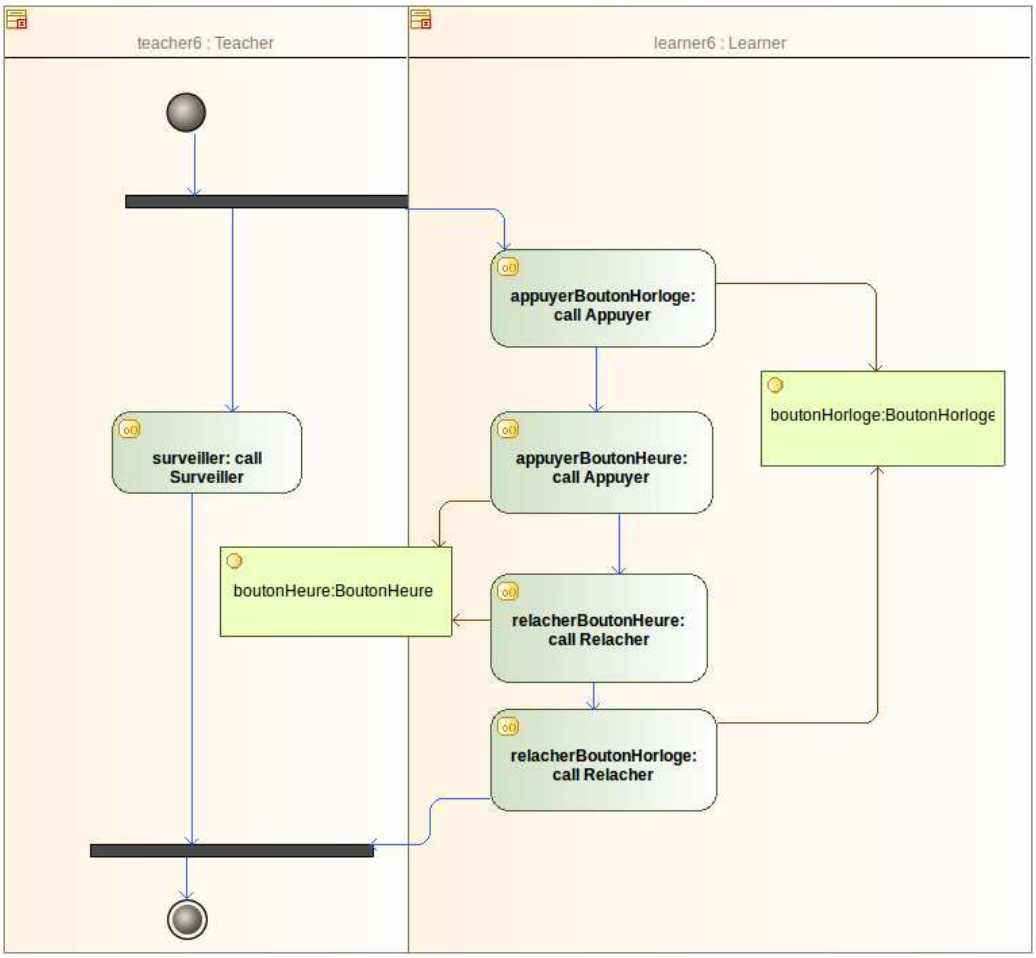

Figure 2.31 – Exemple de diagramme d'activité représentant un scénario pédagogique avec un comportement de haut niveau de surveillance d'une procédure métier

2.4 Système Tutoriel Intelligent

2.4.1 Modularité

Pour les raisons expliquées dans la section 1, nous avons choisi de baser notre STI (Chrysaor) sur Pegase. Le comportement global de Pegase est constitué de cinq phases (*teacher, learner, error, compare, pedagogical, interface, expert*) qui ont chacune leur propre implémentation, et l'ensemble est exécuté de manière autonome et séquentielle (figure 2.32).

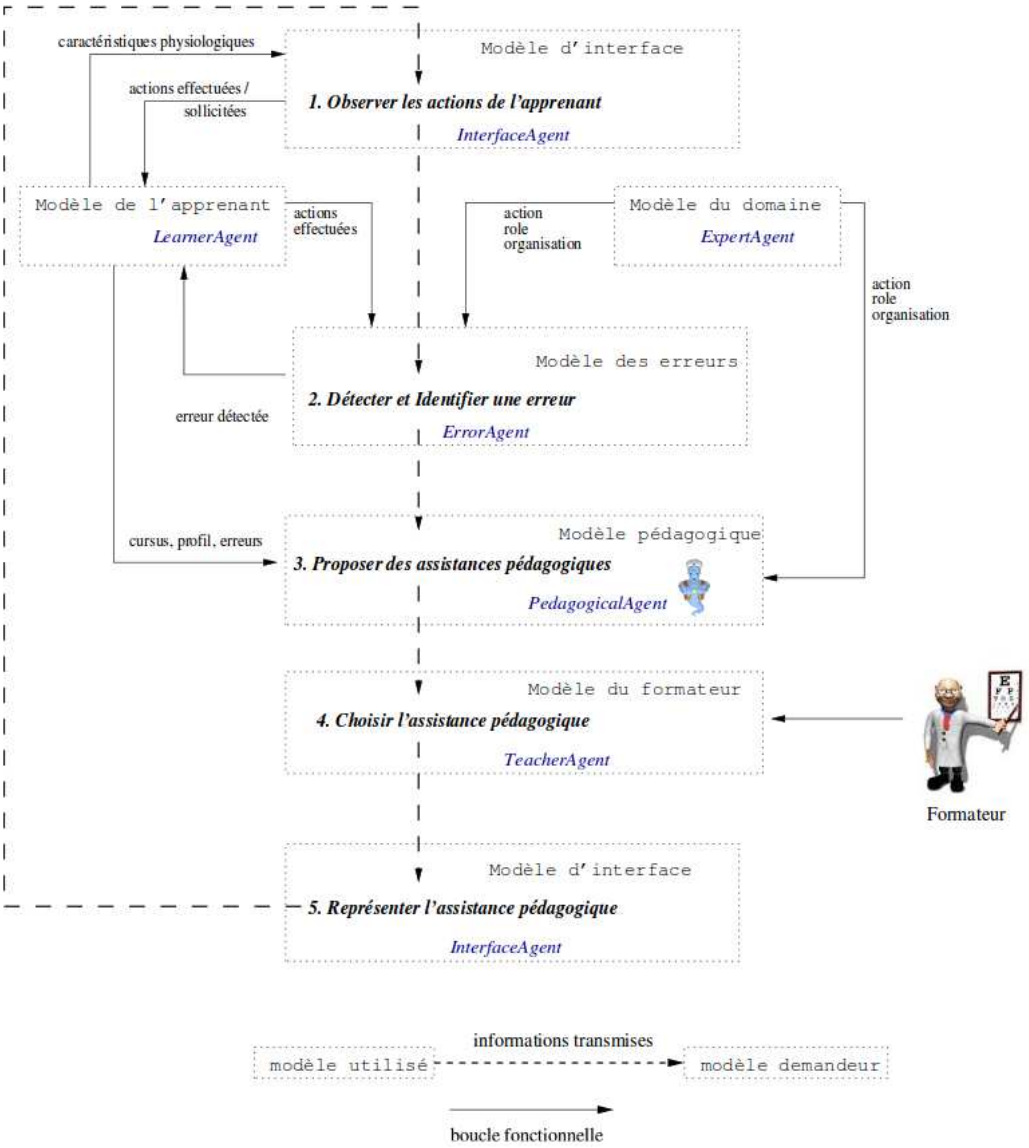

Figure 2.32 – Processus pédagogique de Pegase constitué de cinq étapes

L'inconvénient de cette structuration est que pour réaliser un comportement différemment (*e.g.* avoir un comportement pédagogique qui raisonne d'une autre façon), il faut repenser

le système dans sa globalité. De même, si le formateur souhaite qu'un comportement soit réalisé par un humain à la place d'un processus informatique, dans l'état actuel des choses ceci est impossible avec Pegase. Les concepts proposés dans la section 2.2 nous permettent de pallier ces problèmes. En effet, nous pouvons décrire nos comportements de STI par un diagramme d'activité grâce aux concepts de rôle et d'organisation. Des agents autonomes ou humains se chargeront de réaliser ces rôles. Le formateur pourra ainsi très facilement modifier un comportement, l'enchaînement des comportements ou l'affectation des rôles. Il y a un gain important en modularité.

Nous proposons donc de décrire le modèle de Pegase en utilisant une organisation et un diagramme d'activité, de la même manière que pour le scénario pédagogique (figure 2.33). Ce modèle correspond ainsi à Chrysaor.

Cette organisation est relativement complexe, mais contrairement à Pegase le pédagogue peut aisément modifier les comportements, les rôles et les affectations (humain ou agent autonome). Comme nous l'avions indiqué précédemment, cette organisation pourra jouer un rôle dans une autre organisation (*e.g.* Chrysaor jouera le rôle « Teacher » dans un scénario pédagogique).

2.4.2 Exemples d'utilisation

Ce type de modélisations pour Chrysaor nous apporte énormément de modularité. Nous pouvons ainsi créer d'autres STI aux comportements plus ou moins complexes que nous allons présenter grâce à quelques exemples. Dans la figure 2.34 le formateur décrit le rôle « Learner » qui doit réaliser une procédure de mise à l'heure du programmateur. Le « Teacher » surveille l'environnement (`ObserverEV`) et en cas d'erreur il stocke l'information (`StockerErreur`).

L'exemple de la figure 2.35 est un peu plus évolué, cette fois ci en plus de stocker l'information, le « Teacher » signale l'erreur (`MiseEvidenceErreur` en changeant par exemple la couleur du bouton en rouge), ce qui a pour but d'informer l'apprenant de son erreur.

L'exemple de la figure 2.36 permet quant à lui de faire un `UndoErreur` sur l'erreur effectuée, cela permet à l'apprenant de revenir dans le bon état du système pour réaliser la suite de la procédure.

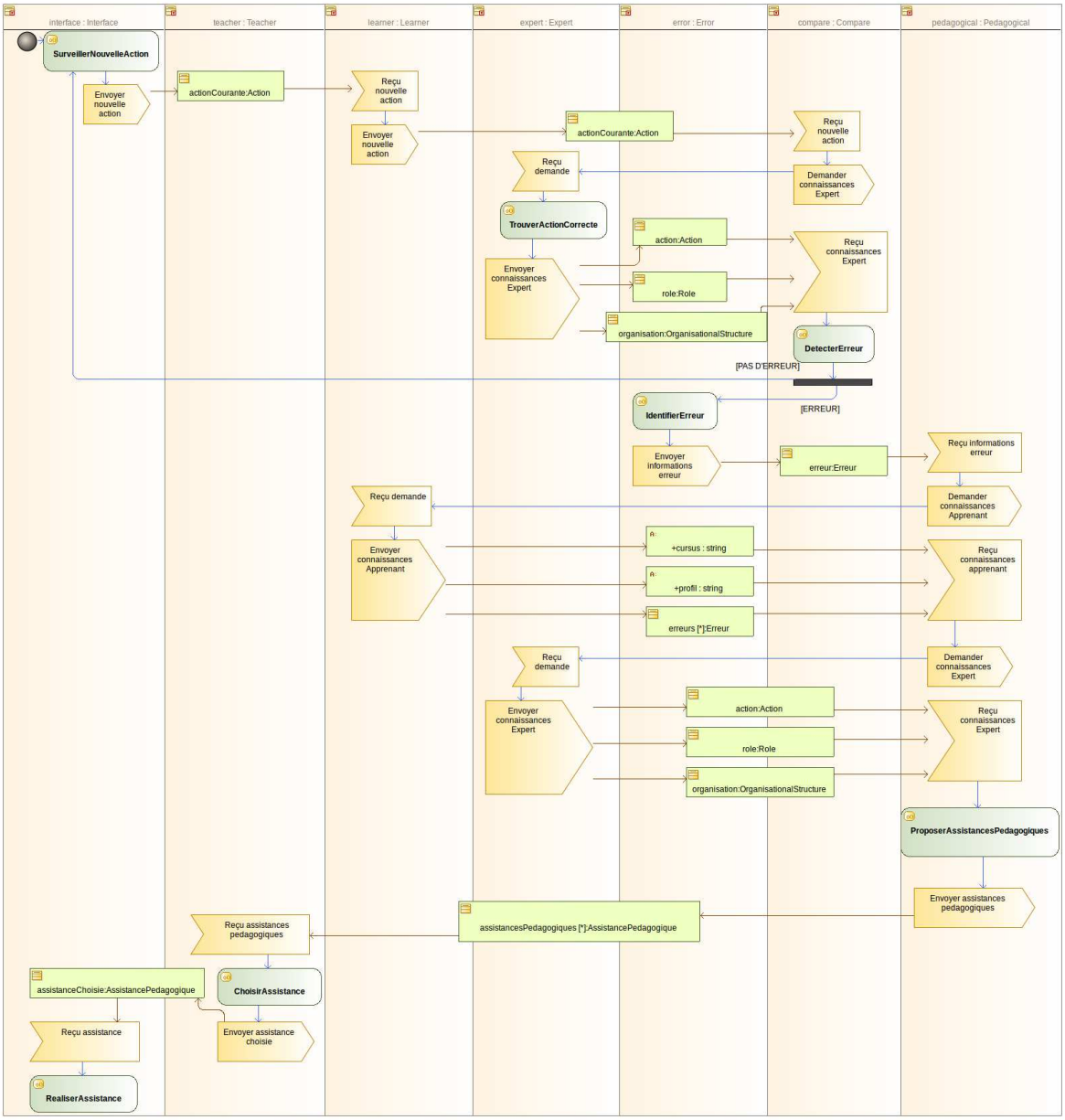

Figure 2.33 – Diagramme d'activité décrivant le comportement de Chrysaor

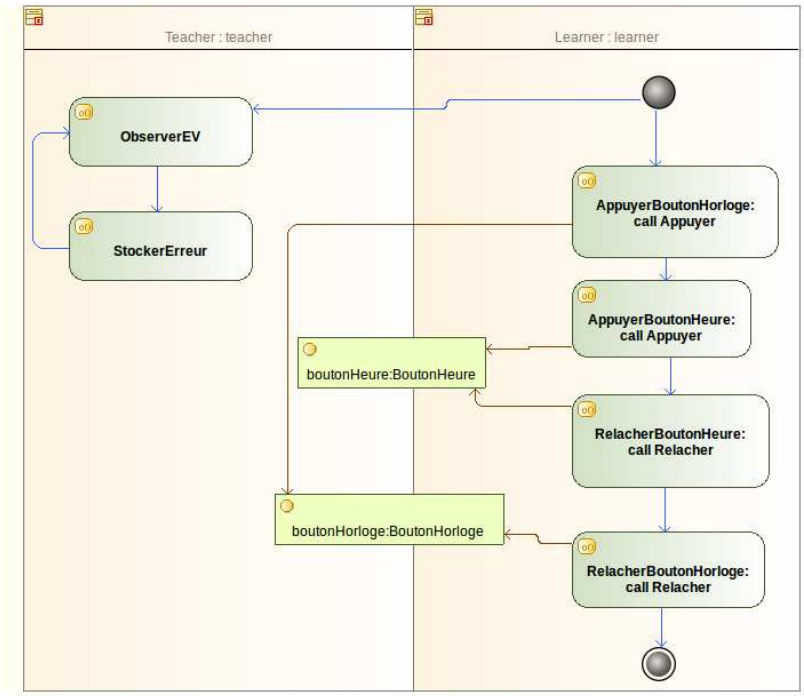

Figure 2.34 – Exemple de diagramme d'activité d'un comportement de STI composé de deux actions

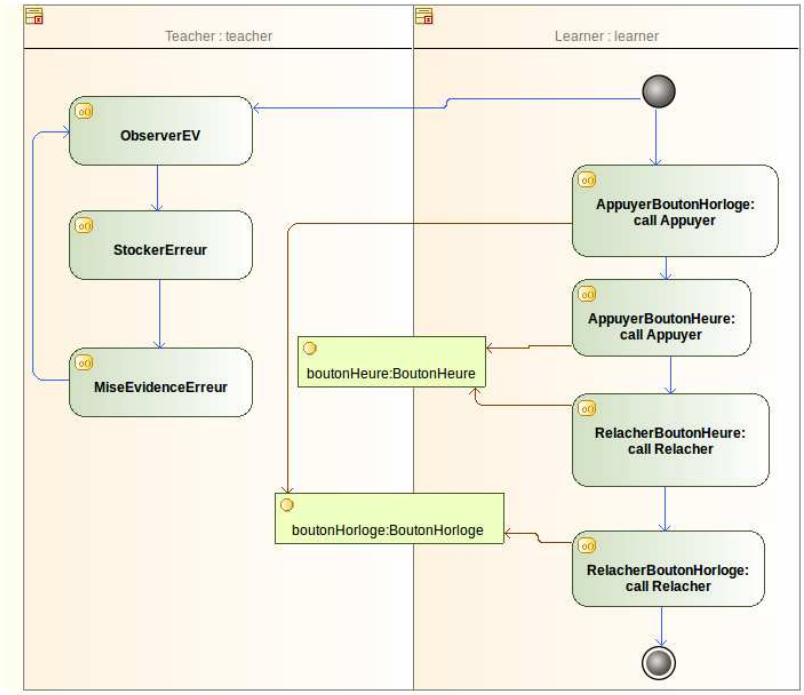

Figure 2.35 – Exemple de diagramme d'activité d'un comportement de STI composé de trois actions

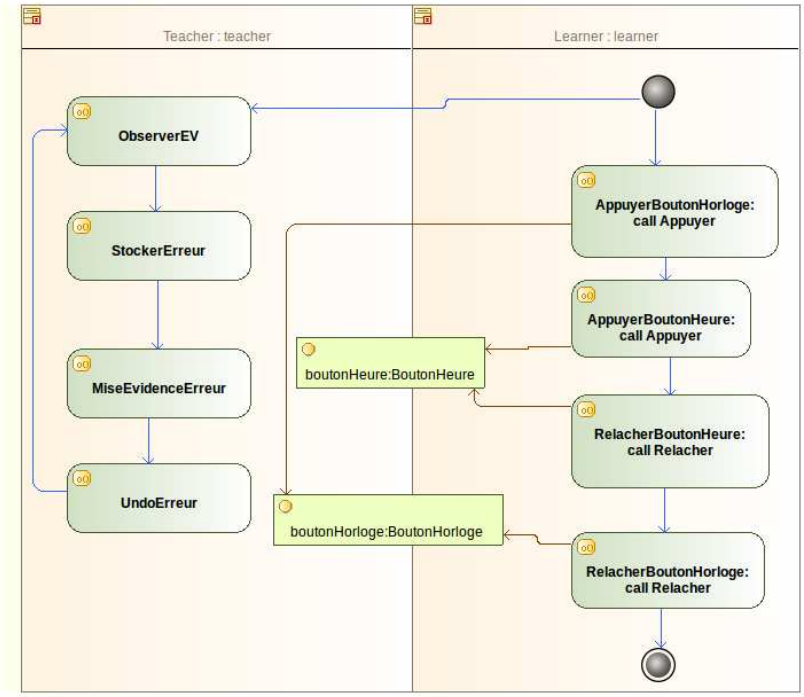

Figure 2.36 – Exemple de diagramme d'activité d'un comportement de STI composé de quatre actions

2.5 Synthèse

Dans le chapitre 1 nous avons identifié les principales faiblesses de Pegase qui sont la modularité et le manque de liens avec les scénarios pédagogiques. Pour combler ces faiblesses, les concepts clés sur lesquels nous devons axer nos travaux sont les rôles, organisations, comportements, bases de connaissance et activités.

Il était nécessaire d'identifier ces concepts pour ensuite nous permettre de les formaliser. De manière simplifiée nous obtenons que :
- une ressource peut être une entité dans une organisation métier, ou une ressource pédagogique dans un scénario,
- un rôle regroupe un ensemble de comportements, de services ou d'actions et peut avoir des objectifs à atteindre,
- une organisation est composée de plusieurs rôles et possède également un objectif,
- une activité permet d'organiser des actions, autour de plusieurs rôles et elle possède un but.

Nous souhaitons réifier ces concepts pour les rendre manipulables par notre STI. Le méta-modèle Uml propose des concepts couvrant certains critères que nous avons décrits, c'est pourquoi nous avons formalisé certaines notions d'Uml afin d'étendre notre propre modèle. Le concept de `UML::Interface` se rapproche de notre description de rôles, c'est pourquoi nous faisons hériter notre `Mascaret::Role` d'un `UML::Interface`. Un

agent pourra jouer un rôle dans une organisation car il aura déclaré un rôle comme étant réalisé grâce au concept de `UML::InterfaceRealization`. Uml propose également le concept de `UML::Collaboration` qui correspond tout à fait à notre définition qu'une organisation peut être composée de plusieurs rôles. Nous faisons ainsi hériter `Mascaret::OrganisationalStructure` de `UML::Collaboration`. L'implémentation de cette collaboration est une `UML::CollaborationUse` à partir de laquelle nous faisons hériter `Mascaret::OrganisationalEntity` qui est l'implémentation de la structure organisationnelle de Mascaret. Enfin, les critères que nous avons identifiés pour décrire une activité sont couverts par le concept de `UML::Activity`. Une `Mascaret::Activity` hérite donc de `UML::Activity`.

Mascaret définit dans son modèle le concept d'agent. Ces agents jouent des rôles définis dans les organisations. Afin de pouvoir raisonner sur la simulation, les agents ont besoin de connaissances sur l'environnement. C'est pourquoi la base de connaissances d'un agent Mascaret référence un `Mascaret::Environment`. Nous avons doté les agents de moyens de communication en utilisant le standard de communication agent : *FIPA-ACL*. Nous proposons également deux comportements automatiques à nos agents. Tout d'abord le comportement de communication qui vérifie et traite les messages reçus par l'agent, comme par exemple gérer une demande de valeur d'un attribut (l'agent qui reçoit le message ayant accès à cette connaissance). Il y a également le comportement procédural qui permettra à l'agent d'avoir son propre suivi de procédure.

Ces concepts, ainsi formalisés dans Mascaret, sont utilisables afin de définir ce qu'est un scénario pédagogique au sein de Mascaret. Afin que Chrysaor puisse raisonner dessus, nous proposons que le scénario pédagogique soit une connaissance au même titre que l'environnement de simulation. Le scénario pédagogique sera donc défini par un `Mascaret::Model` et un `Mascaret::Environment`. Dans un premier temps nous nous sommes positionnés vis-à-vis de la référence en tant que langage de modélisation pédagogique : IMS-LD. En effet, nous avons montré qu'il est possible à partir d'un scénario défini en Chrysaor, de générer un scénario en IMS-LD mais que l'inverse ne l'est pas.

Notre modèle étant bien défini, nous proposons d'adapter le modèle de Pegase en le décrivant de la même manière que pour le scénario pédagogique. Cette organisation correspond ainsi à Chrysaor. Quelques exemples de comportements plus ou moins évolués de Chrysaor sont également proposés afin de montrer la modularité de notre STI.

Dans la section suivante, nous allons présenter notre application de réalité virtuelle basée sur un instrument Stago ainsi que les expérimentations que nous avons menées.

Notre modèle a fait l'objet de plusieurs publications, notamment [Querrec et al., 2011; Le Corre et al., 2012].

Chapitre 3

Application et résultats

Dans la section précédente nous avons présenté notre proposition de modèle ayant pour but d'améliorer la modularité de notre STI, ainsi que la manière dont nous sommes parvenus à coupler notre STI à un scénario pédagogique. Le modèle que nous proposons permet de résoudre la problématique de formation que nous avons présentée en introduction. Nous montrons dans ce chapitre comment nous appliquons notre modèle à la conception d'une formation en réalité virtuelle chez Stago. Dans la section 3.1 nous présenterons le STA-R® : instrument de diagnostic en hémostase de la gamme Stago, choisi pour notre simulateur ainsi que l'application de réalité virtuelle correspondante. En section 3.2 nous détaillerons les scénarios pédagogiques utilisés ainsi que le fonctionnement du STI, au sein de notre application. Pour terminer, en section 3.3 nous présenterons les expérimentations menées sur cette application et les résultats associés.

3.1 Simulateur

Dans un premier temps nous allons présenter les composants principaux du STA-R® ainsi que son fonctionnement (section 3.1.1). Nous présenterons ensuite notre modélisation et notre application nommée StarApplication (section 3.1.2).

3.1.1 Instrument biomédical : STA-R®

Parmi la gamme de produits proposés par Stago nous avons identifié l'instrument haut de gamme : le STA-R® (figure 3.1). Il s'agit d'un automate de coagulation qui permet la réalisation simultanée des tests chronométriques, colorimétriques et immunologiques. Cet instrument travaille de façon autonome ou peut être intégré dans une chaîne automatisée (laboratoires importants ou regroupement de plus petits laboratoires).

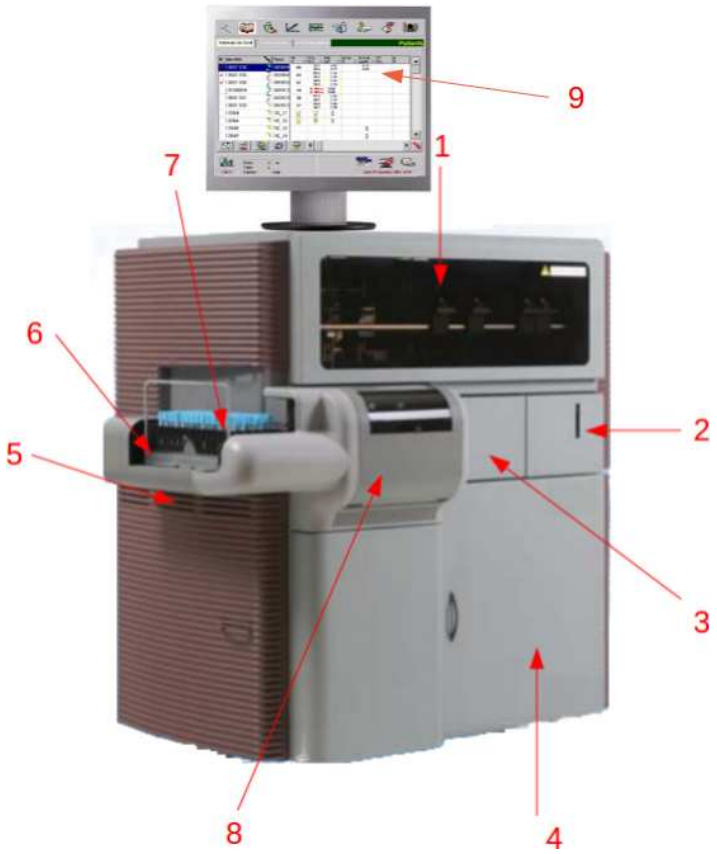

Figure 3.1 – Un instrument de diagnostic en hémostase : le STA-R®

Référence	Description	Détails
1	Capot hayon	Il s'agit d'un capot de protection translucide. Grâce à celui-ci nous pouvons facilement observer le déroulement d'un test. Derrière se trouve tout le système de prélèvements et de tests.
2	Lecteur de code à barre	Il est utilisé afin de scanner les réactifs.
3	Tiroir produits	Zone de chargement des réactifs scannés.
4	Porte côté bas droit	Les poubelles biologiques se trouvent derrière cette porte.
5	Porte côté bas gauche	Nous pouvons y trouver tout le système fluidique (pour le fonctionnement des pompes).
6	Panier	Plusieurs racks peuvent être placés sur le panier.
7	Rack	Un rack se positionne dans le panier et peut contenir 5 échantillons patients.
8	Porte lecteur de code à barre patients	Derrière la vitre se trouve le lecteur qui permet de scanner les échantillons patients à leur chargement.
9	Écran tactile	Permet à l'utilisateur de contrôler l'instrument.

3.1.1.1 Description

Afin de mieux comprendre le fonctionnement de cet instrument, voici le principe simplifié de son utilisation. Le but du STA-R® est de tester un échantillon de sang humain afin de vérifier s'il y a un problème au niveau de la coagulation sanguine. Cet échantillon de sang est stocké dans un tube échantillon (figure 3.2).

Figure 3.2 – Photos de tubes échantillons vides ou contenant du sang

Pour réaliser les tests de coagulation, le STA-R® nécessite l'utilisation de réactifs (figure 3.3) qui contiendront les produits pouvant déclencher certaines réactions biochimiques au contact du sang. Il existe de nombreux réactifs différents en fonction du test que l'on souhaite effectuer. Dans notre simulation nous avons un total de dix flacons différents. L'utilisateur a également accès à des pipettes (pour prélever un liquide), des aimants (pour l'agitation magnétique), des reducers (pour diminuer l'étranglement du flacon).

Figure 3.3 – Ensemble de réactifs commercialisés par Stago

Le choix du ou des tests à réaliser se fait *via* l'interface logicielle présente sur un ordinateur séparé (figure 3.4). Bien évidemment plusieurs autres composants entrent en compte dans le fonctionnement du STA-R® (*e.g.* les cuvettes, les reducers, etc.), mais les détailler n'a que peu d'intérêt dans le cadre de cette thèse.

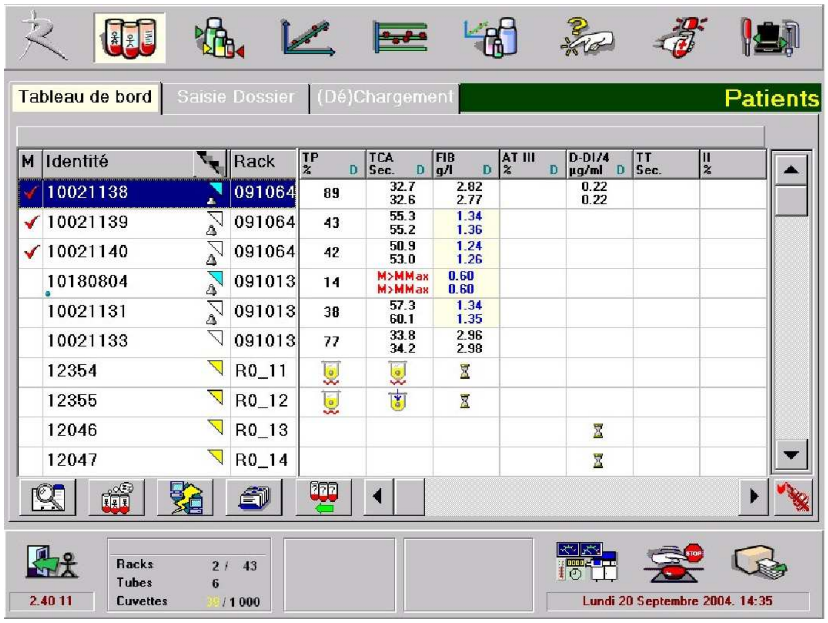

Figure 3.4 – Image issue du logiciel de pilotage du STA-R®

3.1.1.2 Tests de coagulation

Le STA-R® prélève une partie du sang du patient (contenu dans un tube échantillon) grâce à ses bras pipeteurs, pour ensuite y ajouter différents réactifs (dépendants du ou des tests souhaités par le médecin). Ensuite ce mélange sera incubé à 37°C (température du corps humain) et les mesures du tests pourront se faire selon plusieurs méthodes :
- méthode de coagulation : dosage chronométrique (mesure du temps),
- méthode à l'aide d'un substrat biochimique : dosage colorimétrique (mesure d'un changement d'intensité lumineuse du à la coloration),
- méthode immunologique : un anticorps qui se fixe sur la protéine à doser (antigène).

Les résultats des mesures seront visualisés sur l'interface logicielle tactile du STA-R®.

Nous présentons ici trois tests réalisables sur le STA-R® : TP, TCA, FIB. Ces tests figurent parmi les plus couramment utilisés en routine dans les laboratoires.

Le **TP**, ou taux de prothrombine est le test le plus simple et le plus souvent le premier effectué dans le cadre du diagnostic d'anomalies de coagulation. Ce test explore la voie extrinsèque de la coagulation. Ce test mesure un temps de coagulation, ce temps étant rallongé en cas de trouble de la coagulation.

Le **TCA**, ou temps de céphaline activée est un test semi-global qui explore la voie intrinsèque de la coagulation. Ce test est généralement effectué dans un premier temps avec le TP. Ce test mesure également un temps de coagulation qui est également rallongé en cas de trouble de la coagulation. Cependant, contrairement au TP, ce test se concentre sur d'autres facteurs de la coagulation (protéines du plasma sanguin). En général, la combinaison d'un TP et d'un TCA permettent de détecter une grand partie des troubles de coagulation chez les patients.

Le **FIB**, ou taux de fibrinogène reflète la capacité du sang à produire un caillot. En général, ce test est réalisé quand les résultats du TP ou du TCA sont anormaux. Le principe de ce test est de mesurer la quantité de fibrinogène (protéine du plasma sanguin qui intervient dans la formation d'un caillot) qui évolue au cours de la coagulation.

3.1.1.3 Bilan

Le STA-R® est l'automate de coagulation commercialisé par Stago sur lequel nous avons choisi de baser notre application. Nous avons pu comprendre son mode de fonctionnement grâce à cette brève description de sa structure, de son utilisation, ainsi que des tests biologiques pouvant y être réalisés. Cela nous permettra de réaliser un modèle d'instrument de diagnostic, pour notre application en réalité virtuelle, qui est présenté dans la section suivante.

3.1.2 STA-R® virtuel

Nous avons donc choisi de baser notre application appelée StarApplication sur l'instrument de diagnostic *in vitro* STA-R®. Nous avons séparé notre application en deux systèmes de la même manière que dans la réalité (figure 3.5) :
- StarApplication : l'application 3D représentant l'instrument STA-R® réel,
- StarDroid : une interface tactile sur tablette qui simule le logiciel qui s'exécute sur l'ordinateur associé au STA-R® réel (sur un écran tactile).

Figure 3.5 – Bilan des deux systèmes du STA-R® réel

Nous décrivons la structure et le comportement du STA-R® en UML (classes, états, activités) selon le profil Mascaret. Ceci permet à ce modèle d'être manipulé par le scénario pédagogique et de servir de base de connaissances au STI. La description du modèle du STA-R® est structurée selon trois packages :
- Environnement : le dispositif,
- Agent : les actions des utilisateurs,
- Organisation : les procédures métier.

Nous allons présenter le contenu de ces trois parties du modèle dans les sections 3.1.2.1, 3.1.2.2 et 3.1.2.3. Nous détaillerons également l'interface que nous avons développée sous Android (StarDroid) et sa façon de communiquer avec l'application (StarApplication) dans la section 3.1.2.5.

3.1.2.1 Environment

Dans un premier temps nous allons détailler la façon de décrire l'environnement afin que celui-ci soit interprété par le méta-modèle Mascaret. Au sein du `package Environnement` nous allons décrire les entités qui représentent chacune une partie ou sous-partie du STA-R®. Nous avons choisi de représenter les objets par sous-partie de l'instrument et non pas par toutes les pièces qui le composent, car notre niveau de détails se situe justement vis à vis des actions et des mouvements possibles de ces sous-parties. Une modélisation complète de chacune des pièces auraient demandé beaucoup plus de temps.

`Structure`
Pour décrire les entités, nous les avons également segmentées en plusieurs ensembles en fonction de leurs localisations ou de leurs rôles. Par exemple dans le package `Instrument` nous pouvons retrouver différents packages comme par exemple `BlocMVTS`, qui contient un `SystemeMVTS` composé entre autre de trois `BrasAiguille` lui même composé d'un `Bras` et d'une `Tete`.

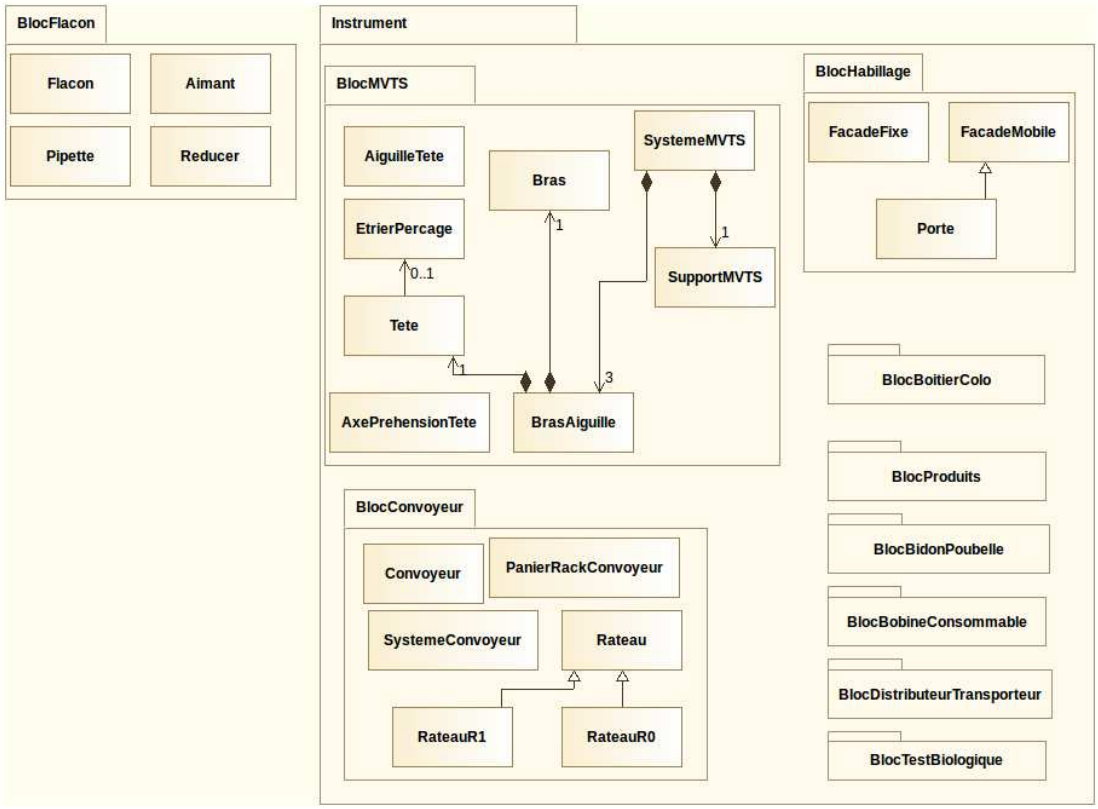

Figure 3.6 – Diagramme de classe des entités de l'environnement du STA-R®

Si nous prenons en exemple le package `BlocHabillage`, nous pouvons remarquer que celui-ci contient plusieurs classes telles que `Facade` ou `Porte`. Dans notre modèle nous avons un total de 66 entités (stéréotype de Mascaret) décrites.

`Attributs`
Nous remarquons que la classe `Porte` contient une description classique en Uml, c'est-à-dire les différents attributs que l'on pourrait retrouver dans cette classe : `angle, sens, coordonnées`, mais également des attributs de type `son` (`sonOuverture, sonFermeture`), qui permettent d'y associer un son (figure 3.7). Ce type est une extension d'Uml fournit par Mascaret pour tenir compte de la spécificité de la réalité virtuelle.

`Opérations`
Nous pouvons également décrire des opérations au sein des classes, comme par exemple `fermer()` ou `ouvrir()` pour la classe `Porte` (figure 3.7).

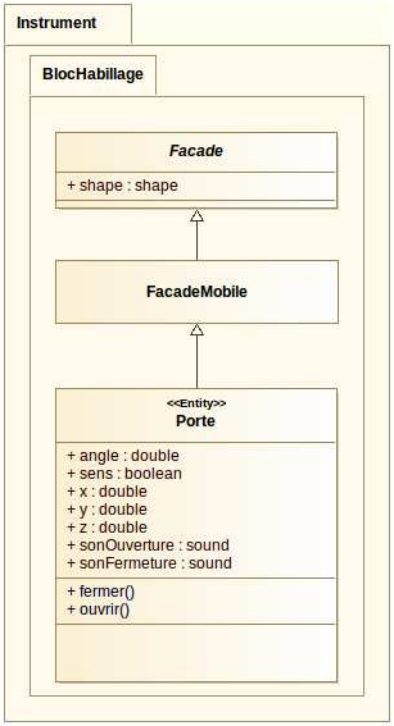

Figure 3.7 – Diagramme de classe de la classe `Porte` du STA-R®

Ces opérations seront interprétées par Mascaret et associées à du code écrit sous forme d'un `OpaqueBehavior`. Tous les `OpaqueBehavior` de la porte seront implémentés dans un plugin.

`Comportement des entités`
Dans Mascaret, les machines à états servent à décrire le comportement autonome, réactif et asynchrone des objets de l'environnement virtuel. Elles sont simulées automatiquement. Dans notre modèle nous pouvons décrire des signaux, tels que `OuvrirPorte` ou `FermerPorte`. Ces signaux reçus permettront entre autre à une entité de changer d'état dans son comportement. En effet, nous pouvons réaliser un diagramme de machine à états pour nos classes. Pour la

classe `Porte` nous avons défini quatre états : `Fermee`, `Ouverte`, `SOuvre` et `SeFerme`. Nous pouvons mettre des conditions pour un changement d'état, tel que un signal reçu (*e.g.* `OuvrirPorte`) (figure 3.8). Un lancement d'opération peut être associé à un état (Do). Cette opération sera exécutée tant que l'état sera actif. En effet, dans notre exemple, quand on transite vers l'état `SOuvre`, l'opération `ouvrir()` s'exécute. Dans notre modèle nous avons un total de 27 machines à états.

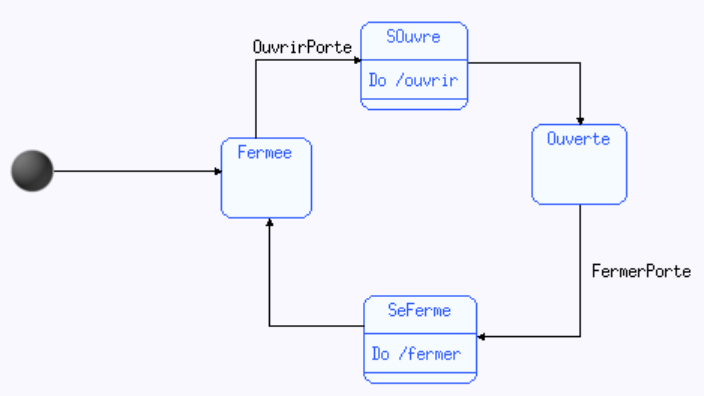

Figure 3.8 – Diagramme de la machine à état de la classe `Porte`

3.1.2.2 Agent

Nous allons maintenant détailler la façon de décrire les agents qui représentent les utilisateurs du système, c'est-à-dire l'ensemble des actions qu'ils peuvent faire. Au sein du `package Agent` nous allons donc décrire les agents qui pourraient prendre part à l'environnement virtuel. Le stéréotype de cette interface est de type « `Virtual Human` ».

`Classes`
Dans le package `Agent` nous devons définir des classes qui représentent les agents qui peuvent prendre part à l'environnement. Dans notre cas, nous avons une classe : `Utilisateur` qui représente l'apprenant qui peut interagir avec l'environnement.

`Opérations`
Au sein de ces classes, nous retrouvons des opérations (*e.g.* `OuvrirTiroirProduits`, `ScannerFlacon`...). Ces opérations représentent les actions que peut faire l'agent et portent sur les objets décrits dans le package `Environnement`. La figure 3.9 liste l'ensemble des actions de l'utilisateur sur le système.

Ces opérations peuvent être décrites par une activité.

`Activités`
Les opérations décrites dans la classe précédente sont réalisées par des activités figure 3.10 qui décrivent la réalisation de ces opérations. Une activité sera réalisée à l'appel de l'opération dont elle associée. Par exemple, dans le cas de l'opération `OuvrirTiroirProduits` qui est associée à l'activité `ActOuvrirTiroir`, l'appel de cette opération réalisera automatiquement l'activité décrite dans la figure 3.10. Dans cet exemple, cette activité consistera à envoyer un signal `OuvrirTiroirProduits` à l'entité `TiroirProduits`. Dans la section 3.1.2.1 nous avons

Figure 3.9 – Liste des actions de l'utilisateur sur le STA-R®

expliqué que ces signaux permettent de changer un état et ainsi réaliser une opération de cette entité.

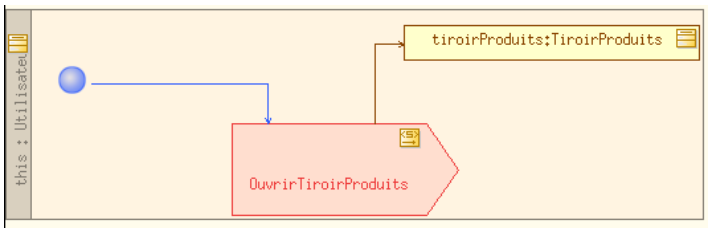

Figure 3.10 – Diagramme d'activité d'une activité du package `Agent` sous Modelio

3.1.2.3 Organisation

Pour terminer, nous allons détailler la façon de décrire les organisations. Au sein du `package Organisation` nous allons décrire les organisations afin de définir les procédures métier. Pour cela, il faut définir des `interfaces` qui contiennent les opérations possibles réalisables pour chacunes des interfaces. Par exemple, dans le cas de l'interface `Utilisateur`, nous pouvons voir toutes les opérations réalisables par la personne correspondante à cette interface (figure 3.11). Le stéréotype de cette interface est de type `Rôle`. Quand Mascaret interprète ces interfaces il les associe aux interfaces d'`Agent` portant le même nom.

Figure 3.11 – Description du rôle Utilisateur de l'organisation du fonctionnement du STAR®

Nous décrivons également une `collaboration` (que nous appelons `Fonctionnement`) afin de définir notre organisation. Dans cette collaboration, nous définissons des `rôles` (figure 3.12). Ces rôles peuvent être utilisés pour définir un rôle qui sera couplé à une interface de la partie Agent (qui est de stéréotype « `Virtual Human` »), mais également pour définir les différents objets présents dans les procédures. Par exemple, nous avons le rôle `flaconNeoL` qui est associé à une classe définie dans la partie environnement (en l'occurence la classe `Flacon`).

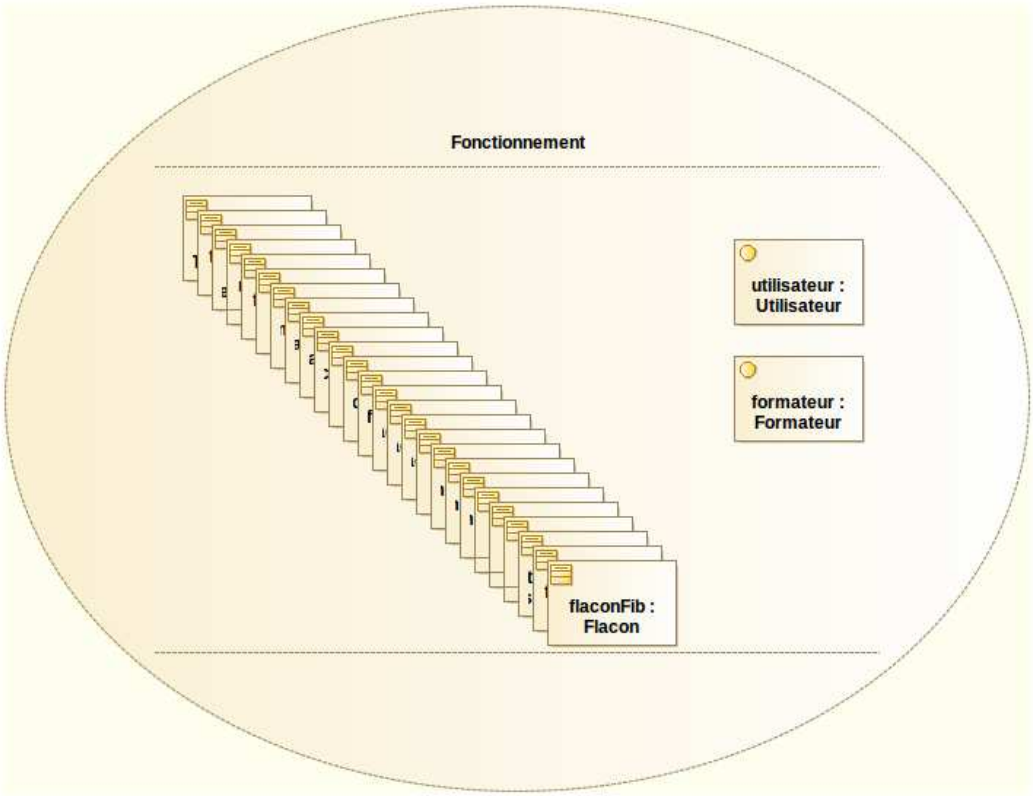

Figure 3.12 – Description de l'organisation du fonctionnement du STA-R®

Une fois les liens effectués entre les interfaces et les objets d'un côté et les classes de l'autre côté, nous pouvons définir différentes procédures métier de notre modèle du STA-R®. Par exemple, dans la figure 3.13 nous pouvons voir que nous définissons une `partition` (que nous avons à nouveau nommé `utilisateur` car elle est liée au rôle utilisateur de la collaboration). Au sein de cette partition nous pouvons définir les activités que l'agent ou la personne jouant le rôle pourra effectuer. Comme par exemple l'activité `PrendreFlaconNeoS`. Cette activité peut être associée à un objet défini également dans cette partition (tel que le `flaconNeoS`). Ces objets sont également liés aux rôles définis dans la collaboration. Notons que chacune des activités possède un lien sur l'opération définie dans l'interface `Utilisateur` qui se trouve à la racine de `Organisation`.

Dans la figure 3.13 nous pouvons voir le diagramme d'activité avec les partitions (en l'occurence nous n'avons qu'une seule partition ici) et l'agencement des actions dans cette activité. Nous pouvons remarquer que les activités sont associées aux objets sur lesquels portent les activités. Cette procédure de formation Stago est composée de 125 actions.

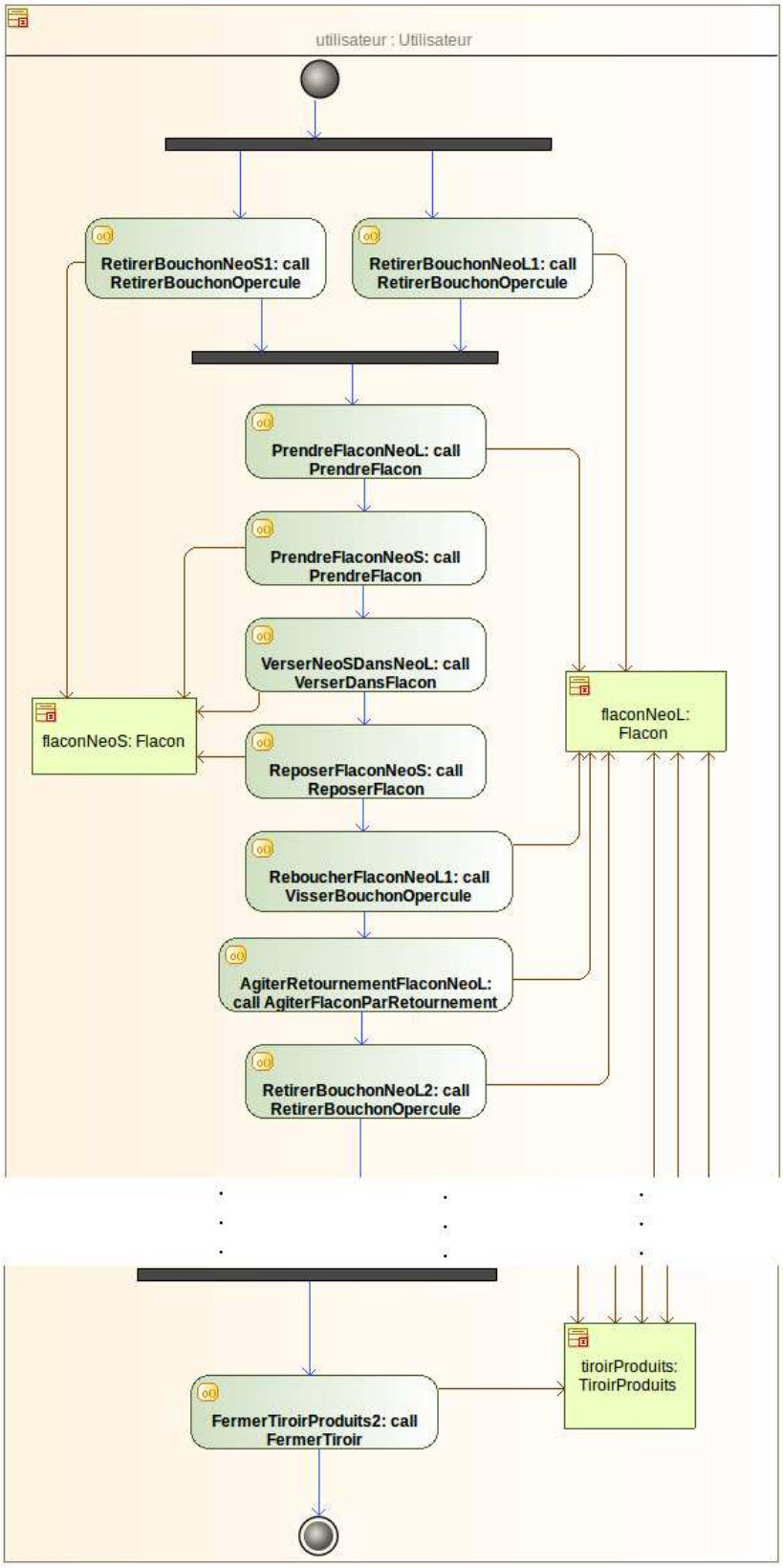

Figure 3.13 – Diagramme d'activité d'une procédure métier de formation dans StarApplication, au sein d'une organisation sous Modelio

3.1.2.4 StarApplication

Toutes ces descriptions nous ont permis de créer le modèle du STA-R® qui pourra ensuite être interprété par Mascaret. Cependant à ce stade nous avons les concepts de créés, mais aucune représentation visuelle. C'est pourquoi le méta-modèle Mascaret est associé avec une bibliothèque permettant de créer des applications de réalité virtuelle : ARéVi (*e.g.* Atelier de RÉalité VIrtuelle) Harrouet et al. [2006]. Plus précisément ARéVi est une bibliothèque de simulation d'entités autonomes et de rendu 3D développée au CERV.

Notre application de réalité vituelle (figure 3.14) est nommée StarApplication.

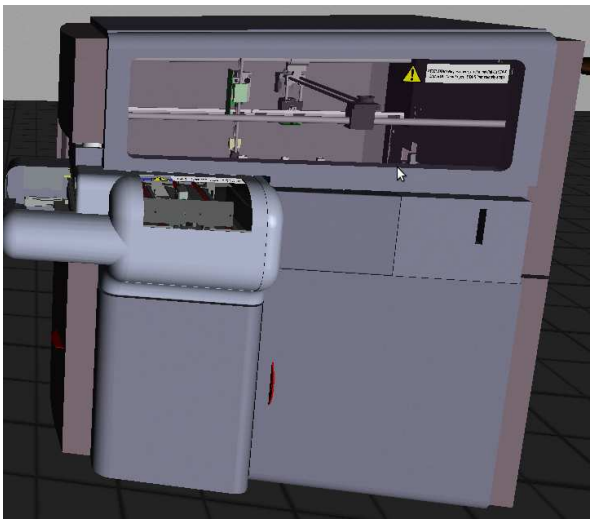

Figure 3.14 – Image de l'application du STA-R® en environnement virtuel

L'apprenant peut interagir avec l'environnement 3D en cliquant sur les objets présents. Pour cela, nous utilisons un `SignalWidget` qui est une action (`BasicAction`) de Mascaret. Ces `SignalWidget` permettent d'envoyer d'un simple clic les signaux associés à une entité, tout en passant des paramètres si nécessaire (figure 3.15) .

3.1.2.5 StarDroid

En section 3.1.1 nous avons vu que le STA-R® était composé de l'instrument et d'un ordinateur sur lequel s'exécutait le logiciel de pilotage. L'instrument a été modélisé en Uml et sera donc interprété par Mascaret pour générer l'application de réalité virtuelle. Les techniciens qui travaillent sur le STA-R® manipulent tout autant l'instrument que le logiciel, c'est pourquoi nous devons également réaliser une interface simulant le comportement du logiciel réel. Pour cela, nous avons choisi de réaliser notre interface sous Android (figure 3.16) afin que notre simulation de logiciel puisse s'exécuter sur une tablette tactile (rapprochement avec l'écran tactile de l'ordinateur du STA-R®). Nous l'avons nommée StarDroid.

`Communication`
StarDroid doit pouvoir communiquer avec l'application principale du STA-R®. En effet, notre interface Android doit connaître des informations de l'environnement pour pouvoir

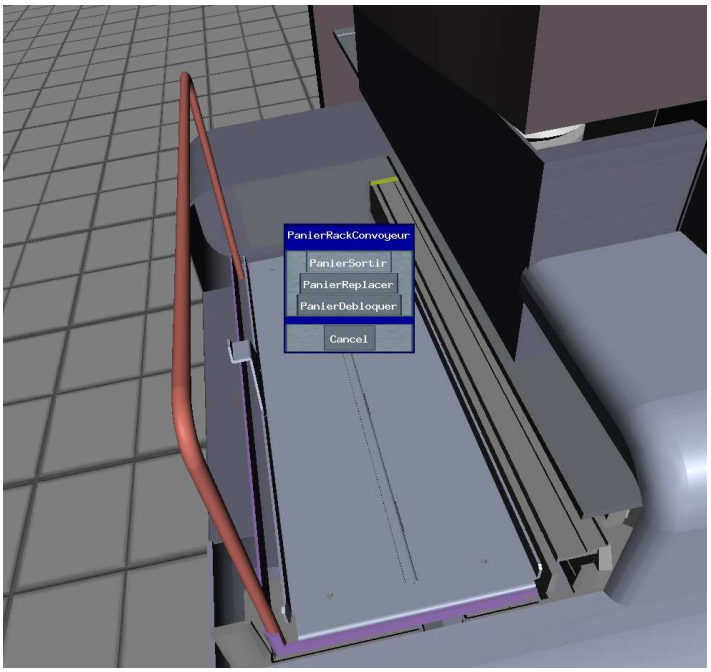

Figure 3.15 – Image de l'interaction utilisateur avec l'environnement

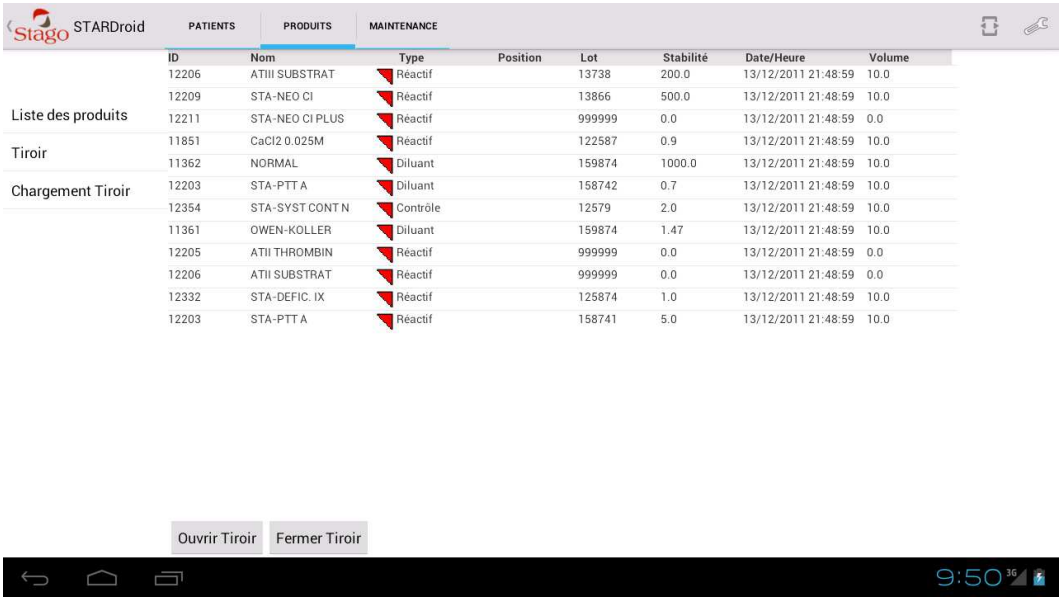

Figure 3.16 – Image de l'interface de StarApplication sous Android

les afficher (*e.g.* les réactifs chargés dans l'instrument), mais StarDroid doit également être capable d'envoyer un signal à l'application pour lui indiquer que l'utilisateur a interagi sur l'interface (*e.g.* ouverture du tiroir). Pour cela, nous avons créé une servlet `InstancesServlet` qui permet de simplifier l'échange de données entre l'application et l'interface Android. Cette servlet fonctionne de manière assez simple : elle permet de récupérer toutes les instances dans l'environnement avec en paramètre un type de classe. Ces données sont représentées sous le format XML que notre interface peut exploiter. Quand l'utilisateur fait une action sur l'interface Android, celle-ci envoie à l'application un message *FIPA* qui sera interprété par Mascaret.

3.1.2.6 Bilan

Dans cette section nous avons vu la manière d'écrire un modèle d'environnement pour que celui-ci soit interprété par Mascaret. Dans la section 3.1.2.1 nous avons expliqué de quelle façon écrire le modèle d'environnement avec les différentes classes ainsi que leurs attributs et leurs opérations, mais également les concepts de machines à états et de signaux. Dans la section 3.1.2.2 nous avons détaillé les différentes classes d'agents intervenant dans l'environnement ainsi que leurs opérations et les activités associées. Dans la section 3.1.2.3 nous avons explicité les différents concepts d'interfaces, de collaborations et des procédures métier. Finalement, en section 3.1.2.5 nous avons détaillé la réalisation de notre interface Android ainsi que la manière dont celle-ci communique avec l'application de réalité virtuelle.

Afin de mieux cerner les relations entre tous les packages du modèle, nous pouvons expliquer le cheminement entre la réalisation d'une action dans l'environnement et son impact sur celui-ci. Dans notre `Organisation` nous avons une activité qui correspond à la procédure métier. Quand une action de cette activité est réalisée (par un apprenant, par un programme extérieur, ou par Mascaret), il y a un appel de l'opération qui se trouve dans l'interface associée. Cette opération est du même nom que celle décrite dans le package `Agent`. Comme l'opération du package `Agent` est associée à une activité, c'est celle-ci qui est réalisée. Cette activité consiste majoritairement à l'envoi d'un signal sur une entité. Ce signal permet à l'entité de changer d'état et de réaliser une opération ou une activité propre à cette entité. La réalisation de cette opération consistera à l'exécution d'un `OpaqueBehavior` codé en dehors du modèle.

Cette segmentation entre Agent, Environnement et Organisation nous permet de facilement ajouter, supprimer ou modifier des éléments de notre modèle. Par exemple nous pourrions ajouter un nouveau comportement à un objet de l'environnement, ou modifier une procédure métier. De plus, la syntaxe employée pour décrire les modèles interprétés par Mascaret est compatible SysML (Systems Modeling Language). C'est-à-dire que nous pouvons facilement importer un modèle en SysML, et ainsi simplifier la conception générale des futurs modèles. Notons que SysML est très utilisé en industrie, ce qui est le cas pour la création des nouveaux instruments chez Stago.

3.2 Atelier d'ingénierie pédagogique

Dans le chapitre 2, nous avons donné quelques exemples de scénarios pédagogiques. Nous allons maintenant présenter des scénarios spécifiques à notre application StarApplication. Avec l'aide d'un spécialiste en sciences cognitives, nous avons identifié plusieurs étapes dans le processus de formation par la réalité virtuelle :
– présentation des composants du système,
– prise en main de l'application,
– tâche métier à réaliser avec un guidage fort,
– tâche métier à réaliser avec un comportement complexe (basé sur Pegase).
Nous proposons donc un scénario pour chacune de ces étapes. Ces scénarios seront utilisés afin de réaliser des expérimentations, sur des apprenants, qui seront présentées en section 3.3.

3.2.1 Scénario de présentation des composants

Dans un premier temps, nous proposons un scénario correspondant à ce que présentent les formateurs Stago au cours de leurs formations traditionnelles sur un instrument réel (figure 3.17). En effet, dans ces formations, le formateur présente individuellement chaque composant de l'instrument afin que les apprenants puissent les identifier et comprendre leur fonctionnement. Ce scénario consiste à changer de vue sur le composant (grâce à l'opération `ChangerVue`), puis à l'identifier en réalisant l'opération `Montrer()`, et enfin expliquer chacun des composants. L'opération `Expliquer()` se fera grâce aux informations associées à chaque partie ; par exemple dans le cadre du tiroir produits, l'information fournie à l'apprenant sera : « Zone de chargement des réactifs scannés » (issue du tableau de la figure 3.1). Dans ce scénario, l'apprenant ne participe pas, il est spectateur : seul le formateur intervient. L'implémentation des opérations est indépendante du scénario. Dans Mascaret l'opération `Montrer()` peut être implémentée par le fait de mettre en rouge le composant. L'implémentation de `Expliquer()` peut très bien être sous forme d'un texte OSD qui s'affiche ou d'une synthèse vocale. Dans ce scénario, le rôle formateur peut être joué par un agent autonome ou bien par un humain.

3.2.2 Scénario de prise en main de l'application

Ce premier scénario permet de présenter et d'identifier les différents sous-systèmes de l'instrument. Avant de présenter un scénario basé sur la procédure complète, nous souhaitons un scénario plus simple afin que l'apprenant puisse assimiler les différents types d'actions qu'il peut réaliser dans l'environnement. Pour cela, nous proposons un scénario avec un rôle « Utilisateur » qui devra réaliser un exemple de procédure métier, ainsi qu'un rôle « Formateur » qui aura pour objectif de surveiller les actions de l'apprenant, et en cas de mauvaise action d'en empêcher sa réalisation (figure 3.18). Dans ce scénario, le rôle formateur sera joué par un agent autonome, dans notre cas ce sera par Chrysaor. En effet, ce comportement étant plus complexe, un humain ne pourrait pas réagir suffisamment rapidement pour réaliser les actions. L'apprenant est par contre acteur dans ce scénario :

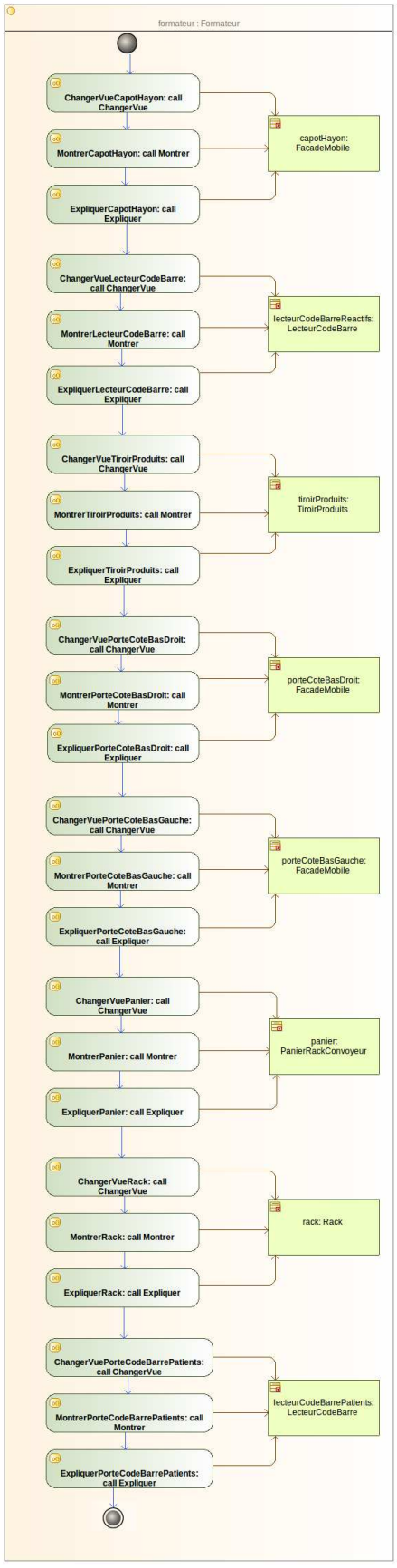

Figure 3.17 – Scénario pédagogique de présentation des sous-systèmes

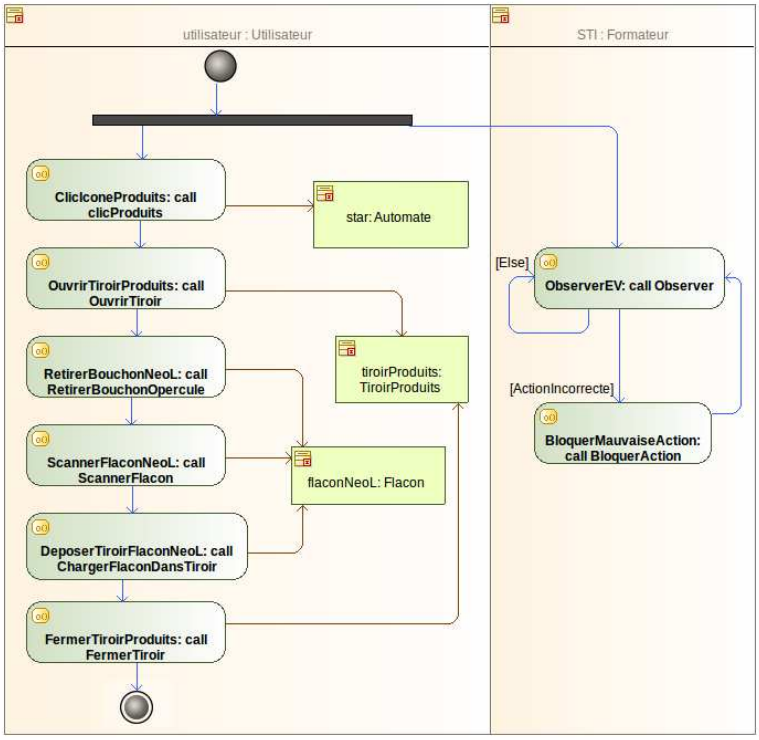

Figure 3.18 – Scénario pédagogique avec une procédure métier courte et un rôle formateur au comportement minimaliste

c'est lui qui réalise la procédure métier.

3.2.3 Scénario de la tâche à réaliser avec un guidage fort

Ce deuxième scénario permet à l'apprenant de s'essayer à la réalisation d'une procédure, tout en ayant un contrôle sur les mauvaises actions. Suite à cela, nous utilisons la procédure métier de 125 actions issue de la figure 3.13 pour proposer un scénario pédagogique (figure 3.19). L'ensemble de ces actions peut être visualisé dans l'annexe A. Nous associons à ce scénario le même comportement formateur que précédemment, c'est-à-dire un comportement qui surveille les actions de l'apprenant et les bloque si celles-ci sont incorrectes.

3.2.4 Scénario de la tâche à réaliser avec le comportement de PEGASE

Un comportement de STI plus complexe, c'est-à-dire qui sera plus proche du comportement de Pegase peut également être proposé. La figure 3.20 montre la facilité de coupler une procédure métier (celle de 125 actions par exemple), à un comportement de STI à plusieurs rôles : procédure décrite dans la figure 2.33 de la section 2.4.

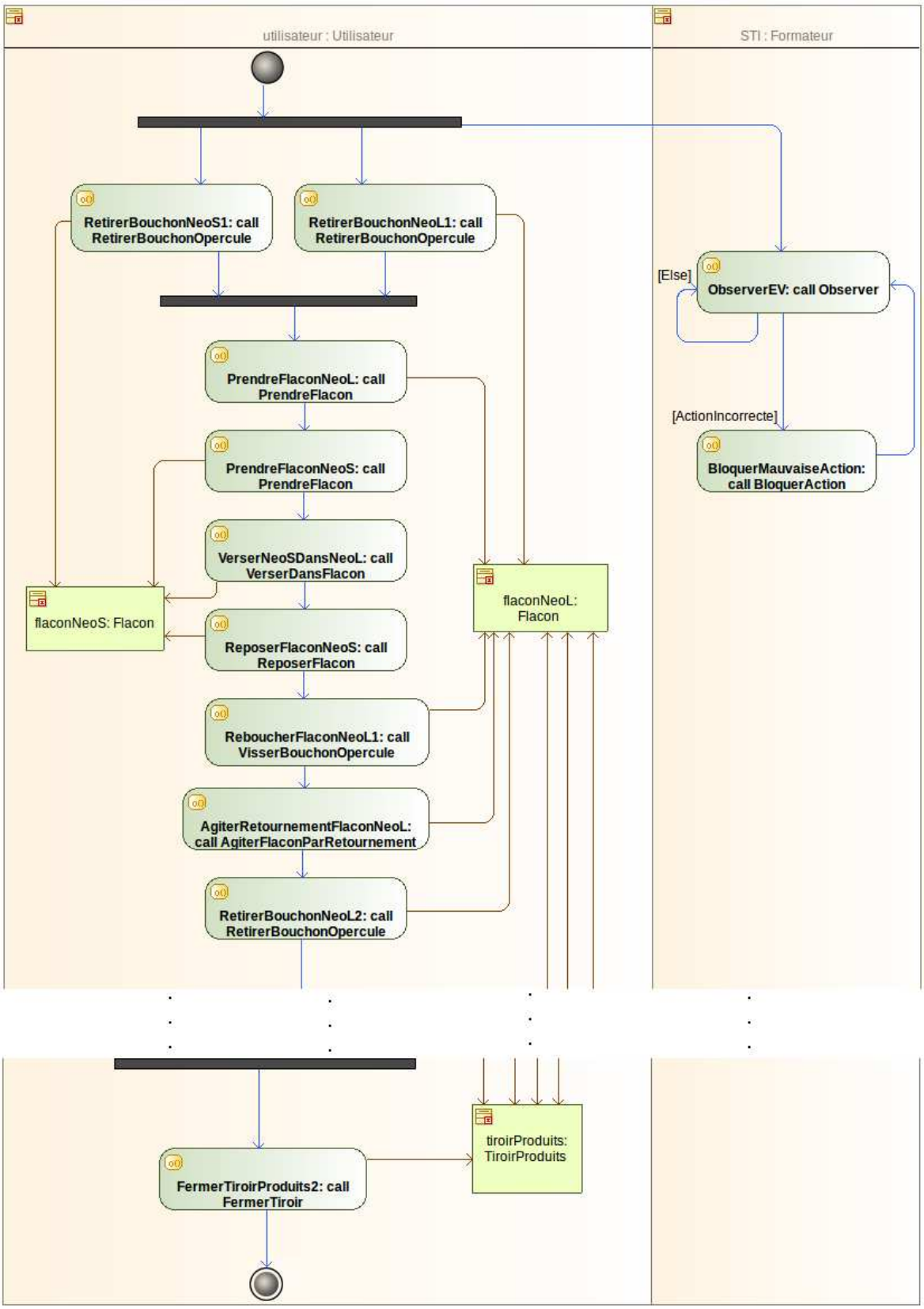

Figure 3.19 – Scénario pédagogique avec une procédure métier complète et un rôle formateur au comportement minimaliste

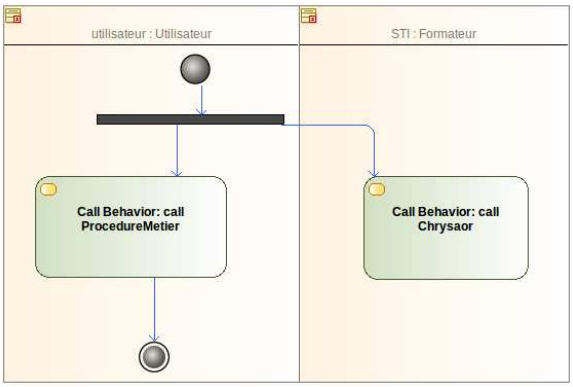

Figure 3.20 – Scénario pédagogique avec une procédure métier complète et un rôle formateur au comportement de Chrysaor

3.2.5 Bilan

Les trois premiers scénarios pédagogiques seront utilisés pour réaliser les expérimentations présentées dans la section suivante. En effet, le comportement de tuteur décrit dans ces scénarios reste basique afin de pouvoir aisément mener nos premières expérimentations, sans avoir trop de paramètres à définir pour les spécialistes en sciences cognitives. Une expérimentation future sera menée à partir du scénario complet décrit dans la figure 3.20

3.3 Expérimentations

Nous avons souhaité réaliser des expérimentations sur notre application StarApplication (section 3.1) et notre atelier d'ingénierie pédagogique (section 3.2) afin de vérifier l'apprentissage de procédure ainsi que la capacité de transférer sur un STA-R® réel ce qui a été appris précédemment [Hoareau et al., 2013b]. Pour cela, nous allons tout d'abord présenter en section 3.3.1 l'objectif de nos expérimentations. Puis nous allons présenter les deux expérimentations que nous avons choisi de mener. Tout d'abord en section 3.3.2 nous présenterons l'expérimentation permettant de vérifier l'efficacité de l'apprentissage par la réalité virtuelle. Puis en section 3.3.3 nous présenterons la seconde expérimentation permettant de vérifier le transfert de l'apprentissage obtenu par la réalité virtuelle, sur un instrument réel.

3.3.1 Objectif de l'étude

L'objectif de ce travail était double : déterminer l'utilité et l'efficacité d'un environnement virtuel pour l'apprentissage d'une procédure. D'une part, il s'agissait d'évaluer l'apprentissage d'une procédure dans un environnement virtuel et son stockage en mémoire à long terme. D'autre part, il s'agissait d'étudier le transfert de cette procédure acquise en environnement

virtuel à une situation réelle. Pour ce faire, nous avons réalisé deux expérimentations.

La première expérimentation s'est déroulée en deux sessions (section 3.3.2). La première session consistait au recueil de données comportementales destinées à observer le déroulement de l'apprentissage de procédure. La deuxième session, après une semaine de délai, consistait au rappel de la procédure afin de vérifier le stockage de la procédure en mémoire à long terme. Nous utilisons les travaux de Anderson [1983, 1993, 1995] pour analyser cette première expérimentation. En effet, Anderson considère qu'il y a trois phases successives dans l'apprentissage.
- la phase cognitive pendant laquelle l'apprenant ne possède que peu de connaissances sur la procédure : cette phase est lente et coûteuse cognitivement,
- la phase associative pendant laquelle l'apprenant utilise mieux les connaissances,
- la phase autonome : l'apprenant ne commet plus d'erreurs et exécute la procédure rapidement.

La deuxième expérimentation opposait deux types de formation : une formation dite « traditionnelle » (avec un formateur) et une formation dite « virtuelle » (avec un environnement virtuel), ainsi qu'un groupe contrôle (section 3.3.3). L'objectif étant de vérifier le transfert en situation réelle des connaissances acquises en environnement virtuel. La question du transfert est fondamentale pour les environnements virtuels de formation. En effet, les connaissances acquises doivent être réutilisables dans un autre contexte que celui du virtuel : le contexte réel [Butterfield, 1989; Prawat, 1989; Laliberté, 1990; Toupin, 1993, 1995].

3.3.2 Expérimentation 1 : apprentissage de procédure en environnement virtuel

3.3.2.1 Protocole

`Population`
Douze étudiants (11 hommes et 1 femme), âgés de 20 ans en moyenne (19-23 ans), ont participé à cette étude. Ces étudiants étaient en 4ème année d'Ecole d'ingénieurs spécialisée en informatique (Ecole Nationale d'Ingénieurs de Brest). Aucun d'entre eux n'avait vu ou manipulé l'environnement auparavant.

`Matériel`
Cette étude a utilisé le StarApplication décrit précédemment. Cet environnement virtuel affiché sur un ordinateur portable était complété par le StarDroid. Ces deux dispositifs communiquaient grâce à un réseau wifi généré par un routeur. Une pré-étude a permis de déterminer quelle procédure utiliser pour l'expérimentation. Nous avons également intégré un comportement du tuteur relativement simple : en cas de mauvaise action, le tuteur bloquait l'environnement, il ne se passait rien. L'apprenant ayant accès, sur sa demande, à des assistances vocales indiquant l'action à réaliser, il pouvait ainsi se rendre compte de son erreur.

`Procédure`
La passation de l'expérimentation se faisait par groupe de quatre apprenants qui disposaient

chacun d'un ordinateur portable avec l'environnement virtuel et d'une tablette tactile avec le logiciel StarDroid pour le lancement de tests. Les participants ne pouvaient pas communiquer entre eux. L'expérimentation se déroulait en deux séances. Au commencement de la première séance, une démonstration de la manipulation de l'interface était faite aux participants avec une procédure créée à cet effet. Ensuite chaque apprenant prenait place à son poste de travail et l'expérimentation débutait. Les apprenants devaient réaliser 7 essais consécutifs d'une procédure de tests d'analyses médicales composée de 125 actions. A la suite d'un délai d'une semaine, les participants revenaient pour effectuer la deuxième session de l'expérimentation. Cette séance consistait en la réalisation de trois essais consécutifs de la procédure. Cette deuxième session utilisait le même matériel, la même procédure et les mêmes mesures que lors de la première session. Sur la figure 3.21 nous pouvons voir les actions réalisables par l'apprenant quand il sélectionne un flacon, ainsi que l'assistant vocal.

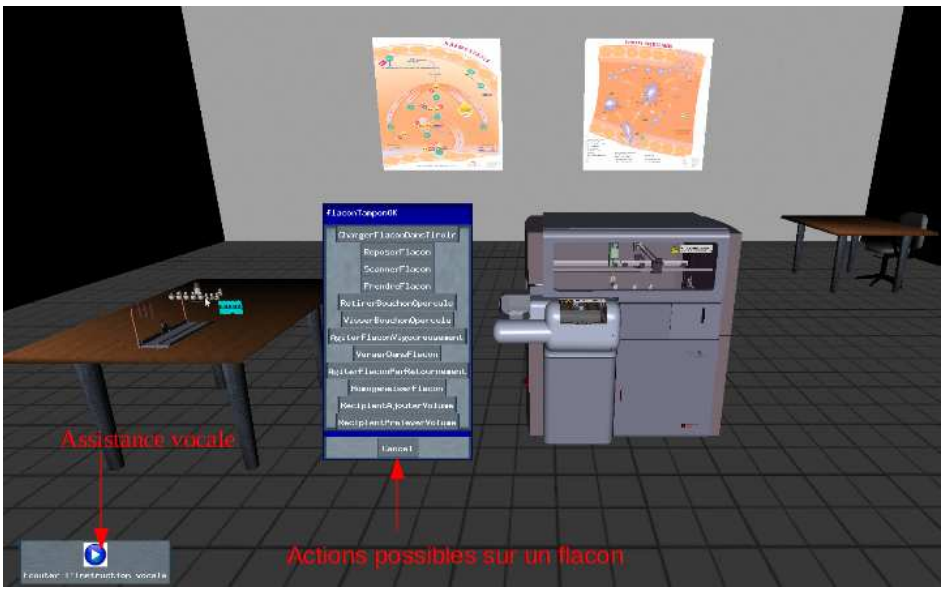

Figure 3.21 – Interactions et assistances vocales dans l'application du STA-R®

`Données recueillies`
Afin d'évaluer l'apprentissage de la procédure, nous avons recueilli des données comportementales : le temps total de réalisation de la procédure, le nombre de consultation des instructions sonores et le nombre d'erreurs (quand l'apprenant ne sélectionne pas le bon élément pour réaliser l'action ou ne sélectionne pas la bonne action dans le menu contextuel).

3.3.2.2 Résultats

`Temps total de réalisation de la procédure`
Le temps total de réalisation de la tâche (figure 3.22) diminue de manière significative en fonction du nombre d'essais : $F\ (9,\ 99) = 98{,}40$; $p < 0.001$. Concernant les essais de la première session (essais 1 à 7), le temps total de réalisation de la tâche ne cesse de diminuer du 1er au 7ème essai. Les comparaisons analytiques montrent des différences significatives entre les trois premiers essais : le temps de réalisation à l'essai 1 est plus long qu'à l'essai 2, qui est lui-même plus long qu'à l'essai 3. L'essai 4 est également significativement plus long

que l'essai 5. La différence de temps des essais 5, 6 et 7 n'est pas significative, nous pouvons ainsi considérer que nous avons atteint un plateau : c'est le début de la phase autonome mais celle-ci ne se maintient pas. Concernant les essais de la deuxième session (essais 8 à 10), les comparaisons analytiques montrent une différence significative entre les essais 8 et 9 mais pas entre les essais 9 et 10. Si l'on compare les performances des deux sessions, les participants mettent significativement plus de temps à réaliser la procédure lors du 1er essai de la deuxième session (essai 8) que lors du dernier essai de la première session (essai 7). Il n'y a pas de différence de performance entre l'essai 7 et les essais 9 et 10, ce qui confirmerait la présence de la procédure en mémoire à long terme.

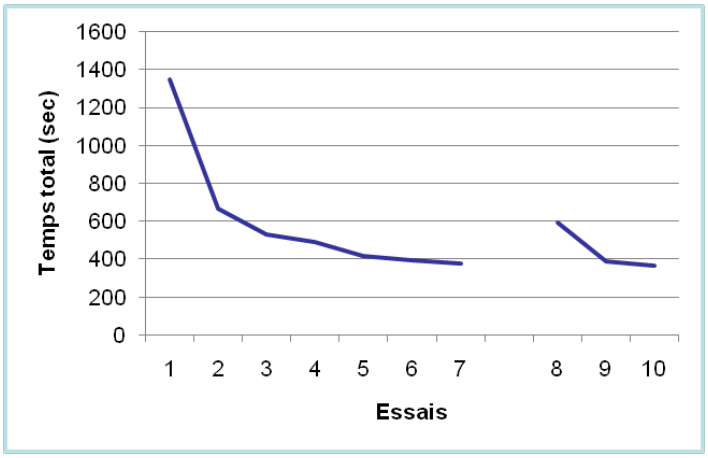

Figure 3.22 – Temps total de réalisation de la tâche en fonction du nombre d'essais de la procédure

`Nombre de consultations des instructions`

Le nombre de consultation des instructions (figure 3.23) diminue significativement en fonction du nombre d'essais : F (9, 99)= 121,53 ; $p < 0.001$. Les comparaisons analytiques par paires montrent une diminution du nombre de consultations des instructions des essais 1 à 4. Après les trois premiers essais, la moyenne du nombre de consultations des instructions est égale ou inférieure à 1. Au début de la deuxième session (essai 8), on observe une augmentation significative du nombre de consultations des instructions par rapport à l'essai 7. L'essai 8 a un nombre plus élevé de consultations que l'essai 9. Les comparaisons entre l'essai 7 et les essais 9 et 10 ne montrent aucune différence significative. Après un délai de 7 jours, la consultation des instructions est nécessaire durant l'essai 8, et redevient en moyenne égale à 1 à l'essai 10.

`Nombre d'erreurs`

Le nombre d'essais exerce un effet sur le nombre d'actions non pertinentes. Cet effet est significatif : F (9, 99)= 8 ; $p < 0.01$. Les comparaisons analytiques montrent une diminution du nombre d'actions non pertinentes entre l'essai 4 et l'essai 5. Il n'y a pas d'autres différences significatives concernant cet indicateur. Le nombre d'actions non pertinentes ne sera pas pris en compte dans l'interprétation des résultats.

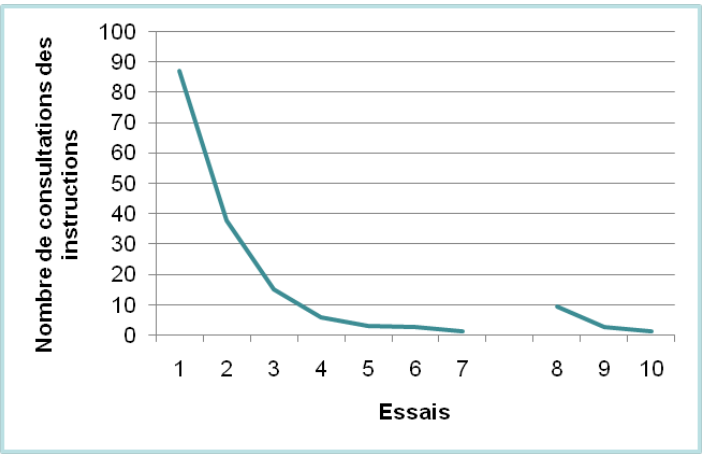

Figure 3.23 – Nombre de consultations des instructions en fonction du nombre d'essais de la procédure

3.3.2.3 Conclusion

L'objectif de cette expérimentation était d'évaluer l'utilité du StarApplication par le biais de l'apprentissage d'une procédure. Les résultats ont montré que l'environnement virtuel utilisé a permis l'acquisition d'une procédure de lancement de tests sanguins d'analyses médicales. Le temps total de réalisation de la tâche et le nombre de consultations des instructions diminuent au fil des essais. À la suite d'une période de 7 jours, les apprenants reconsultent partiellement les instructions uniquement lors du premier essai (essai 8), puis leurs performances deviennent similaires à celles obtenues à la fin de la première session. Le nombre d'actions non pertinentes peut difficilement être pris en compte à cause de problèmes dus à la manipulation même de l'interface. En effet, nous avions défini le nombre d'actions non pertinentes comme un clic au mauvais endroit au mauvais moment mais l'utilisation répétée (et donc rapide) de l'interface et des erreurs de conception nous amènent à réaliser des erreurs de pointage du curseur sur l'action voulue. Ainsi, si cette étude confirme l'utilisabilité de StarApplication pour l'apprentissage de procédure, un nombre plus important d'essais serait nécessaire afin d'obtenir une stabilisation des performances (phase autonome) et un nombre inférieur de consultations des instructions lors du 8ème essai, attestant l'acquisition parfaite de la procédure et son stockage en mémoire à long terme.
Apprendre à l'aide d'un environnement virtuel n'a d'intérêt que si les habiletés acquises grâce à ce dispositif peuvent être utilisées dans la situation réelle. C'est ce que tente d'étudier l'expérimentation suivante.

3.3.3 Expérimentation 2 : transfert du virtuel au réel

3.3.3.1 Protocole

`Population`
Cinquante-huit participants (45 femmes et 13 hommes), âgés de 20 ans (17-22 ans), ont

participé à cette étude. Ces participants, de 1ère et 2ème année de BTS d'analyses en biologie médicale, étaient scolarisés au Lycée Paul Eluard de Saint-Denis en région parisienne. De ce fait, les participants possédaient des connaissances similaires dans le domaine de l'hémostase et dans la manipulation du matériel de laboratoire. Les participants étaient répartis en trois groupes équivalents en nombre : groupe contrôle, formation traditionnelle, formation virtuelle.

`Matériel`
Cette étude a utilisé le même environnement virtuel que lors de la première expérimentation (pour le groupe de formation virtuelle). Par ailleurs, deux automates d'analyses médicales réels accompagnés de leur interface tactile ainsi que tout le matériel nécessaire pour exécuter la procédure étaient utilisés pour les manipulations en situation réelle (pour le groupe de formation traditionnelle et pour la restitution). Ils étaient disposés dans une salle de travaux pratiques de biologie mise à notre disposition par le lycée. Les deux automates, présents dans la même pièce, étaient séparés par un panneau amovible. Au cours de la restitution, les participants avaient accès à un document technique qui reprenait l'ensemble des actions à réaliser pour exécuter correctement la procédure.

`Procédure`
Les participants devaient effectuer la même procédure de lancement de tests que celle utilisée dans l'expérimentation précédente. Après avoir suivi une matinée de formation collective (par groupe de 6), les participants devaient dérouler, dans le laboratoire, l'ensemble de la procédure sur le dispositif réel en passation individuelle. L'échantillon était divisé en trois conditions expérimentales : la condition « formation traditionnelle », la condition « formation en environnement virtuel » et le groupe contrôle (figure 3.24).

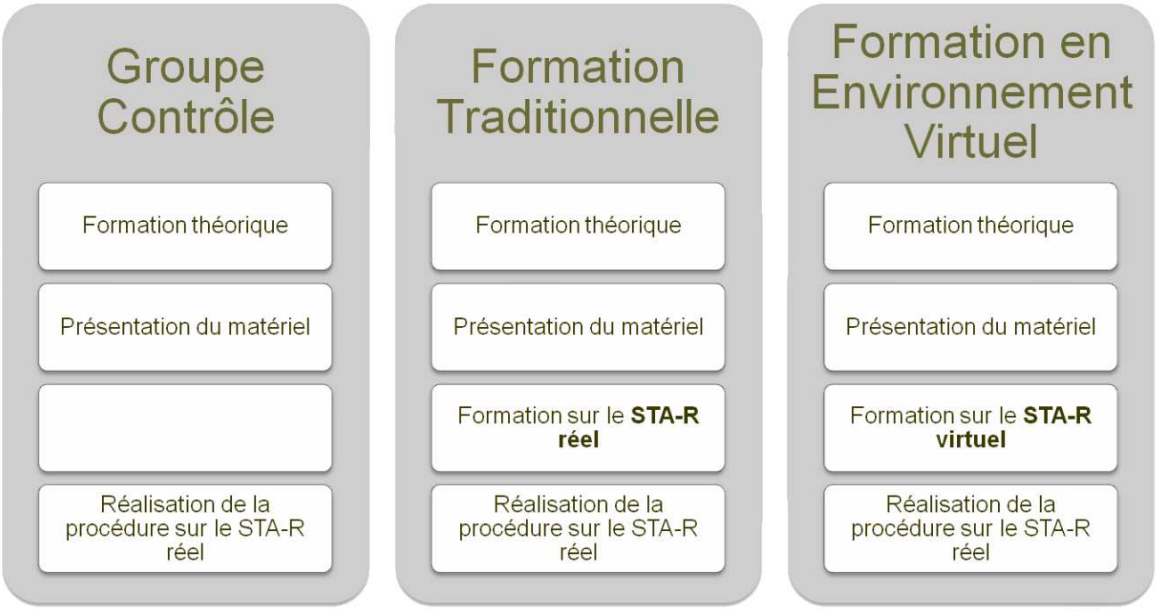

Figure 3.24 – Déroulement de l'expérimentation à Saint-Denis

La matinée de formation des participants du groupe « formation traditionnelle » s'est déroulée de la manière suivante : ils ont assisté à une formation théorique sur le principe de l'hémostase et sur les différentes étapes principales qui composent la procédure de lancement

de tests, en salle sous forme de diaporama (figure 3.25).
Ensuite, dans la salle de TP, la formatrice leur présentait le matériel nécessaire pour réaliser la procédure sans les laisser le manipuler. Enfin, la formatrice déroulait l'ensemble de la procédure sur l'automate de coagulation, en la commentant, puis les participants, divisés en deux groupes de trois, pouvaient exécuter la procédure guidés par la formatrice, pendant deux heures. L'après-midi, les participants devaient exécuter l'ensemble de la procédure sur l'automate de coagulation, en passation individuelle : c'est la phase de restitution (figure 3.26).

Figure 3.25 – Formation théorique lors de l'expérimentation à Saint-Denis

Figure 3.26 – Restitution sur l'instrument lors de l'expérimentation à Saint-Denis

Les participants du groupe « formation en environnement virtuel » ont assisté à la même formation théorique et à la même présentation du matériel que le groupe « formation traditionnelle », la réalisation de la procédure par la formatrice était remplacée par un entraînement en environnement virtuel (figure 3.27) de deux heures également. La restitution de l'après-midi se faisait dans les mêmes conditions.

Les participants du groupe contrôle ont assisté à la même présentation théorique et du

matériel que les deux autres groupes mais ils n'ont pu s'entraîner ni sur le dispositif réel, ni sur le dispositif virtuel. Ils ont ensuite restitué la procédure dans les mêmes conditions l'après-midi.

Figure 3.27 – Formation par la réalité virtuelle lors de l'expérimentation à Saint-Denis

`Données recueillies`
Les mesures de performances des participants, sur le dispositif réel lors de la restitution, ont été réalisées suite à un dépouillement chronométré (enregistrements vidéo) des données comportementales portant sur la restitution individuelle de la procédure. Les données recueillies portent sur le succès de réalisation de la tâche dans un temps imparti (45 minutes), le temps total de réalisation de la procédure, le temps de consultation du document technique, le nombre d'appel à l'aide de la part des apprenants et le nombre d'interventions du technicien pour assister le participant quand il réalisait une mauvaise action qui pouvait avoir des conséquences sur le bon fonctionnement de l'appareil.

3.3.3.2 Résultats

La réussite de la procédure étant effective pour les groupes « formation traditionnelle » et « formation en environnement virtuel », mais pas pour le groupe contrôle. Nous ne présenterons ici que les résultats liés aux données comportementales : Temps total de réalisation de la procédure, Nombre de consultations du document technique, Nombre d'appels à l'aide et Nombre d'interventions du technicien (figure 3.28).

`Temps total de réalisation de la tâche`
Le temps total de réalisation de la procédure lors de la restitution varie selon la condition de formation : $F\ (2,\ 51) = 46{,}60$; $p < 0.001$. Les comparaisons analytiques indiquent que la durée de réalisation de la procédure est significativement plus courte lorsque les participants ont suivi la formation traditionnelle ($m = 1178$ sec ; $\sigma = 155{,}4$) que lorsqu'ils ont suivi la formation en environnement virtuel ($m = 1791$ sec ; $\sigma = 496{,}4$), elle-même significativement plus courte que lorsqu'ils n'ont suivi aucune formation ($m = 2338$ sec ; $\sigma = 346{,}7$).

	Formation traditionnelle	Formation EVAH	Groupe Contrôle
Temps total	19:48	30:02	39:09
Nb Consultation	9,5	1,9	48,7
Nb Aide	0,2	0,7	3,1
Nb Intervention	1,4	2,3	3,9
Validation	100%	100%	72%

Figure 3.28 – Résultats des trois types de formation lors de l'expérimentation à Saint-Denis

`Nombre de consultations du document technique`
Le nombre de consultations du document technique varie selon la condition de formation : $F\ (2,\ 51) = 232{,}93$; $p < 0.001$. Les comparaisons analytiques indiquent que ce nombre est significativement moins élevé pour les participants du groupe « formation traditionnelle » (m = 0,7 ; σ = 1,5) et ceux du groupe « formation en environnement virtuel » (m = 1,9 ; σ = 4,7) par rapport aux participants du groupe contrôle (m = 48,7 ; σ = 12,2). Il n'y a pas de différence significative entre la condition « formation traditionnelle » et la condition « formation en environnement virtuel ».

`Nombre d'appels à l'aide`
Le type de formation a un effet sur le nombre d'appels à l'aide exprimés par les participants auprès du technicien lors de la restitution de la procédure : $F\ (2,\ 51) = 36{,}48$; $p < 0.001$. Les comparaisons analytiques indiquent que nombre est significativement moins élevé pour les participants du groupe « formation traditionnelle » (m = 0,2 ; σ= 0,4) et ceux du groupe « formation en environnement virtuel » (m = 0,7 ; σ = 0,7) par rapport aux participants du groupe contrôle (m = 3,1 ; σ = 1,7). Il n'y a cependant pas de différence significative entre la « formation en environnement virtuel » et la « formation traditionnelle ».

`Nombre d'interventions du technicien`
Le nombre d'interventions du technicien pendant la restitution varie en fonction du type de formation : $F\ (2,\ 51) = 5{,}47$; $p < 0.01$. Les comparaisons analytiques montrent que les participants ayant suivi la formation traditionnelle entraînent moins d'interventions du technicien (m = 1,4 ; σ = 1,8) que les participants ayant suivi la formation en environnement virtuel (m = 2,3 ; σ = 2,2) ou ceux n'ayant suivi aucune formation (m = 3,9 ; σ = 2,8). Le technicien était libre

3.3.3.3 Conclusion

Cette étude avait pour objectif d'évaluer le transfert d'une procédure acquise en environnement virtuel au réel avec pour support un environnement virtuel d'opérations de tests sanguins d'analyses médicales. En termes de performances de réalisation de la procédure en situation réelle, à l'exception du temps total de réalisation de la tâche, les résultats ne montrent pas de différence significative entre la formation traditionnelle et la formation en environnement virtuel. Cette différence persistante sur la durée de restitution pourrait s'expliquer par une appréhension de la part des apprenants à la manipulation des réactifs

chimiques et de l'automate d'analyses biologiques. Elle pourrait également provenir de choix non pertinents lors de la conception de l'environnement virtuel.

3.3.4 Bilan

Ces deux études avaient pour but d'étudier l'utilité et l'efficacité d'un environnement virtuel pour la formation. Afin de l'étudier précisément, nous avons scindé en deux notre étude avec : d'une part la possibilité d'apprendre une procédure grâce à un environnement virtuel, et d'autre part la capacité de transférer cette procédure acquise à une situation réelle. Si les résultats de la première expérimentation montrent qu'il serait possible d'apprendre une procédure de tests sanguins d'analyse médicale en environnement virtuel, après un délai de sept jours, ils doivent reconsulter certaines instructions afin de se remémorer l'intégralité de la procédure. Les résultats de la deuxième expérimentation montrent que les participants formés à l'aide de l'environnement virtuel réussissent aussi bien (mais en mettant plus de temps) que les participants formés par une formatrice.

3.4 Synthèse

Dans la section 3.1 nous avons tout d'abord présenté l'instrument STA-R® qui est l'automate de coagulation commercialisé par Stago sur lequel nous avons choisi de baser notre application. Ceci nous a permis de détailler son mode de fonctionnement ainsi que le principe des tests biologiques pouvant être réalisés sur cet instrument. Nous avons ensuite proposé un modèle du STA-R® que nous avons décrit en nous appuyant sur les concepts proposés dans le chapitre 2. Nous avons ainsi détaillé la structure, les comportements et les procédures du STA-R® selon trois packages. L'application Mascaret du STA-R® a ainsi été développée en se basant sur ce modèle : cette application est composée de l'environnement virtuel 3D représentant le STA-R® ainsi que d'une interface Android représentant son logiciel.

En nous appuyant sur ce modèle de STA-R®, nous avons proposé quelques scénarios pédagogiques. Le premier scénario permet au formateur de présenter et d'expliquer les différents composants du système. Un autre scénario est composé d'une procédure métier simplifiée ainsi que d'un comportement de tuteur. Ce scénario permet à l'apprenant de s'essayer à la réalisation d'une procédure avec un tuteur ayant un contrôle sur les mauvaises actions. Le dernier scénario pédagogique proposé comporte le même comportement de tuteur mais une procédure métier complète de 125 actions.

Ces scénarios pédagogiques ont servi à réaliser des expérimentations sur des apprenants novices. Ces expérimentations avaient deux objectifs : déterminer l'utilité et l'efficacité d'un environnement virtuel pour l'apprentissage d'une procédure. D'une part, il s'agissait d'évaluer l'apprentissage d'une procédure dans un environnement virtuel et son stockage en mémoire à long terme. D'autre part, il s'agissait d'étudier le transfert de cette procédure acquise en environnement virtuel à une situation réelle. Pour ce faire, nous avons réalisé deux expérimentations sur des élèves d'école d'ingénieurs en informatique et de BTS d'analyses en biologie médicale. Ces expérimentations ont permis de montrer qu'il est possible d'apprendre

une procédure de formation à un instrument biomédical en environnement virtuel, et que ces connaissances sont transférables sur un instrument réel.

Les expérimentations ont fait l'objet de plusieurs publications, notamment [Hoareau et al., 2013a,b].

Conclusions

Dans ce manuscrit de thèse nous avons proposé Chrysaor : un Système Tutoriel Intelligent pour l'apprentissage humain en environnement virtuel et son application pour la formation aux instruments médicaux Stago. Dans ce chapitre nous allons dresser un bilan de nos travaux, puis discuter des apports de notre proposition, et ensuite envisager certaines perspectives.

Bilan

Dans le chapitre *Introduction*, nous avons situé le contexte et énoncé notre problématique. Cette thèse est financée par Stago qui est une société de l'industrie du diagnostic *in vitro* entièrement dédiée à l'exploration de l'hémostase (processus physiologique qui permet d'interrompre le saignement pour éviter l'hémorragie) et de la thrombose (formation d'un caillot dans un vaisseau). Nous proposons d'utiliser la réalité virtuelle et les environnements virtuels pour la formation aux instruments de diagnostic biomédical Stago. Dans un premier temps nous avons choisi de baser notre application de formation par la réalité virtuelle sur un seul instrument : le STA-R®, qui est l'instrument haut de gamme chez Stago. Nous avons identifié deux objectifs principaux pour nos travaux. Tout d'abord nous souhaitons que le formateur puisse décrire sa propre situation d'apprentissage. Pour cela il faut coupler l'environnement virtuel à un scénario pédagogique dont l'exécution doit permettre la construction des connaissances visées chez les apprenants. Ensuite, nous souhaitons coupler notre modèle de scénario pédagogique à un STI afin d'utiliser les connaissances du scénario pédagogique. Les caractéristiques importantes pour notre STI Chrysaor sont : la généricité, la modularité, l'individualisation, l'adaptativité et le lien avec le scénario pédagogique.

Dans le *chapitre 1*, nous avons identifié les points forts et les faiblesses des modèles existants de STI, de scénarios pédagogiques et de représentation des connaissances. Nous avons choisi de nous baser sur le modèle de scénario pédagogique Poseidon, le modèle de STI Pegase et le méta-modèle Mascaret. L'essentiel de nos travaux est dédié à notre STI. Nous proposons d'améliorer la généricité et la modularité de Pegase en se basant sur les systèmes multi-agents. En effet, notre STI est composé d'agents aux comportements propres. Nous proposons

également de travailler sur la communication entre ces agents ainsi que leur organisation. Dans un deuxième temps, un couplage entre Poseidon et notre STI est proposé. Pour cela nous proposons qu'un scénario pédagogique soit une instance d'un environnement. En effet, jusqu'à présent Pegase raisonnait en se basant, entre autre, sur les connaissances d'un environnement représentant le modèle du domaine. De la même manière, notre STI peut raisonner sur les connaissances du scénario pédagogique. Pour représenter nos connaissances nous nous basons sur le méta-modèle Mascaret. Notre proposition de STI est appelé Chrysaor.

Dans le *chapitre 2*, nous avons identifié les concepts importants afin de combler les faiblesses définies dans l'état de l'art. Ces concepts sont les rôles, organisations, comportements, bases de connaissances et activités. Nous souhaitons réifier ces concepts pour les rendre manipulables par notre STI. Le méta-modèle Uml propose des concepts couvrant certains critères que nous avons décrits, c'est pourquoi nous avons formalisé certaines notions d'Uml afin d'étendre notre propre modèle. Le concept de `UML::Interface` se rapproche de notre description de rôles, c'est pourquoi nous faisons hériter notre `Mascaret::Role` d'un `UML::Interface`. Un agent pourra jouer un rôle dans une organisation car il aura déclaré un rôle comme étant réalisé grâce au concept de `UML::InterfaceRealization`. Uml propose également le concept de `UML::Collaboration` qui correspond tout à fait à notre définition qu'une organisation peut être composée de plusieurs rôles. Nous faisons ainsi hériter `Mascaret::OrganisationalStructure` de `UML::Collaboration`. L'implémentation de cette collaboration est une `UML::CollaborationUse` à partir duquel nous faisons hériter `Mascaret::OrganisationalEntity` qui est l'instanciation de la structure organisationnelle de Mascaret. Enfin, les critères que nous avons identifiés pour décrire une activité sont couverts par le concept de `UML::Activity`. Une `Mascaret::Activity` hérite donc de `UML::Activity`. Mascaret définit dans son modèle le concept d'agent. Ces agents jouent des rôles définis dans les organisations. Afin de pouvoir raisonner sur la simulation, les agents ont besoin de connaissances sur l'environnement. C'est pourquoi la base de connaissances d'un agent Mascaret référence un `Mascaret::Environment`. Nous avons doté les agents de moyens de communication en utilisant le standard de communication agent : *FIPA-ACL*. Nous proposons également deux comportements automatiques à nos agents. Tout d'abord le comportement de communication qui vérifie et traite les messages reçus par l'agent, comme par exemple gérer une demande de valeur d'un attribut (l'agent qui reçoit le message ayant accès à cette connaissance). Il y a également le comportement procédural qui permettra à l'agent d'avoir son propre suivi de procédure. Ces concepts, ainsi formalisés dans Mascaret, sont utilisables afin de définir ce qu'est un scénario pédagogique au sein de Mascaret. Afin que Chrysaor puisse raisonner dessus, nous proposons que le scénario pédagogique soit une connaissance au même titre que l'environnement de simulation. Le scénario pédagogique est donc défini par un `Mascaret::Model` et un `Mascaret::Environment`. Dans un premier temps nous nous sommes positionnés vis à vis de la référence en tant que langage de modélisation pédagogique : IMS-LD. En effet, nous avons montré qu'il est possible à partir d'un scénario défini en Chrysaor, de générer un scénario en IMS-LD mais que l'inverse ne l'est pas. Notre modèle étant défini, nous proposons d'adapter le modèle de Pegase en le décrivant de la même manière que pour le scénario pédagogique. Cette organisation correspond ainsi à Chrysaor. Quelques exemples de comportements plus ou moins évolués de Chrysaor sont également proposés afin de montrer la modularité de notre STI. Nous allons présenter notre application de réalité virtuelle basée sur un instrument Stago ainsi que les expérimentations que nous avons menées pour tester notre application.

Dans le *chapitre 3*, nous avons présenté l'instrument STA-R® qui est l'automate de coagulation commercialisé par Stago sur lequel nous avons choisi de baser notre application. Ceci nous a permis de détailler son mode de fonctionnement ainsi que le principe des tests biologiques pouvant être réalisés sur cet instrument. Nous avons ensuite proposé un modèle du STA-R® que nous avons décrit en nous appuyant sur les concepts proposés dans le chapitre 2. Nous avons ainsi détaillé la structure, les comportements et les procédures du STA-R® selon trois packages. L'application Mascaret du STA-R® a ainsi été développée en se basant sur ce modèle : cette application est composée de l'environnement virtuel 3D représentant le STA-R® ainsi que d'une interface Android représentant son logiciel. En nous appuyant sur ce modèle de STA-R®, nous avons proposé quelques scénarios pédagogiques. Le premier scénario permet au formateur de présenter et d'expliquer les différents composants du système. Un autre scénario est composé d'une procédure métier simplifiée ainsi que d'un comportement de tuteur. Ce scénario permet à l'apprenant de s'essayer à la réalisation d'une procédure avec un tuteur ayant un contrôle sur les mauvaises actions. Le dernier scénario pédagogique proposé comporte le même comportement de tuteur mais une procédure métier complète de 125 actions. Ces scénarios pédagogiques ont servi à réaliser des expérimentations sur des apprenants novices. Ces expérimentations avaient deux objectifs : déterminer l'utilité et l'efficacité d'un environnement virtuel pour l'apprentissage d'une procédure. D'une part, il s'agissait d'évaluer l'apprentissage d'une procédure dans un environnement virtuel et son stockage en mémoire à long terme. D'autre part, il s'agissait d'étudier le transfert de cette procédure acquise en environnement virtuel à une situation réelle. Pour ce faire, nous avons réalisé deux expériences sur des élèves d'école d'ingénieurs en informatique et de BTS d'analyses en biologie médicale. Ces expérimentations ont permis de montrer qu'il est possible d'apprendre une procédure de formation à un instrument biomédical en environnement virtuel, et que ces connaissances sont transférables sur un instrument réel.

Discussion

Avant de conclure cette thèse, il convient de discuter des apports de notre proposition. La société Stago souhaite utiliser la réalité virtuelle et les environnements virtuels pour la formation à leurs instruments de diagnostic biomédical. La simulation de la situation réelle seule ne suffisant pas, il convient d'incorporer un scénario pédagogique pour construire l'apprentissage. Nous avons également un besoin d'individualiser cette formation, ce que le scénario seul n'apporte pas (celui-ci étant défini pour tous les apprenants). Pour pallier cela nous couplons notre environnement virtuel à un Système Tutoriel Intelligent. Cela permet d'adapter la pédagogie dynamiquement à l'apprenant au cours de la formation : la pédagogie sera propre à chaque individu.

Ainsi, nous avons proposé une application de formation à un instrument Stago nommée StarApplication. Ceci nous a permis de vérifier la qualité de l'apprentissage au cours d'une formation sur cette application, ainsi que vérifier si ces connaissances sont transférables en environnement réel. L'objectif pour Stago est donc atteint. Nous avons également proposé un modèle qui permet au formateur de décrire plus facilement des scénarios pédagogiques et de les modifier. Le formateur peut ainsi adapter sa pédagogie, et les connaissances issues de ce scénario peuvent être exploitées par notre STI. Finalement, nous avons pu définir différents comportements de notre STI en nous appuyant sur notre modèle. Le formateur

peut facilement modifier les comportements du STI et choisir de coupler ou non le STI à l'application StarApplication.

Notre proposition répond ainsi aux objectifs définis. Toutefois, les expérimentations menées ne permettent pas de vérifier l'apport de Chrysaor sur l'apprentissage. En effet, celles-ci sont essentiellement basées sur l'application et non sur un comportement complexe du STI. Nous notons également que l'ergonomie du langage pour décrire les scénarios pédagogiques n'est pas intuitive pour le formateur. Bien que l'interface soit graphique et formalisée, le fait qu'elle s'appuie sur des concepts Uml ne la rend pas très ergonomique et donc ne facilite pas son utilisation par des formateurs métier. De plus, dans le scénario pédagogique, nous ne pouvons proposer que des interactions initiées par le formateur, et non pas par l'apprenant. En effet, le formateur ne peut pas décrire des comportements en réponse à des interrogations de l'apprenant (*e.g.* l'apprenant demande lui même ce qu'est tel ou tel sous-système). Enfin, notre proposition apporte principalement des concepts pour simplifier la description de notre STI. Cependant, nous n'avons pas amélioré son raisonnement qui permet d'individualiser la formation. L'individualisation étant une caractéristique importante (au même titre que la généricité, la modularité, l'adaptativité et le scénario pédagogique), celle-ci mérite d'être approfondie.

Perspectives

Proposition

Dans la section 1.3.2 nous avons identifié les faiblesses de Pegase : manque de lien avec le scénario pédagogique, faible modularité, et manque d'individualisation. Notre proposition a permis de répondre à deux de ces problèmes. En effet, notre modèle permet une grande modularité : le formateur peut facilement mettre en œuvre de nouvelles stratégies pédagogiques. Nous proposons également que le scénario pédagogique soit une connaissance au même titre que l'environnement de simulation. Cela permet au STI d'exploiter les connaissances issues du formateur.

La pédagogie contenue dans Chrysaor ne s'adapte que très peu aux divers utilisateurs : le modèle de l'apprenant est relativement simpliste. En effet, Chrysaor raisonne sur les actions de l'apprenant ainsi que sur certaines caractéristiques connues (*e.g.* le niveau de l'apprenant : débutant, confirmé, expert...). Nos futurs travaux porteront donc sur le modèle de l'apprenant afin de permettre une individualisation de la simulation pour chaque apprenant. Nos travaux pourraient se baser sur une approche récente qui traite du modèle de l'apprenant dans un système basé sur le Web sémantique (qui propose l'utilisation de langages tels que OWL) [Baishuang et Wei, 2009]. Leur modèle de l'apprenant est composé d'un modèle cognitif (historique, facteur émotionnel, motivation) et d'un modèle sur les intérêts (loisirs, objectifs, etc.) de l'apprenant. Toutes ces données seront des connaissances exploitables par Chrysaor afin d'individualiser la pédagogie.

Nous avions également débuté des travaux sur des données issues de questionnaires externes (*e.g.* ceux proposés par Hollnagel [1998]) pour la caractérisation des erreurs en fonction du

profil de l'apprenant. Une perspective d'approche serait donc d'intégrer ces données à notre modèle de l'apprenant.

Application

Les expérimentations que nous avons menées ont permis de déterminer les limites de notre application. En effet, l'analyse sur les retours des participants indique que certaines actions sont répétitives et par conséquent lassantes. Par exemple, dans le cadre de la reconstitution des réactifs, les apprenants doivent régulièrement prendre un flacon, retirer le bouchon, remuer, remettre le bouchon, etc.

Certaines actions de la procédure réalisées pendant les expérimentations nécessitent un geste précis. Nous pouvons citer l'exemple de l'agitation d'un flacon qui peut différer suivant le produit concerné : en faisant une rotation au flacon, en retournant le flacon, en le secouant, etc. Nous avons retranscrit au mieux l'animation du flacon dans l'environnement virtuel afin que les apprenants puissent différencier le mode d'agitation, cependant nous ne vérifions pas si ce geste est appris.

Kolb [1984] indique qu'une personne fait son apprentissage par l'expérience, c'est-à-dire en réalisant lui même l'action. Notre objectif principal étant que l'utilisateur apprenne le mieux possible, nous projetons donc d'ajouter à notre système différents moyens d'interaction avec l'environnement. Nous souhaitons par exemple utiliser des gants de données ou une Kinect. Cependant, nous devrons faire le lien entre ces périphériques et les actions métier. En effet, le formateur devra être capable de pouvoir décrire avec quel périphérique l'apprenant peut réaliser une action. Ce lien devra être générique, pour cela nous définirons une typologie pour décrire les actions, tel que le concept des *Primitives Comportementales Virtuelles* [Fuchs, 1996]. Nous pourrons nous appuyer sur la bibliothèque d'interface VRPN [Taylor et al., 2001] pour coupler notre application aux dispositifs.

Ces périphériques permettront à l'apprenant non seulement d'apprendre des gestes techniques, mais également de moins se lasser sur des actions répétitives.

Expérimentations

Nous avons mené deux expérimentations sur notre application, ce qui nous a permis de valider le fait que la formation par la réalité virtuelle est aussi efficace que la formation traditionnelle (pour l'apprentissage de procédures d'utilisation d'un instrument biomédical). Dans ces expérimentations, le comportement du tuteur était basique : un simple blocage d'une action en cas d'erreur. Nous souhaitons maintenant que l'efficacité d'apprentissage par la réalité virtuelle soit supérieure à la formation traditionnelle.

Pour cela, nous souhaitons évaluer l'apport du tuteur Chrysaor avec un comportement complexe (proche de celui de Pegase) sur notre application StarApplication. Notre objectif est de mener une expérimentation avec un tuteur ayant différents comportements, tels que proposer des assistances en cas de mauvaise action, ou bien encore laisser réaliser

l'action incorrecte pour en observer les effets, mais immédiatement revenir dans l'état précédent (undo). L'objectif de cette expérimentation sera de vérifier l'apport d'un tuteur sur les premiers essais d'une procédure : nous espérons obtenir une diminution du temps de réalisation de la procédure (figure 3.29). Ce sont sur ces premiers essais que l'apprenant passe beaucoup de temps à comprendre la procédure, c'est donc à ce niveau que Chrysaor pourrait apporter un bénéfice pour l'apprentissage de la procédure. En effet, l'efficacité des STI a déjà été montrée en comparant deux groupes d'apprenants dans un environnement de formation bénéficiant ou non de l'assistance des ITS. Par exemple, une expérimentation basée sur le tuteur LISP [Anderson, 1990] montre que les apprenants terminent leurs exercices 30% plus rapidement que ceux qui reçoivent une formation traditionnelle.

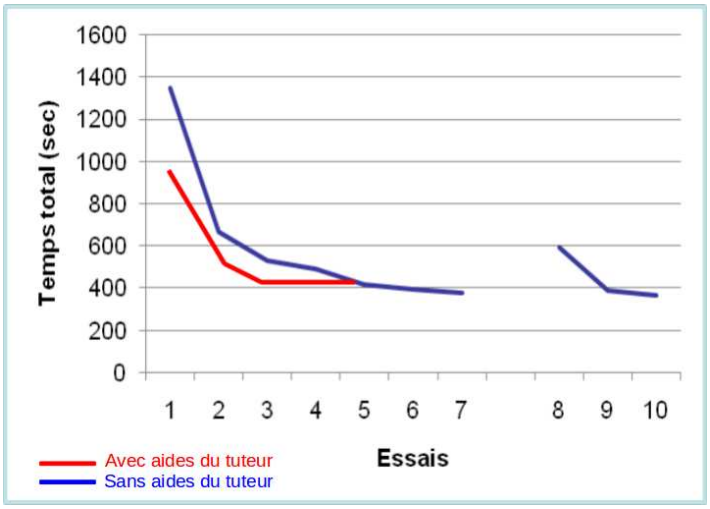

Figure 3.29 – Nombre de consultations des instructions en fonction du nombre d'essais de la procédure

Nous devrons également mener une étude préalable avec des spécialistes en sciences cognitives pour déterminer si nos comportements de tuteurs sont pertinents. Par exemple, nous pourrions avoir un comportement de STI complexe qui, en cas d'action incorrecte induisant un changement d'état du système, propose un scénario alternatif afin que le système revienne dans état permettant à l'apprenant de poursuivre la procédure. Cependant il faut s'interroger sur l'intérêt de ce type de comportement : cela ne risque-t-il pas de perturber l'apprenant ?

Annexe A

Document technique : procédure de bilan pré-opératoire

Sommaire

A.1 Étape 1 : Reconstitution des réactifs

Sur la paillasse vous disposez de :
– **Réactifs** :
  – STA-Néoplastine lyophilisé : 1 flacon,
  – Solvant : 1 flacon,
  – STA-Coag Control N : 1 flacon,
  – STA-Coag Control P : 1 flacon,
  – Tampon Owren Koller : 1 flacon,
  – STA-CaCl2 : 1 flacon,
  – STA-Desorb U : 3 flacon.
– **Matériel auxiliaire** :
  – Maxi-reducer (3) ,
  – Mini-reducer (1),
  – Aimant (1).
– **Matériel de laboratoire** :
  – 1 pipette 1mL et une pipette 5mL,
  – Cônes supplémentaires,
  – Eau distillée,

- Portoir avec 3 tubes échantillons,
- Panier et racks.

Reconstituer les réactifs en suivant les préconisations suivantes :

- **STA-Néoplastine 5** :
  - Ouvrir les 2 flacons.
  - Verser le contenu du flacon n°2 (Solvant) directement dans le flacon n°1 (Néoplastine lyophilisé) , puis reboucher.
  - Laisser Stabiliser 1 min à température ambiante.
  - Agiter par retournement.
  - Retirer le bouchon en caoutchouc.
  - Mettre un aimant blanc dans le flacon de STA-Néoplastine reconstituée.
  - Insérer un mini reducer dans le flacon de Néoplastine reconstitué, en penchant le flacon de STA-Néoplastine reconstituée.
  - Visser le bouchon operculé sur le flacon de STA-Néoplastine reconstituée.
- **STA-Coag Control N et P** :
  - Ouvrir les flacons.
  - Prélever avec une pipette et distribuer 1ml d'eau distillée dans chaque flacon de contrôle, puis reboucher les flacons.
  - Laisser Stabiliser 1 min à température ambiante.
  - Homogénéiser sans retourner le flacon.
  - Retirer les bouchons.
- **Tampon Owren Koller et le STA-CaCl2** :
  - Retirer les bouchons.
- **Pour préparer les 3 flacons de STA-Desorb-U** :
  - Retirer le bouchon en caoutchouc.
  - Insérer un maxi-reducer dans le flacon de Desorb U.
  - Visser le bouchon operculé sur le flacon de Desorb U.

A.2 Étape 2 : Chargement des produits

- Sur le tableau de bord, cliquer sur l'icône ≪ Produits ≫.
- Cliquer sur l'onglet ≪ Ouverture ≫.
- Scanner le premier produit à charger.
- Cliquer sur la case dans la colonne ≪ Position ≫ pour savoir dans quelle zone (R0 R1 et R2) et quelle position du tiroir charger le produit.
- Charger le produit dans une de ces positions.
- Répéter l'opération pour tous les flacons.
  Remarque : Il faut charger un Flacon de Desorb-U par zone : 1 en R0, 1 en R1 et 1 en R2.
- Quand tous les produits sont chargés, vérifier qu'il y ait bien une diode rouge d'allumée à coté de chaque flacon. Si ce n'est pas le cas recharger le flacon concerné.
- Cliquer sur l'onglet ≪ Fermer tiroir ≫.

A.3 Étape 3 : Lancement des Contrôles Qualités

- Cliquer sur l'icône « CQ ».
- Cliquer sur le test TP dans la liste sur le coté droit de l'écran.
- Cliquer sur le test TCA.
- Cliquer sur le test FIB.
- Cliquer sur le bouton de lancement.
- Valider par OK.
- Attendre environs 3 minutes que les CQ soient réalisés pour charger les échantillons :
  - Dans le menu « CQ », les coches passent au vert.
  - Dans le menu « Echantillons », sur l'onglet "Tableau de bord" les identités des CQ (12351 et 12352) disparaissent.

A.4 Étape 4 : Chargement des tubes échantillons

- Prendre 3 tubes échantillons.
- Prendre un rack.
- Positionner les tubes échantillons dans un rack.
- Positionner le rack sur le panier.
- Positionner le panier sur le STA-R®.
- Visualiser la position du premier tube sur le rack.
- Sélectionner cette position en cliquant sur la représentation du rack en bas de l'écran.
- Identifier le tube, à gauche de l'écran, dans le champ correspondant : Identité des tubes = vos initiales _ votre date de naissance _ la position dans le rack. Par exemple : AD_210880_1
- Sélectionner les tests TP TCA FIB sur la droite de l'écran, dans la liste « Etat des tests ».
- Réaliser ces 4 opérations pour tous les tubes chargés dans le rack.
- Cliquer sur l'icône « Rack » pour faire entrer le rack.
- Attendre environ 5 min la fin des analyses pour passer à l'étape suivante, il doit y avoir un résultat dans chaque case sur le tableau de bord.

A.5 Étape 5 : Déchargement du rack

- Cliquer sur l'icône « Echantillons » en haut de l'écran.
- Cliquer sur l'onglet « (Dé)Chargement ».
- Cocher le rack présent sur la gauche de l'écran.
- Cliquer sur le bouton « Rack ».
- Confirmer en cliquant sur OK.
- Retirer le panier du STA-R® et le poser sur la paillasse.
- Retirer le rack du panier.
- Retirer les tubes des racks.

A.6 Étape 6 : Déchargement des produits

– Sur le tableau de bord, cliquer sur l'icône « Produits ».
– Cliquer sur l'onglet « Ouverture ».
– Retirer tous les flacons du tiroir, et les poser sur le STA-R®.
– Refermer le tiroir en cliquant sur l'onglet « Fermer tiroir ».

Bibliographie

Ailiya, Z. Shen, et C. Miaoc. An emotional agent in virtual learning environment. In *Transactions on Edutainment IV, LNCS 6250*, pages 22–33, 2010.

E. Aimeur, C. Frasson, et H. Dufort. Cooperative learning strategies for intelligent tutoring systems. In *Applied Artificial Intelligence*, volume 14, pages 465–489, 2000.

H. Allert. Coherent social systems for learning : An approach for contextualized and community-centred metadata. In *Journal of Interactive Media in Education*, volume 2, 2004.

J. Anderson. The architecture of cognition. Cambridge, Massachussetts, 1983.

J. Anderson. Analysis of student performance with the lisp tutor. In *Diagnostic monitoring of skill and knowledge acquisition.*, pages 27–50, Hillsdale, NJ : Erlbaum, 1990.

J. Anderson. *Rules of the mind.* Hillsdale, NJ : Erlbaum., 1993.

J. Anderson. *Learning and memory : An integred approach.* John Wiley and Sons, 1995.

Q. Baishuang et Z. Wei. Student model in adaptative learning system based on semantic web. In *First International Workshop on Education Technology and Computer Science*, 2009.

K. Barber et J. Kim. Belief revision process based on trust : Simulation experiments. In *4th Workshop on Deception, Fraud and Trust in Agent Societies*, Montreal Canada, 2004.

B. Bauer, J. Muller, et J. Odell. Agent uml : A formalism for specifying multiagent software systems. In *Agent-Oriented Sofware Engineering*, volume 1957 of *Lecture Notes in Computer Science*, pages 109–120. Springer Berlin / Heidelberg, 2001.

L. Braubach, A. Pokahr, et W. Lamersdorf. Jadex : A bdi-agent system combining middleware and reasoning. In *Software Agent-Based Applications, Platforms and Development Kits. Birkhaeuser*, pages 143–168, 2005.

E. Bruillard. *Les machines à enseigner.* ISBN : 2-86601-610-6. Hermès Paris, 1997.

C. Buche et R. Querrec. An expert system manipulating knowledge to help human learners into virtual environment. *Expert Systems with Applications*, 38 :8446–8457, 2011.

C. Buche, C. Bossard, R. Querrec, et P. Chevaillier. Pegase : A generic and adaptable intelligent system for virtual reality learning environments. *International Journal of Virtual Reality*, 9(2) :73–85, 2010.

G. Butterfield, E.C. & Nelson. Theory and practice of teaching for transfer. In *Educational Technology Research and Development*, volume 37, pages 5–38, 1989.

V. K. Chaudhri et R. Fikes. Okbc : A programmatic foundation for knowledge base interoperability. In *Proceedings of AAAI-98*, pages 600–607. AAAI Press, 1998.

M. Dastani, B. van Riemsdijk, F. Dignum, et J. Meyer. A programming language for cognitive agents : Goal directed 3apl. In *Programming Multi-Agent Systems : First International Workshop*, volume 3067, pages 111–130, 2003.

A. De Antonio, J. Ramirez, R. Imbert, et G. Mendez. Intelligent virtual environments for training : An agent-based approach. In *CEEMAS, LNAI 3690*, pages 82–91, 2005.

E. De Sevin, R. Niewiadomski, E. Bevacqua, A.-M. Pez, M. Mancini, et C. Pelachaud. Greta : Une plateforme d'agent conversationnel expressif et interactif. In *Technique et science informatiques*, 2009.

C. T. Dos Santos et F. S. Osorio. Integrating intelligent agents, user models, and automatic content categorization in a virtual environment. In *ITS 2004, LNCS 3220*, pages 128–139, 2004.

L. Ehrler et S. Cranefield. Executing agent uml diagrams. In *Autonomous Agent and Multi-Agent System 2004*, pages 906–913, New York, USA, july 2004.

N. El-Kechai. *Suivi et assistance des apprenants dans les environnements virtuels de formation.* PhD thesis, Universite du Maine, France, 2007.

N. El-Kechai et C. Desprès. Proposing the underlying causes that lead to the trainee's erroneous actions to the trainer. In *EC-TEL : European Conference on Technology Enhanced Learning*, volume 4753, pages 41–55, 2007.

J. Ferber et O. Gutknecht. A meta-model for the analysis and desing of organizations in multi-agent systems. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, pages 128–135, 1998.

J. Ferber et O. Gutknecht. Operational semantics of multi-agent organizations. In N. Jennings et Y. Lespérance, editors, *Intelligent Agents VI. Agent Theories Architectures, and Languages*, volume 1757 of *Lecture Notes in Computer Science*, pages 205–217. Springer Berlin / Heidelberg, 2000.

C. Ferraris, A. Lejeune, L. Vignollet, et J.-P. David. Modélisation de scénarios pédagogiques collaboratifs. In *Conférence EIAH*, 2005.

L. Fricoteaux, I. M. Thouvenin, D. Lourdeaux, A. Pourplanche, et F. Hissel. Guidage auto-adaptatif par des metaphores de visualisation pour l'apprentissage à la navigation fluviale en environnement virtuel informe. In *5èmes journées de l'AFRV, Orsay, France*, 2010.

P. Fuchs. *Les interfaces de réalité virtuelle.* Les Presses de l'École des Mines de Paris, 1996.

S. Gerbaud, N. Mollet, F. Ganier, B. Arnaldi, et J. Tisseau. Gvt : a platform to create virtual environments for procedural training. *IEEE Virtual Reality*, 1(1) :225–232, 2008.

V. Guéraud. Une approche auteur pour les scénarios d'activités. In I. N. de Recherche Pèdagogique, editor, *Scénariser l'enseignement et l'apprentissage : une nouvelle compétence pour le praticien*, pages 33–38, 2006.

V. Guéraud, J.-M. Adam, J.-P. Pernin, G. Calvary, et J.-P. David. L'exploitation d'objets pédagogiques interactifs à distance : le projet FORMID. *STICEF*, 11 :103–163, May 2004.

M. Hannoun, O. Boissier, J. Sichman, et C. Sayettat. Moise : An organizational model for multi-agent systems. In *LNCS(LNAI)*, volume 1952, pages 156–165, 2000.

F. Harrouet, E. Cazeaux, et T. Jourdan. ARéVi. In P. Fuchs, G. Moreau, et J. Tisseau, editors, *Le traité de la Réalité Virtuelle*, volume 3, pages 369–392. Les Presses de l'École des Mines, 3e edition, 2006. ISBN ISBN : 2-911762-64-9.

F. Henri, C. Compte, et B. Charlier. La scénarisation pédagogique dans tous ses débats. In *Revue internationale des technologies en pédagogie universitaire*, volume 4, pages 14–24, 2007.

C. Hoareau, F. Le Corre, C. Buche, R. Querrec, et F. Ganier. Evaluation de l'utilité et de l'efficacité d'un environnement virtuel pour l'apprentissage de procédures. In *7ème conférence de Psychologie Ergonomique (EPIQUE)*, 2013a.

C. Hoareau, F. Le Corre, R. Querrec, C. Buche, et F. Ganier. Evolution of cognitive load when learning a procedure in a virtual environment for training. In *6th International Cognitive Load Theory Conference*, 2013b.

E. Hollnagel. Cognitive reliability and error analysis method. In *Oxford : Elsevier Science Ltd*, 1998.

J. Hubner, O. Boissier, R. Kitio, et A. Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agent and Multi-Agent System*, 20(3) : 369–400, 2010.

M.-P. Huget et J. Odell. Representing agent interaction protocols with agent uml. In *Autonomous Agent and Multi-Agent System 2004*, pages 1244–1245, New York, USA, july 2004.

R. Imbert, L. Sanchez, A. de Antonio, G. Mendez, et J. Ramirez. A multiagent extension for virtual reality based intelligent tutoring systems. In *Seventh IEEE International Conference on Advanced Learning Technologies*, 2007.

W. V. Joolingen et T. de Jong. Simquest, authoring educational simulations. In *Authoring tools for advanced technology educational software : Toward cost-effective production of adaptive, interactive, and intelligent educational software*, pages 1–31, 2003.

M. Kallmann et D. Thalmann. Modeling objects for interaction tasks. In *Proceedings of Computer Animation and Simulation'98*, pages 73–86, 1998.

A. Katasonov et V. Terziyan. Semantic agent programming language (s-apl) : A middleware platform for the semantic web. In *The IEEE International Conference on Semantic Computing*, 2008.

D. Kolb. *Experiential learning : experience as the source of learning and development*. Prentice Hall, Englewood Cliffs, NJ, 1984.

R. Koper. Modeling units of study from a pedagogical perspective. In *Educational Technology Expertise Centre, Open University of the Netherlands*, 2001.

R. Koper, B. Olivier, et T. Anderson. Ims learning design information model. In *IMS Global Learning Consortium*, 2003.

P. Laforcade. *Méta-modélisation UML pour la conception et la mise en œuvre de situations–problèmes coopératives*. Thèse de doctorat, Université de Pau et des Pays de l'Adour, 2004.

J. Laliberté. Comment faciliter le transfert des apprentissages. In *Pédagogie collégiale*, volume 3, pages 30–32, 1990.

F. Le Corre, C. Fauvel, C. Hoareau, R. Querrec, et C. Buche. Chrysaor : an agent-based intelligent tutoring system in virtual environment. In *ICVL 2012, Romania*, 2012.

S. Leman, P. Marcenac, et S. Giroux. Un modèle multi-agents de l'apprenant. In *Sciences et Techniques Educatives*, volume 3, pages 465–483, 1996.

X. Li, R. Liu, et M. Li. Construction of the intelligent tutoring system from the view of distributed cognition. In *ICWL 2010, LNCS 6483*, pages 171–179, 2010a.

X. Li, F. Ma, S. Zhong, L. Tang, et Z. Han. Research on virtual experiment intelligent tutoring system based on multi-agent. In *Edutainment 2010, LNCS 6249*, pages 100–110, 2010b.

D. Lourdeaux. *Réalité virtuelle et formation : conception d'environnements virtuels pédagogiques*. Thèse de doctorat, Ecole des mines de Paris, 2001.

D. Lourdeaux, P. Fuchs, et J.-M. Burkhardt. An intelligent tutorial agent for training virtual environments. In *5th world multiconference on Systemics, Cybernetics and Informatics SCI'01, Orlando, Floride*, 2001.

D. Lourdeaux, J.-M. Burkhardt, et D. M. d'Huart. Aplg project : a library for learning virtual environments. In *Actes de la Conférence VRIC : Virtual Reality International Conference*, pages 109–112, 2005.

P. Maret et J. Calmet. Agent-based knowledge communities. In *International Journal of Computer Science and Applications*, volume 6, pages 1–18, 2009.

N. Marion. *Modèlisation de scénarios pédagogiques pour les environnements de réalité virtuelle d'apprentissage humain*. PhD thesis, Université Européenne de Bretagne, 2010.

N. Marion, R. Querrec, et P. Chevaillier. Integrating knowledge from virtual reality environments to learning scenario models. a meta-modeling approach. In *International conference of computer supported education*, pages 254–259, 2009.

C. Martel, L. Vignollet, et C. Ferraris. Ldl : un langage support à la scénarisation pédagogique. In I. N. de Recherche Pédagogique, editor, *Scénariser l'enseignement et l'apprentissage : une nouvelle compétence pour le praticien*, pages 51–56, 2006.

L. Montealegre Vazquez et F. Lopez y Lopez. An agent-based model for hierarchical organizations. In P. Noriega, J. Vazquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara, et E. Matson, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, volume 4386 of *Lecture Notes in Computer Science*, pages 194–211. Springer Berlin / Heidelberg, 2007.

A. Munro, M. Johnson, Q. Pizzini, D. Surmon, D. Towne, et J. Wogulis. Authoring simulation-centered tutors with rides. In *International Journal of Artificial Intelligence in Education*, volume 8, pages 284–316, 1997.

L. A. Nguyen. The modal logic programming system mprolog. In *JELIA 2004, LNAI 3229*, pages 266–278, 2004.

A. Omicini et A. Ricci. Mas organization within a coordination infrastructure : Experiments in tucson. In A. Omicini, P. Petta, et J. Pitt, editors, *Engineering Societies in the Agents World*, volume 3071 of *Lecture Notes in Computer Science*, pages 520–520. Springer Berlin / Heidelberg, 2004.

A. Ortiz, D. Oyarzun, et M. del Puy Carretero. Elein : E-learning with 3d interactive emotional agents. In *Edutainment 2009, LNCS 5670*, pages 294–305, 2009.

H. V. D. Parunak et J. Odell. Representing social structures in UML. In *Agent Oriented Software Engineering Workshop (AOSE 2001), International Conference on Autonomous Agents*, volume 2222 of *Lecture Notes in Computer Science*, pages 1–16. Springer–Verlag (Berlin), 2002.

H. Peng. The application of affective virtual human in intelligent web tutoring system. In *ICAT 2006, LNCS 4282*, pages 21–27, 2006.

S. Pesty et C. Webber. The baghera multiagent learning environment : An educational community of artificial and human agents. In *UPGRADE*, volume 5, 2004.

D.-M. Popovici, J.-P. Gerval, P. Chevaillier, J. Tisseau, L.-D. Serbanati, et P. Guéguen. Educative distributed virtual environments for children. *Journal of Distance Education Technologies*, 2(4) :18–40, 2004.

J. Portillo-Rodriguez, A. Vizcaino, J. P. Soto, M. Piattini, et G. N. Aranda. Fostering knowledge exchange in virtual communities by using agents. In *CRIWG 2007, LNCS 4715*, pages 32–39, 2007.

R. Prawat. Promoting access to knowledge, strategy and disposition in students : A research synthesis. In *Review of Educational Research*, volume 59, pages 1–41, 1989.

D. Py. Quelques méthodes d'intelligence artificielle pour la modélisation de l'élève. In *Sciences et Techniques Educatives*, volume 5, pages 123–140, 1998.

R. Querrec, C. Buche, F. Le Corre, et F. Harrouet. Agent metamodel for virtual reality applications. In *International Symposium on Methodologies for Intelligent Systems*, 2011.

M. J. Raphael et S. A. Deloach. A knowledge base for knowledge-based multiagent system construction. In *National Aerospace and Electronics Conference (NAECON)*, 2000.

J. Rickel et W. L. Johnson. Animated agents for procedural training in virtual reality : Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13 :343–382, 1999.

G. Rimassa. *Runtime Support for Distributed Multi-Agent Systems*. PhD thesis, University of Parma, 2003.

M. Rodrigues, P. Novais, et F. Santos. Future challenges in intelligent tutoring systems - a framework. In *Formatex*, 2005.

P. V. Rosmalen. Sam, simulation and multimedia. In *Design and production of multimedia and simulation-based learning material*, pages 167–187, 1994.

P. S. Sajja. Multi-agent system for knowledge-based access to distributed databases. In *Interdisciplinary Journal of Information, Knowledge, and Management*, volume 3, 2008.

L. Sanchez et R. Imbert. An agent-based adaptable and configurable tutoring module for intelligent virtual environments for training. In *Edutainment 2007, LNCS 4469*, pages 499–510, 2007.

B. Schiemann. Owl dl as a fipa acl content language. In *The 18th ESSLLI*, Málaga, Spain, 2006.

R. Shi et P. Lu. A multi-criteria programming model for intelligent tutoring planning. In *KES 2006, Part I, LNAI 4251*, pages 780–787, 2006.

B. Sorensen et S. Ramachandran. Simulation-based automated intelligent tutoring. In *Human Interface, Part II, HCII 2007, LNCS 4558*, pages 466–474, 2007.

R. M. Taylor, T. Hudson, A. Seeger, H. Weber, J. Juliano, et A. Helser. Vrpn : A device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology. VRST 2001*, 2001.

V. Torres DaSilva, R. Choren, et C. J.P. De Lucena. A uml based approach for modeling and implementing multi-agent systems. In *Autonomous Agent and Multi-Agent System 2004*, pages 914–921, New York, USA, july 2004.

L. Toupin. L'école a-t-elle un miroir ? le transfert de connaissance comme outil de réflexion sur le projet scolaire. In *Vie Pédagogique*, volume 81, pages 46–49, 1993.

L. Toupin. De la formation au métier : Savoir transférer ses connaissances dans l'action. Paris, 1995.

H. e. a. Vilhjalmsson. The behavior markup language : Recent developments and challenges. In *C. Pelachaud et al. (Eds.) : IVA 2007, LNAI 4722*, pages 99–111, 2007.

L. Wang et W. Yu. Research on knowledge communication of dynamic virtual communities based on ontology. In *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*, 2005.

C. Webber, S. Pesty, et N. Balacheff. A multi-agent and emergent approach to learner modelling. In *European Conference on Artificial Intelligence*, 2002.

E. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann, Los Altos, California, 1987.

L. Winkelhagen, M. Dastani, et J. Broersen. Beliefs in agent implementation. 2005.

B. Woolf. Building knowledge based tutors. In *Computer Assisted Learning : the 4th International Conferences (ICCAL'92)*, pages 46–60, 1992.

K. Zakharov, A. Mitrovic, et L. Johnston. Towards emotionally-intelligent pedagogical agents. In *ITS 2008, LNCS 5091*, pages 19–28, 2008.

F. Zambonelli, N. Jennings, et M. Wooldridge. Developing multiagent systems : The gaia methodology. In *ACM Transaction on Software Engineering Methodology*, volume 12, pages 317–370, 2003.

— **Résumé** —

Ce travail de recherche s'inscrit dans le cadre des environnements virtuels pour la formation aux instruments de diagnostic biomédical. La simulation de la situation réelle seule ne suffisant pas, il convient d'incorporer un scénario pédagogique pour construire l'apprentissage. Le scénario s'applique pour tous les apprenants, il est donc important d'intégrer des possibilités d'individualisation. Pour cela, nous proposons de coupler l'environnement virtuel à un Système Tutoriel Intelligent (STI). Ceci permet à la pédagogie de s'adapter dynamiquement à l'apprenant au cours de la formation : la pédagogie sera propre à chaque individu.

Notre étude débute par une identification des systèmes existants qui nous semblent les mieux adaptés, sur lesquels baser notre proposition : le STI Pegase, le modèle de scénario pédagogique Poseidon et le méta-modèle Mascaret. Ceci nous permet également d'identifier les limites de Pegase qui sont : le manque de lien avec le scénario pédagogique, le manque de modularité et le manque d'individualisation. Notre proposition, appelée Chrysaor, vise à combler ces faiblesses.

Pour cela, nous étendons le méta-modèle Mascaret. Nous proposons ainsi qu'un scénario pédagogique soit une connaissance explicite et par conséquent exploitable par un STI. Les concepts proposés dans notre modèle permettent également au formateur de facilement modifier les comportements, les rôles et les affectations (humain ou agent autonome) : notre système devient ainsi modulaire. Notre proposition répond ainsi aux problèmes identifiés dans notre état de l'art.

Nous illustrons ensuite l'utilisation de Chrysaor dans une application de formation par la réalité virtuelle à un instrument de diagnostic en hémostase utilisé en milieu hospitalier. Sur cette application, des expérimentations comparant les deux types de formation (traditionnelle et virtuelle) permettent de vérifier la qualité de l'apprentissage obtenu par une formation en réalité virtuelle, et de vérifier que ces connaissances sont transférables en environnement réel.

Mots clefs : environnement virtuel d'apprentissage humain, système multi-agents, système tutoriel intelligent, formation, réalité virtuelle.